Brief Announcement: Exact Size Counting in Uniform Population Protocols in Nearly Logarithmic Time

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

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

Othon Michail
Department of Computer Science, University of Liverpool, UK
Othon.Michail@liverpool.ac.uk

Paul G. Spirakis
Department of Computer Science, University of Liverpool, UK and Computer Technology Institute & Press "Diophantus" (CTI), Patras, Greece
P.Spirakis@liverpool.ac.uk

Michail Theofilatos
Department of Computer Science, University of Liverpool, UK
michail.theofilatos@liverpool.ac.uk

Abstract

We study population protocols: networks of anonymous agents whose pairwise interactions are chosen uniformly at random. The *size counting problem* is that of calculating the exact number n of agents in the population, assuming no leader (each agent starts in the same state). We give the first protocol that solves this problem in sublinear time.

The protocol converges in $O(\log n \log \log n)$ time and uses $O(n^{60})$ states ($O(1)+60 \log n$ bits of memory per agent) with probability $1-O(\frac{\log \log n}{n})$. The time to converge is also $O(\log n \log \log n)$ in expectation. Crucially, unlike most published protocols with $\omega(1)$ states, our protocol is *uniform*: it uses the same transition algorithm for any population size, so does not need an estimate of the population size to be embedded into the algorithm.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases population protocol, counting, leader election, polylogarithmic time

Digital Object Identifier 10.4230/LIPIcs.DISC.2018.46

Funding DD, ME: NSF grant CCF-1619343. OM, PS, MT: EEE/CS initiative NeST. MT: Leverhulme Research Centre for Functional Materials Design.

1 Introduction

Population protocols [4] 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.

The *(parallel) time* for some event to happen in a protocol is a random variable, defined as the number of interactions, divided by n, until the event happens. A recent blitz of impressive results in population protocol has shown that leader election [1, 9, 7, 8] and exact majority [3, 2] can be solved in polylog(n) time using polylog(n) states. Most of the protocols with $\omega(1)$ states use a *nonuniform* model: given n, the state set Q_n and transition function $\delta_n : Q_n \times Q_n \to Q_n \times Q_n$ are allowed to depend arbitrarily on n, other than the constraint that $|Q_n| \leq f(n)$ for some function f growing as polylog(n). This nonuniformity is used in most of the cited protocols to encode a value such as $\lfloor \log n \rfloor$ into each agent.

We define a *uniform* variant of the model: the same transition algorithm is used for all populations, though the number of states may vary with the population size. A uniform protocol can be deployed into *any* population without knowing in advance the size, or even a rough estimate of the size. The original, $O(1)$-state model [4, 5, 6], is uniform since there is a single transition function. Because we allow memory to grow with n, our model's power exceeds that of the original, but is strictly less than that of the nonuniform model of most papers using $\omega(1)$ states.

2 Algorithm

The problem of counting the number of agents and storing this number in each agent is clearly solvable by an $O(n)$ time protocol using a straightforward leader election: Agents initially assume they are leaders and the count is 1. When two leaders meet, one agent sums their counts while the other becomes a follower, and followers propagate by epidemic the maximum count. No faster protocol was previously known. Our main result improves this.

▶ **Theorem 2.1.** *There is a leaderless, uniform population protocol solving the exact size counting problem with probability 1. With probability at least* $1 - \frac{10+5\log\log n}{n}$*, the convergence time is at most* $6 \ln n \log\log n$*, and each agent uses* $17 + 60\log n$ *bits of memory. The expected time to convergence is at most* $7 \ln n \log\log n$*.*

Key to our technique is a protocol, due to Mocquard et al. [10] (and similar to that of Alistarh and Gelashvili [3]), that counts the exact difference between the number b of "blue" and r of "red" agents in the initial population. The protocol assumes that each agent initially stores n exactly (so is nonuniform). Blue agents start with an integer value $-\mathtt{M}$, while red agents start with $\mathtt{M}$. When two agents meet, they average their values, one rounding up and the other down if the sum is odd. This eventually converges to all agents sharing the population-wide average $(b-r)\frac{\mathtt{M}}{n}$, and the estimates of this average get close enough for the output to be correct within $O(\log n)$ time [10]. Our protocol essentially inverts this, starting with one blue agent (a leader) and $n-1$ red agents, we compute the population size as a function of the average. (See below for details.) However, for this to work, our protocol requires a leader and for each agent to share a value $\mathtt{M} \geq 3n^3$, which are not present initially. Four sub-protocols are used in total (although all agents run in parallel, each subprotocol runs sequentially within each agent whenever it interacts): UniqueID, ElectLeader, Averaging, and Timer.

UniqueID eventually assigns to every agent a unique ID, represented as a binary string. Agents start with ID ϵ (empty string), and whenever two agents with the same ID meet, all agents double the length of their IDs with uniformly random bits (appending a single bit when two ϵ's meet). This protocol requires $\Omega(n)$ time to converge, but within only $O(\log n \log\log n)$ time can be used by the next subprotocol to elect a leader.

ElectLeader propagates the lexicographically largest ID (considered the ID of the leader) by epidemic (via transition of the form $x, y \to y, y$ if $y > x$ lexicographically). The length of the leader's ID is used as a polynomial-factor upper bound on $3n^3$.

Averaging uses a fast averaging protocol [10, 3]. We assume the initial configuration of this protocol is one leader and $n-1$ followers. (This protocol and the next (Timer) are restarted each time the UniqueID protocol discovers two agents shared an ID; so eventually Averaging will be restarted with a unique leader.) Each agent stores the value `M`, and the leader initializes an integer field `ave` to be `M`, with followers initializing `ave` to be 0. When two agents meet, they average their `ave` fields, with one rounding up and the other rounding down if the sum is odd. Thus the population-wide sum is always `M`. Eventually all agents have $\texttt{ave} = \lceil \frac{\texttt{M}}{n} \rceil$ or $\lfloor \frac{\texttt{M}}{n} \rfloor$, so $n = \lfloor \frac{\texttt{M}}{\texttt{ave}} + \frac{1}{2} \rfloor$ (i.e., $\frac{\texttt{M}}{\texttt{ave}} + \frac{1}{2}$ rounded to the nearest integer). It could take linear time for `ave` to converge this closely to $\frac{\texttt{M}}{n}$, but as long as $\texttt{M} \geq 3n^3$ and `ave` is within n of $\frac{\texttt{M}}{n}$, $\lfloor \frac{\texttt{M}}{\texttt{ave}} + \frac{1}{2} \rfloor$ is the correct population size n; we show that in $O(\log n)$ time all `ave` fields are within n of $\frac{\texttt{M}}{n}$.

Since UniqueID continues restarting beyond the $O(\log n \log \log n)$ time required for initialize convergence to a correct output, Timer is used to detect when Averaging has likely converged, waiting to write output into the `output` field of the agent. Timer is a phase clock [6] that ensures after the correct value is written, on subsequent restarts of Averaging, the incorrect values that exist before Averaging re-converges will not overwrite the correct value recorded into `output` during the earlier restart.

3 Conclusion

$\Omega(n)$ is a clear lower bound on the number of states required for any protocol computing the exact population size, since $\log n$ bits are required merely to write the number n. (Note that our protocol uses $60 \log n$ bits.) It is an open question if there exists a uniform polylog-time, $O(n)$-state population protocol for exact size computation.

References

1 Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. Time-space trade-offs in population protocols. In *SODA*, 2017.

2 Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *SODA*, 2018.

3 Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *ICALP*, 2015.

4 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, 2006.

5 Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *PODC*, 2006.

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

7 Petra Berenbrink, Dominik Kaaser, Peter Kling, and Lena Otterbach. Simple and Efficient Leader Election. In *SOSA*, 2018.

8 Andreas Bilke, Colin Cooper, Robert Elsässer, and Tomasz Radzik. Brief announcement: Population protocols for leader election and exact majority with $\mathrm{O}(\log^2 n)$ states and $\mathrm{O}(\log^2 n)$ convergence time. In *PODC*, 2017.

9 Leszek Gasieniec and Grzegorz Stachowiak. Fast space optimal leader election in population protocols. In *SODA*, 2018.

10 Yves Mocquard, Emmanuelle Anceaume, James Aspnes, Yann Busnel, and Bruno Sericola. Counting with population protocols. In *NCA*, 2015.