Simple and Fast Approximate Counting and Leader Election in Populations*

Othon Michail[1], Paul G. Spirakis[1,2], and Michail Theofilatos[1]

[1] Department of Computer Science, University of Liverpool, UK
[2] Computer Engineering and Informatics Department, University of Patras, Greece
Email: {Othon.Michail, P.Spirakis, Michail.Theofilatos}@liverpool.ac.uk

Abstract. We study the problems of leader election and population size counting for *population protocols*: networks of finite-state anonymous agents that interact randomly under a uniform random scheduler. We provide simple protocols for approximate counting of the size of the population and for leader election. We show a protocol for leader election that terminates in $O(\frac{\log^2 n}{\log m})$ parallel time, where $1 \leq m \leq n$ is a parameter, using $O(\max\{m, \log n\})$ states. By adjusting the parameter m between a constant and n, we obtain a single leader election protocol whose time and space can be smoothly traded off between $O(\log^2 n)$ to $O(\log n)$ time and $O(\log n)$ to $O(n)$ states. We also give a protocol which provides an upper bound $\hat{n}$ of the size n of the population, where $\hat{n}$ is at most n^a for some constant $a > 1$. This protocol assumes the existence of a unique leader in the population and stabilizes in $\Theta(\log n)$ parallel time, using constant number of states in every node, except from the unique leader which is required to use $\Theta(\log^2 n)$ states.

Keywords: population protocol, epidemic, leader election, counting, approximate counting, polylogarithmic time protocol

1 Introduction

Population protocols [1] are networks that consist of very weak computational entities (also called *nodes* or *agents*), regarding their individual capabilities. These networks have been shown that are able to construct complex shapes [2] and perform complex computational tasks when they work collectively. Leader Election, which is a fundamental problem in distributed computing, is the process of designating a single agent as the coordinator of some task distributed among several nodes. The nodes communicate among themselves in order to decide which of them will get into the *leader* state. *Counting* is also a fundamental

* All authors were supported by the EEE/CS initiative NeST. The last author was also supported by the Leverhulme Research Centre for Functional Materials Design. This work was partially supported by the EPSRC Grant EP/P02002X/1 on Algorithmic Aspects of Temporal Graphs.

problem in distributed computing, where nodes must determine the size n of the population. Finally, we call *Approximate Counting* the problem in which nodes must determine an estimation k of the population size n. Counting can be then considered as a special case of population size estimation, where $k = n$.

Many distributed tasks require the existence of a leader prior to the execution of the protocol and, furthermore, some knowledge about the system (for instance the size of the population) can also help to solve these tasks more efficiently with respect both to time and space.

Consider the setting in which an agent is in an initial state a, the rest $n-1$ agents are in state b and the only existing transition is $(a, b) \to (a, a)$. This is the *one-way epidemic* process and it can be shown that the expected time to convergence under the uniform random scheduler is $\Theta(n \log n)$ (e.g., [3]), thus $\Theta(\log n)$ *parallel time*. Here, parallel time is the total number of interactions divided by n. In this work, we make an extensive use of epidemics, which means that information is being spread throughout the population, thus all nodes will obtain this information in $O(\log n)$ expected parallel time. We use this property to construct an algorithm that solves the *Leader Election* problem. In addition, by observing the rate of the epidemic spreading under the uniform random scheduler, we can extract valuable information about the population. This is the key idea of our *Approximate Counting* algorithm.

1.1 Related Work

The framework of population protocols was first introduced by Angluin et al. [1] in order to model the interactions in networks between small resource-limited mobile agents. When operating under a uniform random scheduler, population protocols are formally equivalent to a restricted version of stochastic Chemical Reaction Networks (CRNs), which model chemistry in a well-mixed solution [4]. "CRNs are widely used to describe information processing occurring in natural cellular regulatory networks, and with upcoming advances in synthetic biology, CRNs are a promising programming language for the design of artificial molecular control circuitry" [5,6]. Results in both population protocols and CRNs can be transfered to each other, owing to a formal equivalence between these models.

Angluin et al. [7] showed that all predicates stably computable in population protocols (and certain generalizations of it) are semilinear. Semilinearity persists up to $o(\log \log n)$ local space but not more than this [8]. Moreover, the computational power of population protocols can be increased to the commutative subclass of $\mathbf{NSPACE}(n^2)$, if we allow the processes to form connections between each other that can hold a state from a finite domain [9], or by equipping them with unique identifiers, as in [10]. For introductory texts to population protocols the interested reader is encouraged to consult [11,9] and [12] (the latter discusses population protocols and related developments as part of a more general overview of the emerging theory of dynamic networks).

Optimal algorithms, regarding the time complexity of fundamental tasks in distributed networks, for example leader election and majority, is the key for many distributed problems. For instance, the help of a central coordinator can

lead to simpler and more efficient protocols [3]. There are many solutions to the problem of leader election, such as in networks with nodes having distinct labels or anonymous networks [13,14,15,16,17].

Although the availability of an initial leader does not increase the computational power of standard population protocols (in contrast, it does in some settings where faults can occur [18]), still it may allow faster computation. Specifically, the fastest known population protocols for semilinear predicates without a leader take as long as linear parallel time to converge ($\Theta(n)$). On the other hand, when the process is coordinated by a unique leader, it is known that any semilinear predicate can be stably computed with polylogarithmic expected convergence time ($O(\log^5 n)$) [19].

For several years, the best known algorithm for leader election in population protocols was the pairwise-elimination protocol of Angluin et al. [1], in which all nodes are leaders in state l initially and the only effective transition is $(l, l) \to (l, f)$. This protocol always stabilizes to a configuration with unique leader, but this takes on average linear time. Recently, Doty and Soloveichik [20] proved that not only this, but any standard population protocol requires linear time to solve leader election. This immediately led the research community to look into ways of strengthening the population protocol model in order to enable the development of sub-linear time protocols for leader election and other problems (note that Belleville, Doty, and Soloveichik [21] recently showed that such linear time lower bounds hold for a larger family of problems and not just for leader election). Fortunately, in the same way that increasing the local space of agents led to a substantial increase of the class of computable predicates [8], it has started to become evident that it can also be exploited to substantially speed-up computations. Alistarh and Gelashvili [15] proposed the first sub-linear leader election protocol, which stabilizes in $O(\log^3 n)$ parallel time, assuming $O(\log^3 n)$ states at each agent. In a very nice work, Gasieniec and Stachowiak [16] designed a space optimal ($O(\log \log n)$ states) leader election protocol, which stabilises in $O(\log^2 n)$ parallel time. They use the concept of phase clocks (introduced in [3] for population protocols), which is a synchronization and coordination tool in distributed computing. General characterizations, including upper and lower bounds, of the trade-offs between time and space in population protocols were recently achieved in [22]. Moreover, some papers [23,24] have studied leader election in the mediated population protocol model.

For counting, the most studied case is that of *self-stabilization*, which makes the strong adversarial assumption that arbitrary corruption of memory is possible in any agent at any time, and promises only that eventually it will stop. Thus, the protocol must be designed to work from any possible configuration of the memory of each agent. It can be shown that counting is *impossible* without having one agent (the "base station") that is protected from corruption [25]. In this scenario $\Theta(n \log n)$ time is sufficient [26] and necessary [27] for self-stabilizing counting.

In the less restrictive setting in which all nodes start from the same state (apart possibly from a unique leader and/or unique ids), not much is known.

In a recent work, Michail [28] proposed a terminating protocol in which a pre-elected leader equipped with two n-counters computes an approximate count between $n/2$ and n in $O(n \log n)$ parallel time with high probability. The idea is to have the leader implement two competing processes, running in parallel. The first process counts the number of nodes that have been encountered once, the second process counts the number of nodes that have been encountered twice, and the leader terminates when the second counter catches up the first. In the same paper, also a version assuming unique ids instead of a leader was given.

A uniform protocol for exact population counting, but much more complicated than here is provided by our team and other co-authors in [29].

The task of counting has also been studied in the related context of worst-case dynamic networks [30,31,32,33,34].

1.2 Contribution

In this work we employ the use of simple epidemics in order to provide efficient solutions to approximate counting the size of a population of agents and also to leader election in populations. Our model is that of population protocols. Our goal for both problems is to get polylogarithmic parallel time and to also use small memory per agent. First, we show how to approximately count a population fast (with a leader) and then we show how to elect a leader (very fast) if we have a crude population estimate.

(a) We start by providing a protocol which provides an upper bound $\hat{n}$ of the size n of the population, where $\hat{n}$ is at most n^a for some $a > 1$. This protocol assumes the existence of a unique leader in the population. The runtime of the protocol until stabilization is $\Theta(\log n)$ parallel time. Each node except from the unique leader uses only a constant number of states. However, the leader is required to use $\Theta(\log^2 n)$ states.

(b) We then look into the problem of electing a leader. We assume an approximate knowledge of the size of the population (i.e., an estimate $\hat{n}$ of at most n^a, where n is the population size) and provide a protocol (parameterized by the size m of a counter for drawing local random numbers) that elects a unique leader w.h.p. in $O(\frac{\log^2 n}{\log m})$ parallel time, with number of states $O(\max\{m, \log n\})$ per node.

(c) Finally, we combine our two protocols in order to provide a *size oblivious* protocol which elects a leader in $O(\frac{\log^2 n}{\log m})$ parallel time.

2 The model

In this work, the system consists of a population V of n distributed and anonymous (i.e., do not have unique IDs) *processes*, also called *nodes* or *agents*, that are capable to perform local computations. Each of them is executing as a deterministic state machine from a finite set of states Q according to a transition function $\delta : Q \times Q \to Q \times Q$. Their interaction is based on the probabilistic (uniform random) scheduler, which picks in every discrete step a random edge from the complete graph G on n vertices. When two agents interact, they mutually

access their local states, updating them according to the transition function δ. The transition function is a part of the population protocol which all nodes store and execute locally.

The time is measured as the number of steps until stabilization, divided by n (parallel time). The protocols that we propose do not enable or disable connections between nodes, in contrast with [2], where Michail and Spirakis considered a model where a (virtual or physical) connection between two processes can be in one of a finite number of possible states. The transition function that we present throughout this paper, follows the notation $(x, y) \rightarrow (z, w)$, which refers to the process states before *(x and y)* and after *(z and w)* the interaction, that is, the transition function maps pairs of states to pairs of states.

The Leader Election Problem. The problem of leader election in distributed computing is for each node eventually to decide whether it is a leader or not subject to only one node decides that it is the leader. An algorithm A solves the leader election problem if eventually the states of agents are divided into *leader* and *follower*, a leader remains elected and a follower can never become a leader. In every execution, exactly one agent becomes leader and the rest determine that they are not leaders. All agents start in the same initial state q and the output is $O = \{leader, follower\}$. A randomized algorithm R solves the leader election problem if eventually only one leader remains in the system w.h.p.

Approximate Counting Problem. We define as *Approximate Counting* the problem in which a leader must determine an estimation $\hat{n}$ of the population size, where $\frac{\hat{n}}{a} < n < \hat{n}$. We call the constant a the estimation parameter.

3 Fast Counting with a unique leader

In this section we present our *Approximate Counting* protocol. The protocol is presented in Section 3.1. In Section 3.2 we prove the correctness of our protocol and finally, in Section 5, experiments that support our analysis can be found.

3.1 Abstract description and protocol

In this section, we construct a protocol which solves the problem of approximate counting. Our probabilistic algorithm for solving the approximate counting problem requires a unique leader who is responsible to give an estimation on the number of nodes. It uses the epidemic spreading technique and it stabilizes in $O(\log n)$ parallel time. There is initially a unique leader l and all other nodes are in state q. The leader l stores two counters in its local memory, initially both set to 0. We use the notation $l_{(c_q, c_a)}$, where c_q is the value of the first counter and c_a is the value of the second one. The leader, after the first interaction starts an epidemic by turning a q node into an a node. Whenever a q node interacts with an a node, its state becomes a $((a, q) \rightarrow (a, a))$. The first counter c_q is being used for counting the q nodes and the second counter c_a for the a nodes, that is, whenever the leader l interacts with a q node, the value of the counter c_q is increased by one and whenever l interacts with an a node, c_a

is increased by one. The termination condition is $c_q = c_a$ and then the leader holds a constant-factor approximation of $\log n$, which we prove that with high probability is $2^{c_q+1} = 2^{c_a+1}$.
We first describe a simple terminating protocol that guarantee with high probability $n^{-a} \leq n_e \leq n^a$, for a constant a, i.e., the population size estimation is polynomially close to the actual size. Chernoff bounds then imply that repeating this protocol a constant number of times suffices to obtain $n/2 \leq n_e \leq 2n$ with high probability.

Protocol 1 Approximate Counting (APC)

$Q = \{q,\ a,\ l_{(c_q,c_a)}\}$
δ :
$(l_{(0,0)},\ q) \to (l_{(1,0)},\ a)$
$(a,\ q) \to (a,\ a)$
$(l_{(c_q,c_a)},\ q) \to (l_{(c_q+1,c_a)},\ q),\ if\ c_q > c_a$
$(l_{(c_q,c_a)},\ a) \to (l_{(c_q,c_a+1)},\ a),\ if\ c_q > c_a$
$(l_{(c_q,c_a)},\ \cdot) \to (halt,\ \cdot),\ if\ c_q = c_a$

3.2 Analysis

Lemma 1. *When half or less of the population has been infected, with high probability $c_q > c_a$. In fact, $c_q - c_a \approx \ln(n/2) - \sqrt{\log n} > 0$.*

The previous results show that the counter c_q is a function of n and with high probability greater than c_a until half of the population becomes infected. Chernoff bounds show that w.h.p. $c_q \approx \ln(n/2)$, while $c_a \approx \ln 2$ and w.h.p. $c_a < \sqrt{\log n}$.

Corollary 1. *APC does not terminate w.h.p. until more than half of the population becomes infected.*

When the infected agents are in the majority, c_q is increased by a small constant number, while c_a eventually catches up the first counter. The termination condition ($c_q = c_a$) is satisfied and the leader gives a constant-factor approximation of $\log n$. A proof can be found in the full version of the paper.

Lemma 2. *Our Approximate Counting protocol terminates after $\Theta(\log n)$ parallel time w.h.p..*

It takes $\Theta(\log n)$ parallel time for half agents to become infected. At that point, it holds that $|c_a - c_q| = O(\log n)$. When the a nodes are in the majority, this difference reaches zero after $\Theta(\log n)$ leader interactions. Thus, the total parallel time to termination is $\Theta(\log n)$. A proof can be found in the full version of the paper.

Lemma 3. *When half or less of the population has been infected, with high probability $c_q < \log(n/2) + \epsilon$ and $c_q > \log(n/2) - \epsilon$. When more than half of the population is infected, c_q is expected to increase by $\log 2$ and w.h.p. less than $\log n$.*

Corollary 2. *When $c_q = c_a$, w.h.p. 2^{c_q+1} is an upper bound on n.*

4 Leader Election with approximate knowledge of n

The existence of a *unique leader agent* is a key requirement for many population protocols [3] and generally in distributed computing, thus, having a fast protocol that elects a unique leader is of high significance. In this section, we present our *Leader Election* protocol, giving, at first, an abstract description 4.1, the algorithm 4.2 and then, we present the analysis of it 4.3. Finally, we have measured the stabilization time of this protocol for different population sizes and the results can be found in section 5.

4.1 Abstract description

We assume that the nodes know *an upper bound on the population size* n^b*, where* n *is the number of nodes and* b *is any big constant number.*
All nodes store three variables; the round e, a random number r and a counter c and they are able to compute random numbers within a predefined range $[1, m]$. We define two types of states; the leaders (l) and the followers (f). Initially, all nodes are in state l, indicating that they are all potential leaders. The protocol operates in rounds and in every round, the leaders compete with each other trying to survive (i.e., do not become followers). The followers just copy the *tuple* (r, e) from the leaders and try to spread it throughout the population. During the first interaction of two l nodes, one of them becomes follower, a random number between 1 and m is being generated, the leader enters the first round and the follower copies the round e and the random number r from the leader to its local memory. The followers are only being used for information spreading purposes among the potential leaders and they cannot become leaders again. Throughout this paper, n denotes the *population size* and m *the maximum number that nodes can generate.*

Information spreading. It has been shown that the epidemic spreading of information can accelerate the convergence time of a population protocol. In this work, we adopt this notion and we use the followers as the means of competition and communication among the potential leaders. All leaders try to spread their information (i.e., their round and random number) throughout the population, but w.h.p. all of them except one eventually become followers. We say that a node x wins during an interaction if one of the following holds:

- Node x is in a bigger round e.
- If they are both in the same round, node x has bigger random number r.

One or more leaders L are in the *dominant state* if their tuple (r_1, e_1) wins every other tuple in the population. Then, the tuple (r_1, e_1) is being spread as an epidemic throughout the population, independently of the other leaders' tuples (all leaders or followers with the tuple (r_1, e_1) always win their competitors). We also call leaders L the *dominant leaders.*

Transition to next round. After the first interaction, a leader l enters the first round. We can group all the other nodes that l can interact with into three independent sets.

- The first group contains the nodes that are in a bigger round or have a bigger random number, being in the same round as l. If the leader l interacts with such a node, it becomes follower.
- The second group contains the nodes that are in a smaller round or have a smaller random number, being in the same round as l. After an interaction with a node in this group, the other node becomes a follower and the leader increases its counter c by one.
- The third group contains the followers that have the same tuple (r, e) as l. After an interaction with a node in this group, l increases its counter c by one.

As long as the leader l survives (i.e., does not become a follower), it increases or resets its counter c, according to the transition function δ. When the counter c reaches $b \log n$, where n^b is the upper bound on the population size, it resets it and round r is increased by one. The followers can never increase their round or generate random numbers.

Stabilization. The protocol that we present stabilizes, as the whole population will eventually reach in a final configuration of states. To achieve this, when the round of a leader l reaches $\lceil \frac{2b \log n - \log(b \log^2 n)}{\log m} \rceil$, l stops increasing its round r, unless it interacts with another leader. This rule guarantees the stabilization of our protocol.

4.2 The protocol

In this section, we present our *Leader Election* protocol. We use the notation $p_{r,e}$ to indicate that node p has the random number r and is in the round e. Also, we say that $(r_1, e_1) > (r_2, e_2)$ if the tuple (r_1, e_1) wins the tuple (r_2, e_2). A tuple (r_1, e_1) wins the tuple (r_2, e_2) if $e_1 > e_2$ or if they are in the same round $(e_1 = e_2)$, it holds that $r_1 > r_2$.

4.3 Analysis

The leader election algorithm that we propose, elects a unique leader after $O(\frac{\log^2 n}{\log m})$ parallel time w.h.p.. To achieve this, the algorithm works in stages, called *epochs* throughout this paper and the number of potential leaders decreases exponentially between the epochs. An epoch i starts when any leader enters the ith round $(r = i)$ and ends when any leader enters the $(i+1)$th round $(r = i+1)$. Here we do the exact analysis for $m = \log n$. This can be generalized to any m between a constant and n.

Lemma 4. *During the execution of the protocol, at least one leader will always exist in the population.*

Protocol 2 Leader Election

$Q = \{l, f_{r,e}, l_{r,e}\} : r \in [1, m]$
δ :

#First interaction between two nodes. One of them becomes follower and the other remains leader. The leader generates a random number r and enters the first round ($e = 1$).
$(l, l) \to (l_{r,1}, f_{r,1})$

#A leader in round 0 always loses (i.e., becomes a follower) against a node in a higher round.
$(f_{r,e}, l) \to (f_{r,e}, f_{r,e})$
$(l_{r,e}, l) \to (l_{r,e}, f_{r,e}),\ l_{counter} = l_{counter} + 1$

#The winning node propagates its tuple. If a leader loses, it becomes follower.
$(f_{r,i}, f_{s,j}) \to (f_{k,l}, f_{k,l})$, **if** $(r, i) > (s, j)$ **then** $(k, l) = (r, i)$ **else** $(k, l) = (s, j)$
$(l_{r,i}, l_{s,j}) \to (l_{k,l}, f_{k,l}),\ l_{counter} = l_{counter} + 1,$ **if** $(r, i) \geq (s, j)$ **then** $(k, l) = (r, i)$ **else** $(k, l) = (s, j)$
$(l_{r,i}, f_{s,j}) \to (f_{s,j}, f_{s,j})$, **if** $(s, j) > (r, i)$
$(l_{r,i}, f_{s,j}) \to (l_{r,i}, f_{r,i}),\ l_{counter} = l_{counter} + 1,$ **if** $(r, i) > (s, j)$
$(l_{r,e}, f_{r,e}) \to (l_{k,j}, f_{k,j}),\ l_{counter} = l_{counter} + 1$

#When a leader increases its counter, the following code is being executed. It checks whether it has reached $c \log n$. If yes, it moves to the next round, generates a new random number and checks if it has reached the final round in order to terminate.
if $(l_{counter} = b \log n)$ **then**{
    Increase round;
    Generate a new random number between 1 and m;
    Reset counter to zero;
    if $(Round = \lceil \frac{2b \log n - \log(b \log^2 n)}{\log m} \rceil)$ **Stop increasing the round, unless you interact with a leader;**
}

Our protocol does not allow all nodes to become followers. A proof is in the full version of the paper.

Lemma 5. *Assume an epoch e and k leaders with the dominant tuple (r, e) in this epoch. The expected parallel time to convergence of their epidemic in epoch e is $\Theta(logn)$.*

Lemma 6. *If a counter c of a leader l reaches $b \log n$, its epidemic will have already been spread throughout the whole population w.h.p..*

The previous lemma implies that no leader enters the next round if the epidemic has not been spread throughout the whole population before. This is important as we need to ensure that a non-dominant leader becomes follower by the end of an epoch, otherwise, the number of leaders would not be decreased

exponentially between successive epochs. A proof can be found in the full version of the paper.

Theorem 1. *After $O(\frac{\log n}{\log m})$ epochs, there is a unique leader in the population w.h.p..*

The number of potential leaders decreases exponentially between the epochs, and after $O(\frac{\log n}{\log m})$, a unique leader exists in the population. A proof can be found in the full version of the paper.

Theorem 2. *Our Leader Election protocol elects a unique leader in $O(\frac{\log^2 n}{\log \log n})$ parallel time w.h.p..*

Proof. There are initially n leaders in the population. During an epoch e, by Lemma 5 the dominant tuple spreads throughout the population in $\Theta(\log n)$ parallel time, by Lemma 6 no (dominant) leader can enter to the next epoch if their epidemic has not been spread throughout the whole population before and by Theorem 1, there will exist a unique leader after $O(\frac{\log n}{\log m})$ epochs w.h.p., thus, for $m = b \log n$ the overall parallel time is $O(\frac{\log^2 n}{\log \log n})$. Finally, by Lemma 4, the unique leader can never become follower and according to the transition function in Protocol 2, a follower can never become leader again.
The rule which says the leaders stop increasing their rounds if $r >= \frac{2b \log n - \log (b \log^2 n)}{\log m}$, unless they interact with another leader, implies that the population stabilizes in $O(\frac{\log^2 n}{\log \log n})$ parallel time w.h.p. and when this happens, there will exist only one leader in the population and eventually, our protocol always elects a unique leader.

Remark 1. By adjusting m to be any number between a constant and n and conducting a very similar analysis we may obtain a single leader election protocol whose time and space can be smoothly traded off between $O(\log^2 n)$ to $O(\log n)$ time and $O(\log n)$ to $O(n)$ space.

4.4 Dropping the assumption of knowing $\log n$

Call a population protocol *size-oblivious* if its transition function does not depend on the population size. Our leader election protocol requires a rough estimate on the size of the population in order to elect a leader in polylogarithmic time, while our approximate counting protocol requires a unique leader who initiates the epidemic process and then gives an upper bound on the population size. In this section, we combine our Approximate Counting and Leader Election protocols in order to construct a size-oblivious protocol that elects a unique leader in $O(\frac{\log^2 n}{\log m})$ parallel time and can be executed in any uniform model of population protocols.
To combine our protocols, in the our new *Leader Election* algorithm, the nodes instead of using the c counter, as described in Section 4.1, they use two counters c_q and c_a. The first counter is being used in order to count the non-followers and

the latter to count the followers. Initially, $c_q = 1$ and $c_a = 0$. Let l be a leader with the tuple (r_1, e_1). As in Section 4, a tuple (r_1, e_1) is bigger that the tuple (r_2, e_2) if $r_1 > r_2$ or if $r_1 = r_2$ and $e_1 > e_2$. We can group all the other nodes that l can interact with into three independent sets.

- $(r_1, e_1) > (r_2, e_2)$. l increases its c_q counter by one.
- $(r_1, e_1) = (r_2, e_2)$. l increases its c_a counter by one.
- $(r_1, e_1) < (r_2, e_2)$. l becomes follower and resets its counters to zero.

When $c_q = c_a$ holds, l increases its round e_1 by one and resets c_q to one and c_a to zero. This process simulates the behavior of our *Approximate Counting* protocol, meaning that when $c_q = c_a$ holds, the epidemic of the dominant leaders will have been spread throughout the whole population. Regarding the termination condition, where $\log n$ is needed, the nodes store a variable s which contains the average value of c_q. To this end, whenever a leader enters from round e_1 to $e_1 + 1$, it updates the value of s as follows:

$$s = \frac{s(e_1 - 1) + c_q}{e_1} \tag{1}$$

where s is initially zero. When $e_1 = \lceil \frac{as}{\log m} \rceil$ holds ($a \geq 1$ is a small constant number), the leader stops increasing it's round and the population stabilizes in a configuration with a unique leader. Finally, we show that the variable s of the unique leader is a function of $\log n$. Even though we do not provide any proof of correctness of this protocol, in Section 5 we provide experiments that confirm this behavior.

5 Experiments

We have also measured the stabilization time of all of our protocols for different network sizes. We have executed them 100 times for each population size n, where $n = 2^i$ and $i = [4, 14]$. Regarding the *Leader Election* algorithm which assumes some knowledge on the population size, the results (Figure. 1) support our analysis and confirm its logarithmic behavior. In these experiments, the maximum number that the nodes could generate was $m = 100$. Finally, all executions elected a unique leader in $a\frac{\log^2 n}{\log m}$ parallel time.
The stabilization time of our *Approximate Counting with a unique leader* protocol is shown in Figure 2a. The algorithm always gives a constant factor approximation of $\log n$, as shown in Figure 2b. Moreover, in Figure 3, we show the values of the counters c_q and c_a, when half of the population has been infected by the epidemic. These experiments support our analysis, while the counter of infected nodes reaches a constant number and the counter of non-infected nodes reaches a value close to $\log n$.
Regarding our protocol for leader election with no knowledge of $\log n$, the results are shown in Figure 4. All executions elected a unique leader after $a\frac{\log^2 n}{\log m}$ parallel

time, as shown in Figure 4a. Finally, as shown in Figure 4b, the unique leader holds a constant factor upper bound on $\log n$ after $a\frac{\log^2 n}{\log m}$ parallel time.

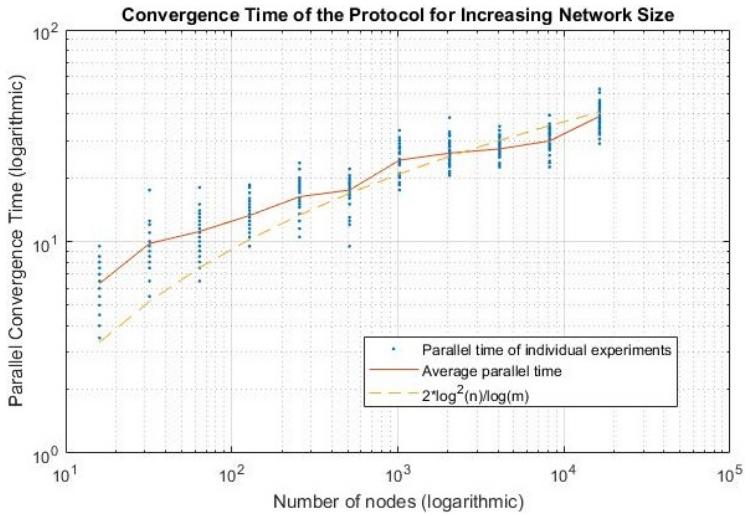
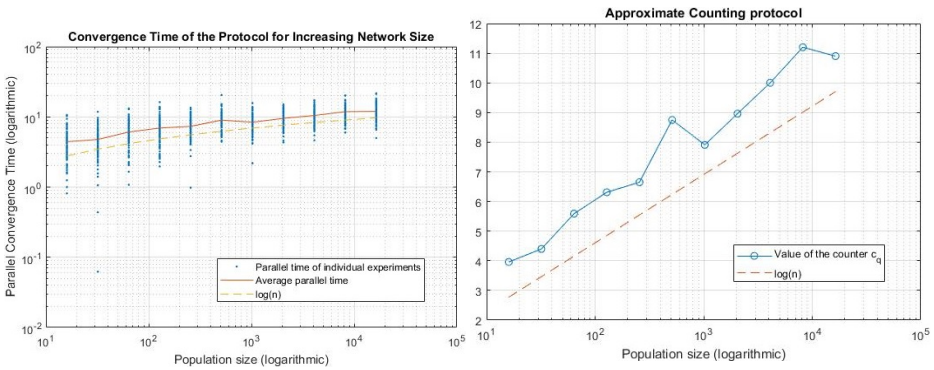


Fig. 1: Leader Election with approximate knowledge of n. Both axes are logarithmic. The dots represent the results of individual experiments and the line represents the average values for each network size.



(a) Convergence time.

(b) Estimations and actual sizes of the population.

Fig. 2: Approximate Counting with a unique leader.

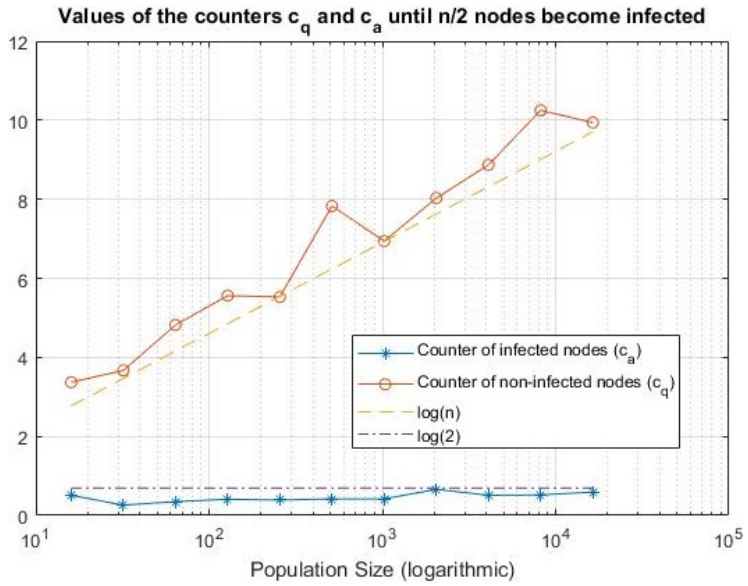
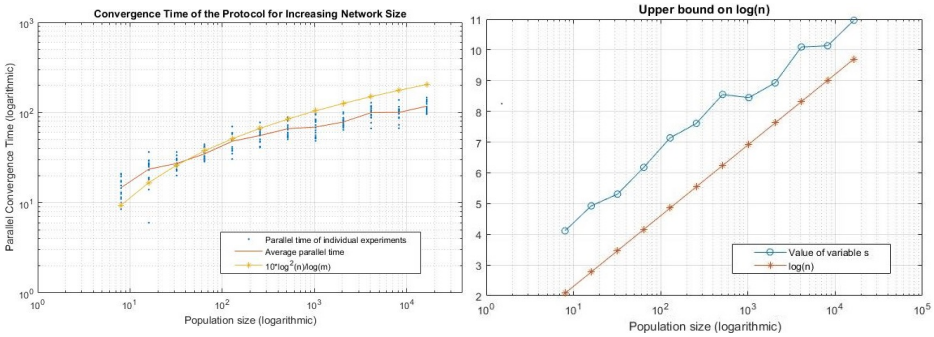


Fig. 3: Approximate Counting with a unique leader. Counters c_q and c_a when half of the population has been infected by the epidemic.



(a) Convergence time.

(b) Upper bounds and actual sizes of $\log n$

Fig. 4: Composition of our Approximate Counting and Leader Election protocols.

6 Open Problems

In our leader election protocol, when two nodes interact with each other, the amount of data which is transfered is $O(max\{\log\log n, \log m\})$ bits. In certain applications of population protocols, the processes are not able to transfer arbitrarily large amount of data during an interaction. Can we design a polylogarithmic time population protocol for the problem of leader election that satisfies this requirement?

Acknowledgments We would like to thank David Doty and Mahsa Eftekhari for their valuable comments and suggestions during the development of this research work.

References

1. Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, March 2006.
2. Othon Michail and Paul G. Spirakis. Simple and efficient local codes for distributed stable network construction. *Distributed Computing*, 29(3):207–237, 2016.
3. Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, 2008.
4. David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *Nat. Comput. 7*, 2008.
5. Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Nat. Comput. 7*, pages 517 – 534, 2014.
6. David Doty. Timing in chemical reaction networks. In *Proc. of the 25th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 772–784, 2014.
7. Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4), 2007.
8. Ioannis Chatzigiannakis, Othon Michail, Stavros Nikolaou, Andreas Pavlogiannis, and Paul G. Spirakis. Passively mobile communicating machines that use restricted space. *Theoretical Computer Science*, 412(46):6469–6483, October 2011.
9. Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. *New Models for Population Protocols*. N. A. Lynch (Ed), Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool, 2011.
10. Rachid Guerraoui and Eric Ruppert. Names trump malice: Tiny mobile agents can tolerate byzantine failures. In *ICALP*. Springer, 2009.
11. James Aspnes and Eric Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*. Springer-Verlag, 2009.
12. Othon Michail and Paul G Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2), 2018.
13. Dana Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th annual ACM symposium on Theory of computing (STOC)*. ACM, 1980.
14. Chagit Attiya, Marc Snir, and Manfred Warmuth. Computing on an anonymous ring. PODC '85. ACM, 1985.
15. Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2015.
16. Leszek Gasieniec and Grzegorz Stachowiak. Fast space optimal leader election in population protocols. In *SODA 2018: ACM-SIAM Symposium on Discrete Algorithms*, 2018. to appear.
17. Michael Fischer and Hong Jiang. Self-stabilizing leader election in networks of finite-state anonymous agents. *OPODIS*, vol 4305, 2006.
18. Giuseppe Antonio Di Luna, Paola Flocchini, Taisuke Izumi, Tomoko Izumi, Nicola Santoro, and Giovanni Viglietta. Population protocols with faulty interactions: the impact of a leader. In *CIAC 2017*. Springer, 2017.
19. Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *PODC 2006*, New York, NY, USA, 2006. ACM Press.

20. David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. In *International Symposium on Distributed Computing (DISC)*, pages 602–616. Springer, 2015. Also in *Distributed Computing*, 2016.
21. Amanda Belleville, David Doty, and David Soloveichik. Hardness of Computing and Approximating Predicates and Functions with Leaderless Population Protocols. In *ICALP 2017*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.
22. Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. Time-space trade-offs in population protocols. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2560–2579. SIAM, 2017.
23. Ryu Mizoguchi, Hirotaka Ono, Shuji Kijima, and Masafumi Yamashita. On space complexity of self-stabilizing leader election in mediated population protocol. *Distributed Computing*, 25(6):451–460, 2012.
24. Shantanu Das, Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, and Giovanni Viglietta. Mediated population protocols: Leader election and applications. In *International Conference on Theory and Applications of Models of Computation*, pages 172–186. Springer, 2017.
25. Joffroy Beauquier, Julien Clement, Stephane Messika, Laurent Rosaz, and Brigitte Rozoy. Self-stabilizing counting in mobile sensor networks with a base station. In *Distributed Computing*, pages 63–76. Springer Berlin Heidelberg, 2007.
26. Joffroy Beauquier, Janna Burman, Simon Claviere, and Devan Sohier. Space-optimal counting in population protocols. In *DISC 2015: International Symposium on Distributed Computing*, pages 631–646. Springer, 2015.
27. James Aspnes, Joffroy Beauquier, Janna Burman, and Devan Sohier. Time and Space Optimal Counting in Population Protocols. In *OPODIS 2016*, volume 70, 2017.
28. Othon Michail. 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*, pages 37–46, 2015. Also in *Distributed Computing*, 2017.
29. David Doty, Mahsa Eftekhari, Othon Michail, Paul G. Spirakis, and Michail Theofilatos. Exact size counting in uniform population protocols in nearly logarithmic time. *CoRR*, abs/1805.04832, 2018.
30. Tomoko Izumi, Keigo Kinpara, Taisuke Izumi, and Koichi Wada. Space-efficient self-stabilizing counting population protocols on mobile sensor networks. *Theor. Comput. Sci.*, 552:99–108, 2014.
31. Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM symposium on Theory of computing (STOC)*, pages 513–522. ACM, 2010.
32. Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. Naming and counting in anonymous unknown dynamic networks. In *15th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 281–295. Springer, 2013.
33. Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Counting in anonymous dynamic networks under worst-case adversary. *IEEE 34th International Conference on Distributed Computing Systems (ICDCS)*, 2014.
34. Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.