Time-Optimal Loosely-Stabilizing Leader Election in Population Protocols

Yuichi Sudo
Hosei University, Tokyo, Japan

Ryota Eguchi
Nagoya Institute of Technology, Japan

Taisuke Izumi
Osaka University, Japan

Toshimitsu Masuzawa
Osaka University, Japan

Abstract

We consider the leader election problem in the population protocol model. In pragmatic settings of population protocols, self-stabilization is a highly desired feature owing to its fault resilience and the benefit of initialization freedom. However, the design of self-stabilizing leader election is possible only under a strong assumption (i.e., the knowledge of the *exact* size of a network) and rich computational resource (i.e., the number of states). Loose-stabilization is a promising relaxed concept of self-stabilization to address the aforementioned issue. Loose-stabilization guarantees that starting from any configuration, the network will reach a safe configuration where a single leader exists within a short time, and thereafter it will maintain the single leader for a long time, but not necessarily forever. The main contribution of this paper is giving a time-optimal loosely-stabilizing leader election protocol. The proposed protocol with design parameter $\tau \geq 1$ attains $O(\tau \log n)$ parallel convergence time and $\Omega(n^\tau)$ parallel holding time (i.e., the length of the period keeping the unique leader), both in expectation. This protocol is time-optimal in the sense of both the convergence and holding times in expectation because any loosely-stabilizing leader election protocol with the same length of the holding time is known to require $\Omega(\tau \log n)$ parallel time.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases population protocols, leader election, loose-stabilization, self-stabilization

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.40

Related Version *Previous Version*: `https://arxiv.org/abs/2005.09944`

Funding This work was supported by JSPS KAKENHI Grant Numbers 19H04085 and 20H04140.

Acknowledgements We truly thank anonymous reviewers for their constructive and helpful comments. We also thank Przemyslaw Uznanski and Eric Severson: The first author of this paper got one of the key ideas of this paper after the discussion with them at PODC 2019.

1 Introduction

We consider the *population protocol* (PP) model [5] in this paper. A network called the *population* consists of n automata called *agents*. Pairs of agents execute *interactions* (i.e., pairwise communication) by which they update their states. These interactions are opportunistic, that is, they are unknown and unpredictable (or only predictable with probability). Agents are strongly anonymous: they do not have identifiers and cannot distinguish neighbors with the same state. As with the majority of studies on population protocols [5, 6, 4, 2, 14, 15, 22, 19], we assume that exactly one pair of agents is selected to have an interaction uniformly at random from all $\binom{n}{2}$ pairs at each step. In the PP model, time complexity such as expected

Table 1 Self/Loosely-stabilizing leader election in the PP model (The convergence/holding time is shown in expected parallel time).

	Type	Knowledge N	Convergence time	Holding time	#states	Design parameter
[10]	SS-LE	$N = n$	$O(n^2)$	∞	n	-
[9]	SS-LE	$N = n$	$O(n)$	∞	$O(n)$	-
[9]	SS-LE	$N = n$	$O(\log n)$	∞	$n^{O(n^{\log n})}$	-
[9]	SS-LE	$N = n$	$O(n^{\frac{1}{H+1}})$	∞	$n^{O(n^H)}$	$H = O(1)$
[18]	LS-LE	$n \leq N = O(n)$	$O(n)$	$\Omega(e^n)$	$O(n)$	-
[16]	LS-LE	$n \leq N = O(n)$	$O(n)$	$\Omega(e^n)$	$O(n)$	-
[22]	LS-LE	$n \leq N = \mathrm{poly}(n)$	$O(\tau \log^3 n)$	$\Omega(n^\tau)$	$O(\tau^2 \log^5 n)$	$\tau \geq 1$
ours	LS-LE	$n \leq N = \mathrm{poly}(n)$	$O(\tau \log n)$	$\Omega(n^\tau)$	$O(\tau \log n)$	$\tau \geq 1$

convergence time is usually evaluated in *parallel time*, that is, the number of steps divided by n (i.e., the number of agents). This is a natural measure of time because in practice, interactions typically occur in parallel in the population. For the remainder of this section, we presume parallel time when we discuss time complexity.

In this paper, we focus on the problem of self-stabilizing leader election (SS-LE). This problem requires that (i) starting from any configuration, a population reaches a safe configuration in which exactly one leader exists; and thereafter, (ii) it keeps this leader forever. These requirements guarantee tolerance against finitely many transient faults. Since many protocols (self-stabilizing or non-self-stabilizing) in the literature assume a unique leader [5, 7, 6], SS-LE is key to improving fault-tolerance of the PP model itself. However, it is known that no protocol can solve SS-LE unless every agent in the population knows the *exact size* n of the population [7, 10] [1]. Under this strong assumption (i.e., all agents know exact n), several SS-LE protocols have been presented in the literature. Cai et al. [10] gave the first SS-LE protocol under this assumption, which elects the unique leader within $O(n^2)$ time starting from any configuration. Recently, Burman et al. [9] gave three SS-LE protocols, which improve the convergence time at the cost of space complexity, that is, the number of states per agent (Table 1). For example, one of their protocols converges in $O(\log n)$ time but uses a super-exponential number of states.

We can discard the assumption of exact knowledge of n by slightly relaxing the requirement of self-stabilization, that is, by taking an approach called *loose-stabilization*. Loose-stabilization guarantees that the population reaches a safe configuration within a relatively short time starting from any initial configuration; after that, the specification of the problem (such as having a unique leader in the leader election) must be sustained for a sufficiently long time, though not necessarily forever. Sudo et al. [18] gave a loosely-stabilizing leader election (LS-LE) protocol by assuming that every agent knows a common *upper bound* N of n. Their protocol is not self-stabilizing; however, it is practically equivalent to an SS-LE protocol because it maintains the unique leader for an exponentially long time after reaching a safe configuration. Further, it converges in a safe configuration within $O(N)$ time starting from any configuration.[2] Hence, the convergence time is $O(n)$ if we have a good upper bound

[1] Strictly speaking, they prove a slightly weaker impossibility. However, we can prove this impossibility based on almost the same technique: a simple partitioning argument. See [22] for details (page 618, footnote).

[2] The convergence time of this protocol was proven to be $O(N \log n)$ in [18]. Later, it was found to be $O(N)$ according to Lemma 1 in [3].

$N = O(n)$. In practice, the knowledge of N is a much weaker assumption than the exact knowledge of n; the protocol works correctly even if we consider a large overestimation of n, such as $N = 100n$. Recently, Sudo et al. [22] gave an LS-LE protocol with poly-logarithmic convergence time, which has a design parameter τ (≥ 1) controlling the convergence and holding times. Given an upper bound $N \geq n$ such that $N = O(n^c)$ for some constant c, their protocol reaches a safe configuration within $O(\tau \log^3 n)$ time, and thereafter, it keeps the single leader for $\Omega(n^\tau)$ time, both in expectation.

Izumi [16] provided a lower bound on the convergence time of an LS-LE protocol, given that it keeps the unique leader for an exponentially long time after reaching a safe configuration. Sudo et al. [22] generalized this lower bound as follows: if the expected holding time of an LS-LE protocol is β/n, its expected convergence time must be $\Omega(\log \beta)$. Therefore, we have a gap of $\log^2 n$ factor between this lower bound and the upper bound given by Sudo et al. [22] when we require an expected holding time of $\Omega(n^\tau)$: the former is $\Omega(\tau \log n)$ and the latter is $O(\tau \log^3 n)$.

1.1 Our Contribution

We close the above-mentioned gap in this paper. That is, we develop an LS-LE protocol whose expected convergence time is $O(\tau \log n)$ and expected holding time is $\Omega(n^\tau)$, where $\tau \geq 1$ is the design parameter of the protocol. Interestingly, this convergence time is optimal for any length of holding time. For $\tau \geq 1$, we have no asymptotic gap between this convergence time and the lower bound given by Sudo et al. [22]. Even if a holding time of $o(n)$ is sufficient, the expected convergence time of our protocol with $\tau = 1$ remains optimal. This is because every LS-LE protocol requires $\Omega(\log n)$ time to reach a safe configuration regardless of the length of its holding time. Consider an execution of any LS-LE protocol starting from a configuration where all agents are leaders. Then, $n - 1$ agents must have at least one interaction before electing the unique leader. However, a simple analysis on the famous *coupon collector's problem* yields that this requires $\Omega(\log n)$ time (i.e., $\Omega(n \log n)$ steps) in expectation. In addition to time-optimality, the proposed protocol has a small space complexity: The number of states per agent is $O(\tau \log n)$, which is much smaller than $O(\tau^2 \log^5 n)$ in [22].

The proposed protocol also shows how useful loose-stabilization is in the PP model. When we set $\tau = 100$, its expected convergence time is $O(\log n)$, and the expected holding time is $\Omega(n^{100})$, practically forever. This protocol needs only the knowledge of N such that $n \leq N = O(n^c)$ holds for some constant c. Under self-stabilization, if we require the same convergence time, the only known solution [9] uses an super-exponential number of states and requires a much stronger assumption, i.e., the knowledge of *exact* n.

1.2 Further Related Work

Leader election has been extensively studied in the PP model. When we design non-self-stabilizing protocols, we can assume that all agents are in a specific state at the initial configuration. Leader election is then achieved by employing a simple protocol [5]. In this protocol, all agents are initially leaders, and we have only one transition rule: when two leaders meet, one of them becomes a follower (i.e., a non-leader). This simple protocol elects a unique leader in linear time and uses only two states at each agent. This protocol is time-optimal: Doty and Soloveichik [13] showed that any constant space protocol requires linear time to elect a unique leader. In a breakthrough result, Alistarh and Gelashvili [4] designed a (non-self-stabilizing) leader election protocol that converges in $O(\log^3 n)$ parallel time and uses $O(\log^3 n)$ states at each agent. Thereafter, a number of papers have been

devoted to fast leader election, to name a few, [2, 14, 15, 19, 8]. Gąsieniec, Staehowiak, and Uznanski [15] gave an algorithm that converges in $O(\log n \log\log n)$ time and uses a surprisingly small number of states: only $O(\log\log n)$ states per agent. This is space-optimal because it is known that every leader election protocol with $O(n/\mathrm{polylog}(n))$ time uses $\Omega(\log\log n)$ states [1]. Sudo et al. [19] gave a simple protocol that elects a unique leader within $O(\log n)$ time and uses $O(\log n)$ states per agent. This is time-optimal because any leader election protocol requires $\Omega(\log n)$ time even if it uses an arbitrarily large number of states and the agents know the exact size of the population [17]. [3] At last, Berenbrink et al. [8] gave a time and space optimal protocol, i.e., an $O(\log n)$-time and $O(\log\log n)$-states leader election protocol.

SS-LE and LS-LE protocols have been presented also for a population where some pairs of agents may not have interactions, i.e., the interaction graph is not complete [11, 12, 20, 21, 23, 24].

2 Preliminaries

2.1 Model

We denote the set of integers $\{z \in \mathbb{N} \mid x \leq z \leq y\}$ by $[x, y]$. The omitted bases of logarithms are 2.

A *population* is the set V of n *agents* (i.e., $|V| = \mathrm{n}$) that change their states by pairwise interactions. Every pair of agents $(u, v) \in E = V \times V \setminus \{(w, w) \mid w \in V\}$ can interact with each other. A *protocol* P on the population is defined by a 4-tuple $P = (Q, Y, T, \pi_{out})$ consisting of a finite set Q of states, a finite set Y of output symbols, a transition function $T : Q \times Q \to Q \times Q$, and an output function $\pi_{out} : Q \to Y$. When two agents interact, T determines their next states based on their current states. The output function π_{out} maps the current local state $q \in Q$ to a value in the output domain $\pi_{out}(q) \in Y$. The state of each agent including the current output are often described as a set of local variables. Throughout this paper, we use the notation $v.x$ to denote the value of a variable x managed by agent v.

We assume that all agents have a common knowledge N on n such that $n \leq N = O(n^c)$ holds for some constant c, which is equivalent to the assumption that the agents have a constant-factor approximation m of $\log n$, i.e., $\alpha \log n \geq m \geq \log n$ for some constant $\alpha \geq 1$[4].

A *configuration* is a mapping $C : V \to Q$ that specifies the states of all agents. Given a protocol P on n agents, the set of all possible configurations for P is denoted by $\mathcal{C}_{\mathrm{all}}(P)$. We say that a configuration C changes to C' by an interaction $e = (u, v)$, denoted by $C \overset{P,e}{\to} C'$, if $(C'(u), C'(v)) = T(C(u), C(v))$ and $C'(w) = C(w)$ for all $w \in V \setminus \{u, v\}$. Then u and v are respectively called the *initiator* and the *responder* of e. Given an interaction e, we say that agent $v \in V$ *participates* in e if v is either the initiator or the responder of e.

We assume the *uniformly random scheduler* $\boldsymbol{\Gamma}$, which selects two agents to interact at each step uniformly at random from all pairs of agents. Specifically, $\boldsymbol{\Gamma} = \Gamma_0, \Gamma_1, \ldots$ where each $\Gamma_t \in E$ is a random variable such that $\Pr(\Gamma_t = (u, v)) = \frac{1}{n(n-1)}$ for any $t \geq 0$ and any distinct $u, v \in V$. Given an initial configuration $C_0 \in \mathcal{C}_{\mathrm{all}}(P)$, the *execution* of protocol P under the uniformly random scheduler $\boldsymbol{\Gamma}$ is defined as $\Xi_P(C_0, \boldsymbol{\Gamma}) = C_0, C_1, \ldots$ where $C_t \overset{P,\Gamma_t}{\to} C_{t+1}$ holds for all $t \geq 0$. Note that each C_i is also a random variable.

[3] This lower bound may look obvious, but it is not: it does not immediately follow from a simple coupon collector argument because unlike SS-LE/LS-LE setting, we can now specify an initial configuration such that all agents are followers.

[4] In this sense, any protocol P should be parametric (with respect to m) such as $P_m = (Q_m, Y_m, T_m, \pi_{\mathrm{out},m})$ strictly. In this paper, we do not explicitly state parameter m of P for simplicity.

2.2 Loosely-Stabilizing Leader Election

In leader election protocols, every agent is equipped with an output variable $\texttt{leader} \in \{0, 1\}$, which indicates whether the agent is a leader. That is, if $v.\texttt{leader} = 1$ holds, v is a leader, and a follower otherwise. A configuration C is called *correct with leader* $v \in V$ if v outputs 1 and all other agents output 0. Given any configuration C, we define $\mathrm{EIH}_P(C)$ as the expected length of the longest prefix of $\Xi_P(C, \mathbf{\Gamma})$, where any configuration is correct with a common leader $v \in V$. Note that $\mathrm{EIH}_P(C) = 0$ holds if a configuration C is not correct. For any configuration C and any subset $\mathcal{S} \subseteq \mathcal{C}_{\mathrm{all}}(P)$ of configurations, we also define $\mathrm{EIC}_P(C, \mathcal{S})$ as the expected length of the longest prefix of $\Xi_P(C, \mathbf{\Gamma})$, where any configuration is not in $\mathcal{S}$. The notation EIH (resp. EIC) stands for the Expected number of Interactions to Hold (resp. Converge).

▶ **Definition 1** (Loosely-stabilizing leader election [18]). *Let α and β be positive real numbers. Protocol $P(Q, Y, T, \pi_{out})$ is an (α, β)-loosely-stabilizing leader election protocol if there exists a set $\mathcal{S}$ of configurations satisfying the two inequalities*

$$\max_{C \in \mathcal{C}_{\mathrm{all}}(P)} \mathrm{EIC}_P(C, \mathcal{S}) \leq \alpha \quad \textit{and} \quad \min_{C \in \mathcal{S}} \mathrm{EIH}_P(C) \geq \beta.$$

We call $\mathcal{S}$ defined by the definition above the set of *safe* configurations of P. Note that the condition $\beta > 0$ guarantees the correctness (i.e., uniqueness of leader) of configurations in $\mathcal{S}$. In terms of parallel time, an (α, β)-loosely-stabilizing leader election protocol P reaches a safe configuration within α/n parallel time in expectation, and it keeps the elected leader during the following β/n parallel time in expectation. We call α/n and β/n *the expected convergence time* and *the expected holding time* of P, respectively.

3 Toolbox

3.1 Epidemic

The protocol *epidemic* [6], denoted by P_{EP}, is often used to propagate the maximum value of a variable to the whole population, which is defined as: (i) each agent has only one variable x, and (ii) when two agents u and v interact, they substitute $\max(u.x, v.x)$ for their variables (i.e., $u.x$ and $v.x$). Then, we have the following lemma.

▶ **Lemma 2** ([6]).[5] *Let k be any non-negative integer, $D_0 \in \mathcal{C}_{\mathrm{all}}(P_{\mathrm{EP}})$ be any configuration of P_{EP}, and $l = \max_{v \in V} v.x$ in configuration D_0. The execution $\Xi_{P_{\mathrm{EP}}}(D_0, \mathbf{\Gamma})$ reaches the configuration such that $u.x = l$ holds for any $u \in V$ within $O(kn \log n)$ steps with probability $1 - O(n^{-k})$.*

3.2 Countdown with Higher Value Propagation

The protocol of *counting down with higher value propagation* (CHVP) [18] is a useful technique to design loosely-stabilizing protocols, particularly for detecting the absence of a leader. It is defined as the following protocol P_{CD}: each agent has only one variable y, and when two agents u and v interact, they substitute $\max(u.y - 1, v.y - 1, 0)$ for their y. We have the following two lemmas.

[5] While the original protocol by Angluin et al. [6] is an one-way version of P_{EP} (i.e., higher value is propagated only from an initiator to a responder), there is no difference on asymptotic propagation time between them (Lemma 8 in [18]).

▶ **Lemma 3** (Lemma 1 in [3]). [6] *Let l_1 and l_2 be any two integers such that $l_1 > l_2 \geq 0$, k be any non-negative integer, and $D_0 \in \mathcal{C}_{\mathrm{all}}(P_{\mathrm{CD}})$ be any configuration of P_{CD} such that $l_1 = \max_{v \in V} v.y$ holds. The execution $\Xi_P(D_0, \mathbf{\Gamma})$ reaches a configuration satisfying $\max_{v \in V} v.y \leq l_2$ within $O(n(l_1 - l_2 + k \log n))$ steps with probability $1 - O(n^{-k})$.*

▶ **Lemma 4** (Lemma 5 in [22]). [7] *Let $D_0 \in \mathcal{C}_{\mathrm{all}}(P_{\mathrm{CD}})$ be any configuration and l be the integer that satisfies $l = \max_{v \in V} v.y$ at D_0. There exists a constant c such that $\Xi_{P_{\mathrm{CD}}}(D_0, \mathbf{\Gamma})$ reaches a configuration satisfying $\min_{v \in V} v.y \geq l - ck \log n$ within $O(kn \log n)$ steps with probability $1 - O(n^{-k})$.*

3.3 Lottery Game and Quick Elimination

The *lottery game*, originally introduced by Alistarh et al. [1] as a part of their leader election protocol, is a probabilistic process of filtering leaders. An abstract form of the lottery game is stated as follows: Let V' be the set of leaders. Every leader $v \in V'$ makes independent fair coin flips until it observes tail for the first time. Then, the number of observed heads s_v (called the *level* of v) is propagated to other leaders. The agent identifying another agent with a higher level drops out as a loser.

There are a few implementations of the lottery game in population protocol models. Alistarh et al. [1] and Sudo et al. [19] develop (non-loosely-stabilizing) leader election protocols, based on their own implementations and analyses for this game. In this paper, we adopt the implementation shown in [19], called *quick elimination* (QE). The pseudocode of QE is given in Algorithm 1, which describes the state transition when two agents a_0 and a_1 interact, where a_0 is an initiator and a_1 is a responder. Since the propagation of level values is easily implemented by the epidemic, the main non-trivial point is how to synthesize coin flips using the randomness of the scheduler. The implementation QE simply utilizes the asymmetry of interactions. That is, if v joins an interaction as the initiator, it receives head as the result of its coin flip, and receives tail if it joins as the responder. Each agent maintains two variables, `done` and `level`, in addition to an output variable `leader`. The flag `done` $\in \{0, 1\}$ implies whether the agent is still in the decision of its level (i.e., it continues (synthetic) coin flips during `done` $= 0$). Starting from the state with `done` $= 0$ and `level` $= 0$, the agent v with v.`leader` $= 1$ first decides its level: it increments v.`level` every time it observes head, and it stops incrementation and sets `done` to 1 when it observes tail for the first time. Agents that have decided their levels perform the epidemic to share the maximum level (lines 6-7). If an agent sees a higher level, it becomes a follower (line 7).

While the lottery game was used as a scheme to eliminate leaders in the past literature, we rather see it as a Monte Carlo protocol for leader election, i.e., we focus on the probability that exactly one player wins (or survives as a leader). The following lemma is the key ingredient of our protocol, which is simple but a new observation that has not been addressed so far.[8]

---

[6] Precisely, k is assumed to be a constant in the original lemma, but the same proof applies in the case that k depends on n.

[7] We obtain this lemma by substituting $d = k + 3$, $d' = k + 3$, $d'' = 6$, and $t = \lceil kn \ln n \rceil$ for the first inequality in Lemma 5 in [22].

[8] This observation might not be new if the results of coin flips by the agents were independent of each other. However, they are not independent in our model because when two agents have an interaction, one of them observes head and the other observes tail. Thus, to the best of our knowledge, this observation is new.

Algorithm 1 ***QE***() (a_0 is an initiator and a_1 is a responder).

for $i \in \{0,1\}$ s.t. $a_i.\texttt{done} = 0 \wedge a_i.\texttt{leader} = 1$ **do**
  if $i = 0$ **then**
    $a_0.\texttt{level} \leftarrow \min(a_0.\texttt{level} + 1, 2m)$
  else
    $a_1.\texttt{done} \leftarrow 1$
if $\begin{pmatrix} a_0.\texttt{done} = 1 \wedge a_1.\texttt{done} = 1 \\ \wedge \exists i \in \{0,1\} : a_i.\texttt{level} < a_{1-i}.\texttt{level} \end{pmatrix}$ **then**
  $a_i.\texttt{leader} \leftarrow 0$; $a_i.\texttt{level} \leftarrow a_{1-i}.\texttt{level}$

▶ **Lemma 5.** *Consider the execution of QE under the uniformly random scheduler* $\mathbf{\Gamma}$ *starting from a configuration where at least one leader exists and* $\texttt{done} = 0$ *and* $\texttt{level} = 0$ *hold for all agents. When all leaders finish deciding their levels (i.e.,* $\texttt{done} = 1$ *holds for all leaders), exactly one leader has the maximum level (*$\max_{v \in V} v.\texttt{level}$*) with probability at least* $1/16$.

Proof. Let V' be the set of leaders at the initial configuration. Let X_v be the level computed by agent $v \in V'$. That is, X_v is the integer such that $v \in V'$ joins X_v interactions as an initiator before it joins an interaction as a responder for the first time. We show that $p_u = \Pr(X_u \geq \lceil \log n \rceil + 2 \wedge \bigwedge_{v \in V' \setminus \{u\}} X_v \leq \lceil \log n \rceil + 1) \geq 1/16n$ holds for any $u \in V'$. Then, the probability that some agent becomes the unique winner is obviously lower bounded by $\sum_{u \in V'} p_u \geq 1/16$. Thus, the lemma holds because when $\texttt{done} = 1$ holds for all leaders, every agent except for the unique winner has a smaller level or must have become a follower before. Since $\Pr(X_u \geq \lceil \log n \rceil + 2) > 1/8n$ holds, it suffices to show $q = \Pr(\bigwedge_{v \in V' \setminus \{u\}} X_v \leq \lceil \log n \rceil + 1 \mid X_u \geq \lceil \log n \rceil + 2) \geq 1/2$. By the union bound, we have $q \geq 1 - \sum_{v \in V' \setminus \{u\}} \Pr(X_v \geq \lceil \log n \rceil + 2 \mid X_u \geq \lceil \log n \rceil + 2)$. When two agents u and v both with $\texttt{done} = 0$ interact with each other, one of them necessarily reaches the decision of its level. That is, u and v have at most one common interaction before either one decides its level. This implies that to obtain $X_v \geq \lceil \log n \rceil + 2$ under the condition $X_u \geq \lceil \log n \rceil + 2$, v must observe at least $\lceil \log n \rceil + 1$ heads at the coin flips independently of the first $\lceil \log n \rceil + 2$ coin flips by u. That is, we have $\Pr(X_v \geq \lceil \log n \rceil + 2 \mid X_u \geq \lceil \log n \rceil + 2) \leq (1/2)^{\lceil \log n \rceil + 1} \leq 1/2n$, and thus, $q \geq 1 - n \cdot (1/2n) \geq 1/2$. ◀

4 Time-optimal LS-LE

In this section, we give an LS-LE protocol $P_{\mathrm{TO}}(\tau)$, where the integer $\tau \geq 1$ is a design parameter controlling the performance of the protocol. Starting from any initial configuration, this protocol reaches a safe configuration within $O(\tau n \log n)$ steps and keeps the single leader in the following $\Omega(n^\tau)$ steps. The number of states per agent is $\Theta(\tau m) = \Theta(\tau \log n)$. In the rest of this paper, we use terminologies "*with high probability*" to mean "with probability $1 - O(1/n)$" and "*with very high probability*" to mean "with probability $1 - O(1/n^\tau)$". Further, the terminology "quickly" is used for implying "within $O(\tau n \log n)$ steps".

4.1 Protocol in a Nutshell

The protocol $P_{\mathrm{TO}}(\tau)$ elects a unique leader by iteratively performing the following two phases, both taking $\Theta(\tau n \log n)$ steps with very high probability.

- **Check phase**: The protocol checks whether the population has at least one leader. Each leader agent propagates a heartbeat message to all others using the epidemics. The agents not receiving that message until the end of the phase conclude that the population has no leader, and they become leaders. Since two or more agents may become leaders, they are filtered in the election phase.
- **Election phase**: Each agent performs QE. As shown in Lemma 5, this phase decreases the number of leader agents to one with a constant probability.

There are two major issues for implementing these phases: how to realize a loosely-stabilizing synchronization mechanism to yield the transition between two phases, and how to combine it with the task of each phase using only a small number of states. The protocol CHVP, stated in Section 3.2, is one of the possible solutions for the first issue, which provides a loosely-stabilizing (synchronized) timeout mechanism; thus, it can be utilized for global phase synchronization. However, addressing the second issue is more challenging. Since the check phase only consumes a constant number of states, it is easily combined with CHVP. In the election phase, both QE and CHVP internally keep a variable of a non-constant size. The former manages a variable `level` $\in [0, 2m]$, and the latter manages a variable whose range is $[0, O(\tau m)]$, as we will see in Section 4.2. Thus, to bound the number of states by $O(\tau m) = O(\tau \log n)$ in total, they must share a single non-constant variable.

We resolve this matter by designing a new loosely-stabilizing task sharing scheme called *mode switching*. Unlike the task-sharing techniques in the past literature [15, 19], it *dynamically* changes the mode of each agent during the election phase. The two modes respectively correspond to synchronization and QE, and each agent is engaged in the task associated with its own mode. In total, the protocol is equipped with three different roles of agents, i.e., check phase, synchronization in election phase, and QE in election phase. We call each role a *class* of agents, and they are respectively referred to as *checker*, *synchronizer*, and *elector*. It should be noted that dynamic mode change is crucial for attaining loose stabilization: A non-correct initial configuration filled by electors obviously causes a deadlock because the timeout of the election phase never occurs. Thus, it is indispensable to install a mechanism that changes electors to synchronizers. That mechanism, however, prevents the quick propagation of the maximum level in QE owing to the lack of a sufficiently large number of electors (recall that even agents not involved in the lottery game must work as a medium in the epidemic). In fact, if only $o(n)$ electors remain, we cannot guarantee with very high probability that the epidemic of the maximum level finishes quickly. This observation implies that the mode change from synchronizers to electors is also necessary.

The remaining concern is how to design synchronization and QE with adapting to dynamic change. The task of QE is robust for such dynamics if an agent with mode change always joins as a follower. However, CHVP is not robust because the countdown timer is rewound by a newly joining agent with a high counter value. Fortunately, we can obtain an alternative solution for this matter: simply using a local countdown timer, which just counts the number of interactions performed by the timer holder. While CHVP is necessary to recover global synchronization from the highly deviated situations where two agents are in different phases or have two counter values with a large difference, we can delegate such a role entirely to the check phase. Then, the election phase can use the timeout mechanism not necessarily synchronized among all agents.

4.2 Variables and Groups

For describing the protocol, we use two (hard-coded) fixed values, $r_{\max}$ and $b_{\max}$, both of which are $\Theta(\tau m) = \Theta(\tau \log n)$ for sufficiently large hidden constants. We also define $r_{\mathrm{mid}} = cr_{\max}$ for an appropriate $1 > c > 0$ such that $c/(1-c)$ becomes sufficiently large. All

Table 2 Variables used in protocol P_{TO}. Each time an agent changes its class, class-specific variables are set to the initial value specified below. The initial value of a variable `detect` is not a fixed value: the value of a common variable v.`leader` is copied to v.`detect` each time an agent v becomes a checker.

	Variable name	Initial value
Common variables	`leader` $\in \{0,1\}$	-
	`phase` $\in \{CH, EL\}$	-
	`mode` $\in \{A, B\}$	-
Variables for checkers	$\mathtt{timer}_R \in [0, r_{\max}]$	$r_{\max}$
	`detect` $\in \{0,1\}$	`leader`
Variables for electors	`level` $\in [0, 2m]$	0
	`done` $\in \{0,1\}$	0
Variables for synchronizers	$\mathtt{timer}_B \in [0, b_{\max}]$	$b_{\max}$

Table 3 Descriptors for specific subsets of agents.

$V_L = \{v \in V \mid v.\mathtt{leader} = 1\}$, $V_F = \{v \in V \mid v.\mathtt{leader} = 0\}$
$V_{CH} = \{v \in V \mid v.\mathtt{phase} = CH\}$, $V_{EL} = \{v \in V \mid v.\mathtt{phase} = EL\}$
$V_A = \{v \in V_{EL} \mid v.\mathtt{mode} = A\}$, $V_B = \{v \in V_{EL} \mid v.\mathtt{mode} = B\}$
$V_{CH\geq} = \{v \in V_{CH} \mid r_{\mathrm{mid}} \leq v.\mathtt{timer}_R \leq r_{\max}\}$
$V_{CH<} = \{v \in V_{CH} \mid 0 \leq v.\mathtt{timer}_R < r_{\mathrm{mid}}\}$
$V_{\mathrm{done}} = \{v \in V_L \cap V_A \mid v.\mathtt{done} = 1\}$
$V_{\mathrm{undone}} = \{v \in V_L \cap V_A \mid v.\mathtt{done} = 0\}$

the hidden constants are appropriately fixed in the "on-demand" manner in the proof details. We also assume $n \geq 3$ for the simplicity of argument; however, it is not essential. It can be easily observed that this protocol is a self-stabilizing leader election protocol in the case of $n = 2$.

The set of variables used in protocol P_{TO} is shown in Table 2. As stated in Section 4.1, in protocol P_{TO}, there are three classes of agents: checkers, synchronizers, and electors. Each class has a set of variables specific for the associated task, and an agent manages the variables related to its own class as well as the set of common variables. Note that the list of variables in Table 2 contains two $\Theta(\tau \log n)$-state variables ($\mathtt{timer}_R$ and $\mathtt{timer}_B$) and one $\Theta(\log n)$-state variable (`level`), but they are used exclusively. That is, at any configuration, each agent has the responsibility of managing only one of the three. Thus, the total number of states necessary for storing all variables in Table 2 is bounded by $O(\tau \log n)$. The column "Initial value" in Table 2 indicates the initial values set for class-specific variables. The initialization occurs when the agent changes its class. For avoiding unnecessary complication, this initialization process is not explicitly stated in the pseudocode presented later. The class of each agent is identified by two common variables `phase` and `mode`. More precisely, the agent v with $v.\mathtt{phase} = CH$ is a checker, that with $v.\mathtt{phase} = EL$ and $v.\mathtt{mode} = A$ an elector, and that with $v.\mathtt{phase} = EL$ and $v.\mathtt{mode} = B$ a synchronizer. The set of agents belonging to each class is denoted by V_{CH}, V_A, and V_B respectively. In addition, we introduce several notations for describing the set of agents satisfying some condition, as listed in Table 3.

4.3 Details of the Protocol

The pseudocode of P_{TO} is shown in Algorithm 2. The main bodies of the two phases are realized by lines 3 and 16 and the procedure *GoToElection*(). Line 3, which corresponds to the check phase, performs the propagation of detect flags (i.e., the existence of leader

Algorithm 2 P_{TO} (a_0 is an initiator and a_1 is a responder).

```
1  for each i ∈ {0,1} s.t. a_i ∈ V_B do a_i.leader ← 0
2  if a_0, a_1 ∈ V_CH then
3      a_0.detect ← a_1.detect ← max(a_0.detect, a_1.detect)
4      a_0.timer_R ← a_1.timer_R ← max(a_0.timer_R − 1, a_1.timer_R − 1, 0)
5      if a_0.timer_R = 0 then
6          GoToElection(0); GoToElection(1)
7  else if ∃i ∈ {0,1} : a_i ∈ V_EL ∧ a_{1−i} ∈ V_{CH≥} then
8      a_i.phase ← CH                    // Reset checker's variables by Table 2
9  else if ∃i ∈ {0,1} : a_i ∈ V_{CH<} ∧ a_{1−i} ∈ V_EL then
10     GoToElection(i)
11 if a_0, a_1 ∈ V_EL then
12     if a_0, a_1 ∈ V_A ∩ V_F ∧ a_0.level = a_1.level then
13         a_1.mode ← B                  // Reset synchronizer's variables by Table 2
14     else if a_0, a_1 ∈ V_B then
15         a_i.mode ← A where i = max{j ∈ {0,1} | a_j.timer_B ≥ a_{1−j}.timer_B}
                                         // Reset elector's variables by Table 2
16     QE()
17     if a_0, a_1 ∈ V_done ∧ a_0.level = a_1.level then a_1.leader ← 0      // V_done ⊆ V_L
18     for each i ∈ {0,1} s.t. a_i ∈ V_B do a_i.timer_B ← max(a_i.timer_B − 1, 0)
19     for each i ∈ {0,1} s.t. a_i ∈ V_B ∧ a_i.timer_B = 0 do a_i.phase ← CH
                                         // Reset checker's variables by Table 2
20 function GoToElection(i):
21     if a_i.detect = 0 then a_i.leader ← 1
22     (a_i.phase, a_i.mode) ← (EL, A)    // Reset elector's variables by Table 2
```

agents) using the epidemic. Line 16 indeed corresponds to the task of QE. The procedure *GoToElection*() corresponds to the phase transition from check to election, where the agent not detecting the existence of leaders becomes a leader. The remaining part is devoted to the synchronization mechanism including the mode switching scheme. Lines 4-6 correspond to the implementation of CHVP, where the timer variable $\texttt{timer}_R$ is updated (line 4), and the transition to the election phase is triggered when timeout occurs (lines 5 and 6). Lines 7-10 are the mechanism supporting smooth phase transition, which is crucial for guaranteeing the correctness criteria of the synchronization mechanism explained later. Lines 7-8 and 9-10 respectively address the transition from check to election and its reversal. Lines 11-19 correspond to the task for synchronizers and electors. The core of this part is the mode switching scheme, described in lines 12-15. The switch from elector to synchronizer happens when a follower agent interacts with another follower with the same level (lines 12-13), and the opposite occurs when two synchronizers interact with each other (lines 14-15). It is shown in the next section that this scheme appropriately control the size of two classes. Line 18 is the countdown of local timers held by synchronizers, and line 19 is the phase transition from election to check. The leader elimination in line 17 is not for the leader election itself, but rather to handle the initial configurations consisting only of leaders with the same level. Without this code, the protocol would be deadlocked in that case. A synchronizer is always a follower, as guaranteed by line 1.

For stating the precise goal of the synchronization mechanism, we explain its intended behavior as well as the concise reason why such a behavior is attained.

Behavior 1 Starting from any configuration in $\mathcal{C}_{\mathrm{all}}(P_{\mathrm{TO}}(\tau))$, the population quickly reaches a configuration where $V = V_{CH\geq}$ holds with very high probability. The CHVP protocol shrinks the large deviation among all timers in the check phase. Therefore, once an agent goes back to the check phase from the election phase and resets its $\texttt{timer}_R$ to $r_{\max}$, the population quickly reaches a configuration where $V = V_{CH\geq}$. One may think that the population gets stuck in the election phase once it reaches a configuration where all agents are electors (i.e., there is no synchronizers). However, even starting from such a configuration, the population quickly creates at least one synchronizer because the following events occur with very high probability: (i) all leaders quickly decide their levels; (ii) the maximum level quickly propagates to the whole population as long as there is no synchronizer; (iii) since all the agents have the same level, the number of followers quickly becomes $\Theta(n)$ as long as there is no synchronizer; and (iv) two followers with the same level have an interaction quickly and one of them becomes a synchronizer. Once a synchronizer is created, some agent quickly goes back to the check phase because each synchronizer simply counts down its local timer.

Behavior 2 Once the current configuration satisfies $V = V_{CH\geq}$, CHVP decreases timer values (i.e., $\texttt{timer}_R$) while maintaining a relatively smaller deviation among agents. Since timer values of agents in $V_{CH\geq}$ are all $\Theta(\tau \log n)$, the check phase continues during $\Theta(\tau n \log n)$ steps with very high probability. When an agent is timed out, it moves to the election phase. Then, owing to the low deviation of CHVP timers, no agent is still in $V_{CH\geq}$, and thus, the transition in lines 9-10 quickly takes all other agents to the election phase with very high probability. During this period, no agent goes back to the check phase from the election phase with very high probability because the upper limit $b_{\max}$ of $\texttt{timer}_B$ is $\Theta(\tau \log n)$ with a sufficiently large hidden constant.

Behavior 3 In the election phase, the fastest timer (i.e., the agent with the smallest timer value) of all synchronizers determines the pace. Since it is never rewound, the election phase keeps $\Theta(\tau n \log n)$ steps with very high probability. Similar to the behavior from check to election, when an agent becomes a checker, all other agents are quickly brought back to the check phase with very high probability. During this period, no agent goes to the election phase from the check phase with very high probability because the upper limit $r_{\max}$ of $\texttt{timer}_R$ is $\Theta(\tau \log n)$ with a sufficiently large hidden constant.

The correctness criteria of the synchronization mechanism is that the system iterates Behaviors 2 and 3 with very high probability after recovery from unintended situations (by Behavior 1), which is necessary for our protocol to elect a unique leader in the loosely-stabilizing manner. The formal proof of the correctness is given in the next section.

In $QE()$, we expect that the largest level is quickly propagated to all leaders with very high probability. Sudo et al. [19] proved that this is true if $|V_A| = \Theta(n)$ holds and V_A remains the same during this period. However, P_{TO} frequently executes the mode switching from A to B and from B to A. Without the mode switching, the number of agents with $\texttt{level} = s_{\max}$ is monotonically non-decreasing, while with the mode switching, it decreases when an agent with $\texttt{level} = s_{\max}$ changes its mode from A to B. Therefore, we must evaluate the effect of the mode switching on the speed of the propagation. Fortunately, there is no severe effect of the mode switching for our purpose: every leader in V_{done} whose $\texttt{level}$ is not the largest becomes a follower within $O(n \log n)$ steps with probability $1 - o(1)$.

5 Analysis

To express claims in a formal manner, we first define the following notations.

- $\mathcal{A}_X$: the set of all configurations $\mathcal{C}_{\mathrm{all}}(P_{\mathrm{TO}}(\tau))$ where $V_X = V$ holds. For example, $\mathcal{A}_{CH}$ is the set of all configurations where every agent is in the check phase (i.e., $V_{CH} = V$).
- $\mathcal{C}_{EL\geq}$: the set of all configurations in $\mathcal{A}_{EL}$ where $v.\mathtt{timer}_B \geq b_{\max}/2$ holds for every $v \in V_B$.
- $\mathcal{C}_{\mathrm{reset}}$: the set of all configurations in $\mathcal{C}_{\mathrm{all}}(P_{\mathrm{TO}}(\tau))$ where there is at least one agent $v \in V_{CH}$ such that $v.\mathtt{timer}_R = r_{\max}$.

The first goal of this section is to prove the following three lemmas (Lemmas 6, 7, 8). Intuitively, Lemma 6 claims that synchronization is recovered quickly with very high probability from any configuration in $\mathcal{C}_{\mathrm{all}}(P_{\mathrm{TO}}(\tau))$, and Lemmas 7 and 8 claim that once the synchronization is recovered, the check phase and the election phase are iterated thereafter, both taking $\Theta(\tau n \log n)$ steps with sufficiently large hidden constants, with very high probability.

In the rest of this paper, for any set $\mathcal{C}$, we say that an execution *enters* $\mathcal{C}$ when it reaches a configuration in $\mathcal{C}$.

▶ **Lemma 6.** *Let C_0 be any configuration in $\mathcal{C}_{\mathrm{all}}(P_{\mathrm{TO}}(\tau))$ and let $\Xi = \Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \boldsymbol{\Gamma})$. Execution Ξ enters $\mathcal{A}_{CH\geq}$ quickly with very high probability.*

▶ **Lemma 7.** *Let C_0 be any configuration in $\mathcal{A}_{CH\geq}$ and let $\Xi = \Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \boldsymbol{\Gamma})$. Then, the following hold with very high probability:*

1. *execution Ξ enters $\mathcal{C}_{EL\geq}$ quickly,*
2. *no agent moves from the election phase to the check phase before Ξ enters $\mathcal{C}_{EL\geq}$, and*
3. *execution Ξ stays in $\mathcal{A}_{CH}$ for $\Omega(nr_{\mathrm{mid}}) = \Omega(\tau n \log n)$ steps.*

▶ **Lemma 8.** *Let C_0 be any configuration in $\mathcal{C}_{EL\geq}$ and let $\Xi = \Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \boldsymbol{\Gamma})$. Then, the following hold with very high probability:*

1. *execution Ξ enters $\mathcal{A}_{CH\geq}$ quickly,*
2. *no agent moves from the check phase to the election phase before Ξ enters $\mathcal{A}_{CH\geq}$, and*
3. *execution Ξ stays in $\mathcal{A}_{EL}$ for $\Omega(nr_{\mathrm{mid}}) = \Omega(\tau n \log n)$ steps.*

In what follows, we first prove Lemma 6 by giving four supplemental lemmas (Lemmas 9, 10, 11, and 12). We next prove Lemmas 7 and 8.

▶ **Lemma 9.** *Starting from any configuration $C_0 \in \mathcal{A}_{EL}$, execution $\Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \boldsymbol{\Gamma})$ quickly enters $\mathcal{C}_{\mathrm{reset}}$ or reaches a configuration in $\mathcal{A}_{EL}$ satisfying $V_{\mathrm{undone}} = \emptyset$ with very high probability.*

Proof. Let $\Xi = \Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \boldsymbol{\Gamma})$. When an agent goes back to the check phase from the election phase, it substitutes $r_{\max}$ for its $\mathtt{timer}_R$ (lines 8 and 19). Therefore, Ξ never leaves $\mathcal{A}_{EL}$ until it enters $\mathcal{C}_{\mathrm{reset}}$. Let v be any agent that satisfies $v \in V_{\mathrm{undone}}$ in C_0. As long as $v \in V_{\mathrm{undone}}$ holds, v makes a coin flip every time v has an interaction. Since every agent joins an interaction with probability $2/n$ at each step, by the Chernoff bound, v has $2 \log n$ or more interactions within sufficiently large $O(n \log n)$ steps with probability $1 - O(1/n^2)$. Therefore, $v.\mathtt{done} = 1$ holds within $O(n \log n)$ steps with probability $1 - (1/2)^{2 \log n} - O(1/n^2) = 1 - O(1/n^2)$ because each coin flip results in "tail" with probability exactly $1/2$, by which v leaves V_{undone}. By the union bound, execution Ξ enters $\mathcal{C}_{\mathrm{reset}}$ or reaches a configuration in $\mathcal{A}_{EL}$ satisfying $V_{\mathrm{undone}} = \emptyset$ within $O(n \log n)$ steps with probability $1 - O(1/n)$. We obtain the lemma by repeating this analysis τ times. ◀

▶ **Lemma 10.** *Starting from any configuration $C_0 \in \mathcal{A}_{EL}$ satisfying $V_{\mathrm{undone}} = \emptyset$, execution $\Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \boldsymbol{\Gamma})$ quickly reaches a configuration in $\mathcal{A}_{EL}$ satisfying $V_B \neq \emptyset$ with very high probability.*

Proof. Let $\Xi = \Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \mathbf{\Gamma})$. In execution Ξ, no leader has $\texttt{done} = 0$ before Ξ reaches a configuration in $\mathcal{A}_{EL}$ where $V_B \neq \emptyset$. Therefore, it suffices to show that Ξ reaches a configuration in $\mathcal{A}_{EL}$ where $V_B \neq \emptyset$ within $O(n \log n)$ steps with high probability because we obtain the lemma by repeating this trial τ times.

While both $V_{\mathrm{undone}} = \emptyset$ and $V_B = \emptyset$ hold, nothing prevents the epidemic from propagating the maximum value of `levels`. Thus, by Lemma 2, the maximum value is propagated to the whole population within $O(n \log n)$ steps with high probability [6]. Once all agents have the same level, the number of followers increases by one every time two leaders meet (line 17). As long as $|V_F| < n/2$ and $V_B = \emptyset$ hold, two leaders meet each other with probability at least $1/4$ at each step. Hence, $|V_F| \geq n/2$ or $V_B \neq \emptyset$ holds within $O(n \log n)$ steps with high probability. In the former case, at each step thereafter, two followers have an interaction and one of them becomes a synchronizer (line 13) with a constant probability. Therefore, $|V_B| \neq \emptyset$ holds within $O(\log n)$ steps with high probability. ◀

▶ **Lemma 11.** *Starting from any configuration $C_0 \in \mathcal{A}_{EL}$ satisfying $V_B \neq \emptyset$ holds, execution $\Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \mathbf{\Gamma})$ quickly enters $\mathcal{C}_{\mathrm{reset}}$ with very high probability.*

Proof. Before $\Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \mathbf{\Gamma})$ enters $\mathcal{C}_{\mathrm{reset}}$, the smallest $\texttt{timer}_B$ in the population (i.e., $\min_{v \in V_B} \texttt{timer}_B$) is monotonically non-increasing. It decreases by one or a timeout of $\texttt{timer}_B$ occurs when a synchronizer with the smallest $\texttt{timer}_B$ has an interaction (line 18), which occurs with probability at least $2/n$ at each step. Since $b_{\max} = \Theta(\tau \log n)$, by the Chernoff bound, some synchronizer encounters the timeout of $\texttt{timer}_B$ quickly with very high probability. ◀

▶ **Lemma 12.** *Starting from any configuration $C_0 \in \mathcal{C}_{\mathrm{all}}(P_{\mathrm{TO}}(\tau))$, execution $\Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \mathbf{\Gamma})$ quickly enters $\mathcal{C}_{\mathrm{reset}}$ with very high probability.*

Proof. Let $\Xi = \Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \mathbf{\Gamma})$. Note that Ξ enters $\mathcal{C}_{\mathrm{reset}}$ whenever an agent goes back to the check phase from the election phase. Before Ξ enters $\mathcal{C}_{\mathrm{reset}}$ or $\mathcal{A}_{EL}$, $\max_{v \in V_{CH}} v.\texttt{timer}_R$ is monotonically non-increasing and this value decreases at a pace faster than or equal to the pace at which the maximum value of variable y decreases in the CHVP protocol in Section 3. Therefore, by Lemma 3 with $l_1 = r_{\max}$, $l_2 = 0$, $k = \tau$, Ξ reaches a configuration $C' \in \mathcal{C}_{\mathrm{reset}} \cup \mathcal{A}_{EL}$ quickly with very high probability. Once Ξ enters $\mathcal{A}_{EL}$, it enters in $\mathcal{C}_{\mathrm{reset}}$ quickly with very high probability by Lemmas 9, 10, and 11. ◀

Proof of Lemma 6. By Lemma 12, we can assume that $C_0 \in \mathcal{C}_{\mathrm{reset}}$. Since we assume $r_{\max} - r_{\mathrm{mid}} = O(\tau \log n)$ with a sufficiently large hidden constant, by Lemma 4 with $l = r_{\max}$ and $k = \tau$, execution $\Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \mathbf{\Gamma})$ enters $\mathcal{A}_{CH_{\geq}}$ quickly with very high probability. ◀

Proof of Lemma 7. The last claim is trivial because each agent has an interaction with probability $2/n$ at each step and at least one agent must have r_{mid} interactions until execution Ξ leaves $\mathcal{A}_{CH}$. We prove the first and the second claims below.

By Lemma 3 with $k = \tau$, Ξ enters $\mathcal{A}_{CH_<}$ within $O(n(r_{\max} - r_{\mathrm{mid}})) = O(\tau n \log n)$ steps or at least one agent goes to the election phase during the period with very high probability. Since we assume that $r_{\mathrm{mid}}/(r_{\max} - r_{\mathrm{mid}})$ is a sufficiently large constant, together with the third claim and the union bound, we observe that Ξ enters $\mathcal{A}_{CH_<}$ quickly with very high probability. Thereafter, by Lemma 3 with $k = \tau$, at least one agent goes to the election phase within $O(\tau n \log n)$ steps with very high probability. Remember that whenever an agent in $V_{CH_<}$ and an agent in V_{EL} meet, the former moves to the election phase. Hence, once an agent goes to the election phase, the agents in the population go to the election phase one after another in completely the same way as the epidemic protocol in Section 3.1.

Therefore, by Lemma 2 with $k = \tau$, Ξ reaches a configuration $C \in \mathcal{A}_{EL}$ quickly with very high probability. Since $b_{\max}$ is sufficiently large, no agent has more than $b_{\max}/2$ interactions during this period with very high probability. Therefore, $C \in \mathcal{C}_{EL\geq}$ holds with very high probability. From the above, we conclude that the first and second claims also hold. ◀

Proof of Lemma 8. The last claim is trivial because each agent has an interaction with probability $2/n$ at each step and at least one agent must have $b_{\max}/2$ interactions until execution Ξ leaves $\mathcal{A}_{EL}$. By Lemmas 9, 10, and 11, Ξ reaches a configuration $\mathcal{C}_{\mathrm{reset}}$ within $O(\tau n \log n)$ steps with very high probability. Thereafter, in completely the same way as given in the second paragraph of the proof of Lemma 7, we can prove that Ξ enters $\mathcal{A}_{CH\geq}$ quickly (by the epidemic) and $V_{CH<} = \emptyset$ always holds during this period with very high probability. Thus, the first claim holds. No agent goes to the election phase when it belongs to $V_{CH\geq}$, from which the second claim follows. ◀

▶ **Lemma 13.** *Let C_0 be a configuration in $\mathcal{A}_{EL}$ where $V_{\mathrm{undone}} = \emptyset$ holds and let $\Xi = \Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \mathbf{\Gamma})$. Let $V' \subset V_L \cap V_A$ be the set of leaders whose* `level` *are not the largest in C_0. Then, execution Ξ reaches a configuration in $\mathcal{A}_{EL}$ where all agents in V' are followers or enters $\mathcal{C}_{\mathrm{reset}}$ within $O(n \log n)$ steps with probability $1 - o(1)$.*

Proof. In this proof, we ignore the case that some agent goes back to the check phase (i.e., Ξ enters $\mathcal{C}_{\mathrm{reset}}$) because this ignorance only decreases the probability claimed in the lemma. Let l be the maximum level of the population (i.e., $\max_{v \in V_A} v.\mathtt{level}$) in C_0. To obtain the lemma, it suffices to show that all leaders in V' observe the maximum level l and become followers within $O(n \log n)$ steps with probability $1 - o(1)$.

First, we analyze $n_A = |V_A|$ in execution Ξ. This value increases by one if two agents in V_B meet, and it decreases by one if two followers with the same level in V_A meet. Therefore, at each step where $n_A \leq n/3$, n_A increases with probability at least $4/9$, while n_A decreases with probability at most $1/9$. The gap of these probabilities and the Chernoff bound guarantee that even if $n_A < n/3$ in C_0, n_A reaches $n/3$ within $O(n)$ steps with high probability. Let C' be the configuration at this time. Once Ξ reaches C', the above gap of probabilities, the Chernoff bound, and the union bound guarantee that $n_A \geq n/4$ always holds for arbitrarily large $\Omega(n \log n)$ steps with high probability. Thus, we can assume $n_A \geq n/4$ in the following discussion on execution Ξ after C'.

Consider the suffix of Ξ after C'. Let $n_M = |\{v \in V_A \mid v.\mathtt{level} = l\}|$. This value increases by one if an interaction happens between two agents in V_A such that one has the maximum level l and the other has a lower level. It decreases by one if an interaction happens between two followers in V_A, both with level l. Note that $n_M \geq 1$ always holds because n_M decreases only if two agents with level l have an interaction. Since we assume $n_A \geq n/4$, at each step where $n_M \leq n/8$, n_M increases with probability $p_{\mathrm{inc}} \geq (n_M \cdot (n/4 - n_M))/\binom{n}{2} \geq n_M/(4n)$, while n_M decreases with probability $p_{\mathrm{dec}} \leq n_M^2/n^2$. As long as $n/2^9 \leq n_M \leq n/2^8$, we have $p_{\mathrm{inc}} \geq 1/2^{11}$ and $p_{\mathrm{dec}} \leq 1/2^{16}$. This large difference between p_{inc} and p_{dec} guarantees that once n_M reaches $n/2^8$, $n_M \geq n/2^9$ always holds for arbitrarily large $\Omega(n \log n)$ steps with high probability, by the Chernoff bound and the union bound. During this period, each leader in V' meets an agent in V_A with the maximum level l with probability $\Omega(1/n)$ at each step. Thus, once $n_A \geq n/2^8$ holds, all leaders in V' become followers within $O(n \log n)$ steps with high probability.

Thus, all we have to do is to show that $n_M \geq n/2^8$ holds within $O(n \log n)$ steps starting from C'. First, we show that n_M reaches $24 \ln n$ or larger within $O(n \log n)$ steps starting from C'. When $n_M < 24 \ln n$, $p_{\mathrm{inc}} = \Omega(1/n)$ and $p_{\mathrm{dec}} = O((\log^2 n)/n^2)$ always hold. Therefore, by the Chernoff bound, n_M reaches $24 \ln n$ or larger within $O(n \log n)$ steps with high probability.

Next, for any integer k such that $24 \ln n \leq k < n/2^8$, we show that once n_M reaches k, n_M reaches $2k$ with high probability. As long as $k/2 \leq n_M \leq 2k$, we have $p_{\mathrm{inc}} \geq k/8n$ and $p_{\mathrm{dec}} \leq 4k^2/n^2 < k/64n$. Therefore, by the Chernoff bound, we have the followings:

- during the first $16n$ steps, n_M is always $k/2$ or larger with probability at least $1 - e^{-(1/3)\cdot(k/4)} = 1 - O(1/n^2)$,
- during the first $x \geq 16n$ steps, n_M increases at least $xk/16n$ times with probability $1 - O(1/n^2)$, and
- during the first $x \geq 16n$ steps, n_M decreases at most $xk/32n$ times with probability $1 - O(1/n^2)$.

Therefore, by the union bound (for $x = 16n, 16n+1, \ldots, 32n$), n_M reaches $k + (2k - k) = 2k$ within $32n$ steps with high probability. Therefore, once n_M reaches $24 \ln n$ or larger value, it doubles in every $32n$ steps with high probability until it reaches $n/2^8$. Thus, n_M reaches $n/2^8$ within $O(n \log n)$ steps with probability $1 - O((\log n)/n) = 1 - o(1)$. ◀

By the correctness of the synchronization and Lemma 13, we can easily show Lemmas 15 and 16 (see Appendix for a complete proof), where we define a set $\mathcal{S}$ of safe configurations by Definition 14.

▶ **Definition 14** (Safe configurations). *Define $\mathcal{S}$ as the set of all configurations where $V = V_{CH_{\geq}}$ holds, exactly one leader v_l exists in the population, and v_l.*`detect` $= 1$ *holds.*

▶ **Lemma 15.** $\min_{C \in \mathcal{S}} \mathrm{EIH}_{P_{\mathrm{TO}}(\tau)}(C) = \Omega(n^{\tau+1})$.

▶ **Lemma 16.** $\max_{C \in \mathcal{C}_{\mathrm{all}}(P_{\mathrm{TO}}(\tau))} \mathrm{EIC}_{P_{\mathrm{TO}}(\tau)}(C, \mathcal{S}) = O(\tau n \log n)$.

Thus, we obtain the main theorem.

▶ **Theorem 17.** *For any $\tau \in \mathbb{N}^+$, $P_{\mathrm{TO}}(\tau)$ is an $(O(\tau n \log n), \Omega(n^\tau))$-LS-LE protocol.*

6 Conclusion

We gave a time-optimal LS-LE protocol in the population protocol model. Given a design parameter $\tau \geq 1$ and integer $N \geq n$ such that N is at most polynomial in n, the proposed protocol elects the unique leader within $O(\tau \log n)$ parallel time starting from any configuration and keeps it for $\Omega(n^\tau)$ parallel time, both in expectation.

References

1. Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. Time-space trade-offs in population protocols. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2560–2579. SIAM, 2017.
2. Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2221–2239. SIAM, 2018.
3. Dan Alistarh, Bartłomiej Dudek, Adrian Kosowski, David Soloveichik, and Przemysław Uznański. Robust detection in leak-prone population protocols. In *International Conference on DNA-Based Computers*, pages 155–171. Springer, 2017.
4. Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming*, pages 479–491, 2015.
5. 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.
6. Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, 2008.

7 Dana Angluin, James. Aspnes, Michael J Fischer, and Hong Jiang. Self-stabilizing population protocols. *ACM Transactions on Autonomous and Adaptive Systems*, 3(4):13, 2008.

8 Petra Berenbrink, George Giakkoupis, and Peter Kling. Optimal time and space leader election in population protocols. In *Proceedings of 52nd Annual ACM Symposium on Theory of Computing*, 2020.

9 Janna Burman, Ho-Lin Chen, Hsueh-Ping Chen, David Doty, Thomas Nowak, Eric Severson, and Chuan Xu. Time-optimal self-stabilizing leader election in population protocols. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 33–44, 2021.

10 Shukai Cai, Taisuke Izumi, and Koichi Wada. How to prove impossibility under global fairness: On space complexity of self-stabilizing leader election on a population protocol model. *Theory of Computing Systems*, 50(3):433–445, 2012.

11 Hsueh-Ping Chen and Ho-Lin Chen. Self-stabilizing leader election. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 53–59, 2019.

12 Hsueh-Ping Chen and Ho-Lin Chen. Self-stabilizing leader election in regular graphs. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 210–217, 2020.

13 David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 31(4):257–271, 2018.

14 Leszek Gąsieniec and Grzegorz Stachowiak. Fast space optimal leader election in population protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2653–2667. SIAM, 2018.

15 Leszek Gąsieniec, Grzegorz Stachowiak, and Przemyslaw Uznanski. Almost logarithmic-time space optimal leader election in population protocols. In *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures*, pages 93–102. ACM, 2019.

16 Taisuke Izumi. On space and time complexity of loosely-stabilizing leader election. In *International Colloquium on Structural Information and Communication Complexity*, pages 299–312, 2015.

17 Yuichi Sudo and Toshimitsu Masuzawa. Leader election requires logarithmic time in population protocols. *Parallel Processing Letters*, 30(01):2050005, 2020.

18 Yuichi Sudo, Junya Nakamura, Yukiko Yamauchi, Fukuhito Ooshita, Hirotsugu. Kakugawa, and Toshimitsu Masuzawa. Loosely-stabilizing leader election in a population protocol model. *Theoretical Computer Science*, 444:100–112, 2012.

19 Yuichi Sudo, Fukuhito Ooshita, Taisuke Izumi, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Time-optimal leader election in population protocols. *IEEE Transactions on Parallel and Distributed Systems*, 2020.

20 Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Loosely-stabilizing leader election on arbitrary graphs in population protocols without identifiers nor random numbers. In *International Conference on Principles of Distributed Systems*, 2015.

21 Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, Toshimitsu Masuzawa, Ajoy K Datta, and Lawrence L Larmore. Loosely-stabilizing leader election for arbitrary graphs in population protocol model. *IEEE Transactions on Parallel and Distributed Systems*, 30(6):1359–1373, 2018.

22 Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, Toshimitsu Masuzawa, Ajoy K Datta, and Lawrence L Larmore. Loosely-stabilizing leader election with polylogarithmic convergence time. *Theoretical Computer Science*, 806:617–631, 2020.

23 Yuichi Sudo, Masahiro Shibata, Junya Nakamura, Yonghwan Kim, and Toshimitsu Masuzawa. Self-stabilizing population protocols with global knowledge. *IEEE Transactions on Parallel and Distributed Systems*, 32(12):3011–3023, 2021.

24 Daisuke Yokota, Yuichi Sudo, and Toshimitsu Masuzawa. Time-optimal self-stabilizing leader election on rings in population protocols. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, (to appear), 2021.

A Omitted proofs

We prove Lemmas 15 and 16 in Sections A.1 and A.2, respectively.

A.1 Holding Time

We prove Lemma 15, which claims $\min_{C\in\mathcal{S}} \mathrm{EIH}_{P_{\mathrm{TO}}(\tau)}(C) = \Omega(n^{\tau+1})$.

Proof of Lemma 15. Let C_0 be any configuration in $\mathcal{S}$ and let $\Xi = \Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \mathbf{\Gamma})$. Since the unique leader v_l in C_0 satisfies $v_l.\texttt{detect} = 1$ and r_{mid} is a sufficiently large $\Theta(\tau \log n)$ value, by Lemma 2 with $k = \tau$ and the third claim of Lemma 7, all agents detect the existence of a leader quickly with very high probability. Therefore, by the first and second claims of Lemma 7, it holds with very high probability that Ξ quickly reaches a configuration $C' \in \mathcal{C}_{EL\geq}$ where v_l is the unique leader and no agent has a higher level than $v_l.\texttt{level}$. This is because v_l goes to the election phase exactly once before Ξ reaches C' with very high probability, and the level of every agent is initialized to zero when it goes to the election phase. Thereafter, v_l never becomes a follower before it goes back to the check phase and goes to the election phase again; this is because only a leader can increase $\max_{v\in V} v.\texttt{level}$. By Lemma 8, Ξ quickly enters $\mathcal{A}_{CH\geq}$ again and v_l does not move to the election phase from the check phase during this period with very high probability. At this time, from the above discussion, v_l is still the unique leader in the population and $v_l.\texttt{detect} = 1$ holds. This means that the population has come back to $\mathcal{S}$.

Now, we observed that an execution of P_{TO} under the uniformly random scheduler $\mathbf{\Gamma}$ starting from any configuration in $\mathcal{S}$ goes back to a configuration in $\mathcal{S}$ after $\Theta(\tau n \log n)$ steps and v_l is always the unique leader during this period with very high probability. Therefore, letting $X = \min_{C\in\mathcal{S}} \mathrm{EIH}_{P_{\mathrm{TO}}(\tau)}(C)$, we have $X \geq (1 - O(n^{-\tau}))(\Theta(\tau n \log n) + X)$. Solving this inequality gives $X = \Omega(\tau n^{\tau+1} \log n) = \Omega(n^{\tau+1})$. ◀

A.2 Convergence Time

We prove Lemma 16, which claims $\max_{C\in\mathcal{C}_{\mathrm{all}}(P_{\mathrm{TO}}(\tau))} \mathrm{EIC}_{P_{\mathrm{TO}}(\tau)}(C, \mathcal{S}) = O(\tau n \log n)$.

Proof of Lemma 16. Let C_0 be any configuration in $\mathcal{C}_{\mathrm{all}}(P_{\mathrm{TO}}(\tau))$ and let $\Xi = \Xi_{P_{\mathrm{TO}}(\tau)}(C_0, \mathbf{\Gamma})$. It suffices to show that Ξ enters $\mathcal{S}$ quickly with a constant probability; this is because, letting $Y = \max_{C\in\mathcal{C}_{\mathrm{all}}(P_{\mathrm{TO}}(\tau))} \mathrm{EIC}_{P_{\mathrm{TO}}(\tau)}(C, \mathcal{S})$, it yields $Y \leq O(\tau n \log n) + (1 - \Omega(1))Y$, and this inequality gives $Y = O(\tau n \log n)$.

We can assume $C_0 \in \mathcal{C}_{EL\geq}$ by Lemmas 6 and 7. By Lemma 8, Ξ reaches a configuration $C' \in \mathcal{A}_{CH\geq}$ within $O(\tau n \log n)$ steps and no agent goes to the election phase from the check phase during this period with very high probability. An agent executes $\texttt{detect} \leftarrow \texttt{leader}$ when it goes back to the check phase (See Table 2). Therefore, after Ξ reaches C', at least one follower becomes a leader when it goes to the election phase if there exists no leader in C'. Moreover, the agents initialize their $\texttt{level}$ and $\texttt{done}$ to 0 when they move to the election phase. Therefore, by Lemmas 5, 7, 8, and 9, Ξ quickly reaches a configuration $C'' \in \mathcal{C}_{EL\geq}$ where exactly one leader, say v_l, has the maximum level with a constant probability. Thereafter, by Lemmas 8 and 13, Ξ reaches a configuration in $\mathcal{A}_{EL}$ where only v_l is a leader within $O(n \log n)$ steps with probability $1 - o(1)$. In the next $O(\tau n \log n)$ steps, Ξ enters $\mathcal{S}$ with very high probability by Lemma 8. Thus, we conclude that Ξ enters $\mathcal{S}$ within $O(\tau n \log n)$ steps with a constant probability. ◀