Brief Announcement: Loosely-stabilizing Leader Election with Polylogarithmic Convergence Time

Yuichi Sudo
Graduate School of Information Science and Technology, Osaka University, Japan
y-sudou@ist.osaka-u.ac.jp

Fukuhito Ooshita
Graduate School of Science and Technology, Nara Institute of Science and Technology, Japan
f-oosita@is.naist.jp

Hirotsugu Kakugawa
Graduate School of Information Science and Technology, Osaka University, Japan
kakugawa@ist.osaka-u.ac.jp

Toshimitsu Masuzawa
Graduate School of Information Science and Technology, Osaka University, Japan
masuzawa@ist.osaka-u.ac.jp

Abstract

We present a fast loosely-stabilizing leader election protocol in the population protocol model. It elects a unique leader in a poly-logarithmic time and holds the leader for a polynomial time with arbitrarily large degree in terms of parallel time, i.e, the number of steps per the population size.

2012 ACM Subject Classification Theory of computation → Self-organization

Keywords and phrases Self-stabilization, Loose-stabilization, Population protocols

Digital Object Identifier 10.4230/LIPIcs.DISC.2018.52

Acknowledgements This work was supported by JSPS KAKENHI Grant Numbers 17K19977, 16K00018, 18K11167, and 18K18000 and Japan Science and Technology Agency(JST) SICORP.

1 Introduction

We consider the *population protocol* (PP) model [1] in this paper. A network called *population* consists of a large number of finite-state automata, called *agents*. Agents often make *interactions*, each between a pair of agents to communicate with, by which agents update their states. As with the majority of studies on population protocols, we consider only the population of complete graphs and the uniformly-random scheduler, which selects an interacting pair of agents at each step uniformly at random.

We focus on Self-Stabilizing Leader Election (SS-LE) problem, which is one of the most important and well-studied problems in the PP model. This problem requires that starting from any configuration, a population reaches a safe configuration in which exactly one leader exists; and after that, the population keeps that leader forever. Unfortunately, it is well known that no protocol solves SS-LE unless every agent knows the exact size n of the population (i.e., the number of agents). To circumvent this impossibility, our previous work [3] introduced the concept of *loose-stabilization*, a relaxed variant of self-stabilization: Starting from any initial configuration, the population must reach a safe configuration within a short time; after that, the specification of the problem must be sustained for a sufficiently long time, though not necessarily forever. This previous work gave a loosely-stabilizing leader

Table 1 LS-LE protocols in the PP model. Time complexities are presented in parallel time.

	Convergence Time	Holding Time	Agent Memory
Sudo et al. [3]	$O(N \log N)$	$\Omega(e^N)$	$O(\log N)$
Izumi [2]	$O(N)$	$\Omega(e^N)$	$O(\log N)$
Proposed Protocol (P_{PL})	$O(c \log^3 N)$	$\Omega(cn^{10c})$	$O(\log \log N)$

Algorithm 1 P_{PL}: specifying a state-transition at interaction between agents a_0 and a_1.

1: $a_0.\mathtt{timer}_P \leftarrow a_1.\mathtt{timer}_P \leftarrow \max(a_0.\mathtt{timer}_P - 1,\ a_1.\mathtt{timer}_P - 1, 0)$
2: **for** $i \in \{0,1\}$ such that $a_i.\mathtt{timer}_P = 0$ **do** $a_i.\mathtt{leader} \leftarrow \top$ **endfor**
3: **if** $\exists i \in \{0,1\} : a_i.\mathtt{leader} = \top$ **then** $a_0.\mathtt{timer}_P \leftarrow a_1.\mathtt{timer}_P \leftarrow t_{\max}$ **endif**
4: $a_0.\mathtt{virus} \leftarrow a_1.\mathtt{virus} \leftarrow \max(a_0.\mathtt{virus} - 1,\ a_1.\mathtt{virus} - 1, 0)$
5: **for** $i \in \{0,1\}$ such that $\neg a_i.\mathtt{shield} \wedge (a_i.\mathtt{virus} > 0)$ **do** $a_i.\mathtt{leader} \leftarrow \bot$ **endfor**
6: **for** $i \in \{0,1\}$ **do** $a_i.\mathtt{timer}_L \leftarrow \max(a_i.\mathtt{timer}_L - 1, 0)$ **endfor**
7: **if** $a_0.\mathtt{timer}_L = 0 \wedge a_0.\mathtt{leader} = \top$ **then**
8: $\quad a_0.\mathtt{virus} \leftarrow t_{\mathrm{virus}}$
9: $\quad a_0.\mathtt{shield} \leftarrow \top$
10: **end if**
11: **if** $a_1.\mathtt{timer}_L = 0 \wedge a_1.\mathtt{leader} = \top$ **then** $a_1.\mathtt{shield} \leftarrow \bot$ **endif**
12: **for** $i \in \{0,1\}$ such that $a_i.\mathtt{timer}_L = 0$ **do** $a_i.\mathtt{timer}_L \leftarrow t_{\mathrm{emit}}$ **endfor**

election (LS-LE) protocol assuming that every agent knows only a common upper bound N of n. This protocol is practically equivalent to an SS-LE protocol since it maintains the unique leader for exponential time in n after reaching a safe configuration within $O(N \log N)$ parallel time, i.e., the number of steps (interactions) per the population size n. Recently, Izumi [2] presented a method to improve the convergence time of this protocol to $O(N)$ parallel time. He also proved the optimality of its convergence time by showing that any LS-LE protocol whose holding time is exponential requires $\Omega(N)$ parallel time for convergence.

In this paper, we break through the barrier of this linear lower bound of convergence time and achieve poly-logarithmic parallel convergence time. Given a parameter $c \geq 1$ and an upper bound N of n, our protocol converges to a safe configuration in $O(c \log^3 N)$ time, and preserves the unique leader for $\Omega(cn^{10c})$ time thereafter (Table 1). Owing to the above impossibility result by [2], the holding time of our protocol is no longer exponential but polynomial in n. However, we can arbitrarily increase the degree of the polynomial using parameter c. For example, the holding time is $\Omega(n^{100})$ if we assign $c = 10$, which is expected to be large enough in all practical situations.

2 Proposed Protocol

The pseudo code of P_{PL} is shown in Algorithm 1. Each agent has five variables $\mathtt{leader} \in \{\top, \bot\}$, $\mathtt{shield} \in \{\top, \bot\}$, $\mathtt{virus} \in [0, t_{\mathrm{virus}}]$, $\mathtt{timer}_P \in [0, t_{\max}]$, and $\mathtt{timer}_L \in [0, t_{\mathrm{emit}}]$. The first two variables $\mathtt{leader}$ and $\mathtt{shield}$ are Boolean variables: $v.\mathtt{leader} = \top$ means that agent v is a leader, and $v.\mathtt{shield}$ will be explained later. The variables $\mathtt{virus}$, $\mathtt{timer}_P$, and $\mathtt{timer}_L$ are count-down timers where their maximum values are $t_{\mathrm{virus}} = 60\lceil \log N \rceil$, $t_{\max} = 12c \cdot t_{\mathrm{virus}} \lceil \log N \rceil$, and $t_{\mathrm{emit}} = 12c \cdot t_{\mathrm{virus}} \lceil \log N \rceil$, respectively ($t_{\max} = t_{\mathrm{emit}}$).

Protocol P_{PL} consists of a *timeout mechanism* (Lines 1-3) and a *virus-war mechanism* (Lines 4-12). The timeout mechanism creates a leader when no leader exists in the population while the virus-war mechanism reduces the number of leaders when two or more leaders exist.

The timeout mechanism of P_{PL} (Lines 1-3) is almost the same as that of the protocol given in [3]. This mechanism uses a propagating timer $\texttt{timer}_P$, which indicates the possibility of existence of a leader. A leader agent always keeps the maximum value of the timer, i.e., $\texttt{timer}_P = t_{\max}$, and resets the timer of the other agent to $t_{\max}$ every time it interacts with a non-leader agent (Line 3). When two non-leaders interact, the higher value of the two timers is propagated, but is decremented by one (Line 1). When the timer of a non-leader decreases to zero, it suspects that no leader exists in the population, and it becomes a new leader (Line 2). The loosely-stabilizing property of this mechanism holds because (i) when no leader exists, some agent detects the timeout of its timer within a short time ($O(t_{\max} \log n)$ parallel time) and it becomes a leader, and (ii) when at least one leader exists, timeout rarely happens thanks to the timer reset by the leader(s) and the larger-value propagation.

We uses the virus war mechanism presented in [4], but implements it in a considerably different way to achieve a poly-logarithmic convergence time. Every leader tries to kill other leaders by using viruses and become the unique leader. We say that agent v has a virus if $v.\texttt{virus} > 0$, and that v is wearing a shield if $v.\texttt{shield} = \top$. Every agent has a local timer $\texttt{timer}_L$ to create a new virus periodically. This timer is decreased by one every time the agent interacts (Line 6). When the local timer of a leader reaches zero at an interaction, the leader meets a different fate according to its role, initiator or responder, in the interaction. If the leader is an initiator, it succeeds in creating a new virus and wears a shield, that is, it substitutes $\texttt{virus} \leftarrow t_{\mathrm{virus}}$ and $\texttt{shield} \leftarrow \top$ (Lines 8-9). If it is a responder, it fails to create a new virus and its shield gets broken if it wears (Line 11). Note that the uniformly-random scheduler gives each side of the coin-toss (initiator or responder) the same probability, i.e., 1/2. Thereafter, the local timer is reset to the maximum value t_{emit} (Line 12). A virus spreads by interactions (Line 4). A leader is kelled and becomes a non-leader if it catches a virus when it does not wear a shield (Line 5). The value of `virus` corresponds to the TTL (time to live) of a virus and decreases in the large-value propagation fashion. The loosely-stabilizing property of this mechanism holds because (i) as long as multiple leaders exist, the number of leaders decreases by half in every $O(t_{\mathrm{emit}})$ parallel time thanks to the fair coin-toss of the uniformly random scheduler and (ii) viruses rarely remove all leaders from the population thanks to $t_{\mathrm{virus}} \ll t_{\mathrm{emit}}$. ($t_{\mathrm{virus}} \ll t_{\mathrm{emit}}$ guarantees that at least one leader wears a shield with high probability when viruses exist in the population.)

The above intuitive explanation holds if t_{virus} is sufficiently large. However, we assign logarithmic value to t_{virus} to get poly-logarithmic convergence time. A critical question arises here: Are created viruses propagated to the whole population with high probability? A similar question arises for the propagating timers. Careful and non-trivial analysis, omitted from this paper, affirms these questions and proves the performance of P_{PL} in Table 1.

References

1 D. Angluin, J Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.

2 T. Izumi. On space and time complexity of loosely-stabilizing leader election. In *SIROCCO*, pages 299–312, 2015.

3 Y. Sudo, J. Nakamura, Y. Yamauchi, F. Ooshita, et al. Loosely-stabilizing leader election in a population protocol model. *Theoretical Computer Science*, 444:100–112, 2012.

4 Y. Sudo, F. Ooshita, H. Kakugawa, and T. Masuzawa. Loosely-stabilizing leader election on arbitrary graphs in population protocols. In *OPODIS*, pages 339–354, 2014.