Smoothed Analysis of Population Protocols

Gregory Schwartzman
JAIST, Nomi, Japan

Yuichi Sudo
Hosei University, Tokyo, Japan

Abstract

In this work, we initiate the study of *smoothed analysis* of population protocols. We consider a population protocol model where an adaptive adversary dictates the interactions between agents, but with probability p every such interaction may change into an interaction between two agents chosen uniformly at random. That is, p-fraction of the interactions are random, while $(1-p)$-fraction are adversarial. The aim of our model is to bridge the gap between a uniformly random scheduler (which is too idealistic) and an adversarial scheduler (which is too strict).

We focus on the fundamental problem of leader election in population protocols. We show that, for a population of size n, the leader election problem can be solved in $O(p^{-2}n\log^3 n)$ steps with high probability, using $O((\log^2 n)\cdot(\log(n/p)))$ states per agent, for *all* values of $p \leq 1$. Although our result does not match the best known running time of $O(n\log n)$ for the uniformly random scheduler ($p = 1$), we are able to present a *smooth transition* between a running time of $O(n \operatorname{polylog} n)$ for $p = 1$ and an infinite running time for the adversarial scheduler ($p = 0$), where the problem cannot be solved. The key technical contribution of our work is a novel *phase clock* algorithm for our model. This is a key primitive for much-studied fundamental population protocol algorithms (leader election, majority), and we believe it is of independent interest.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Dynamic graph algorithms

Keywords and phrases Population protocols, Smoothed analysis, Leader election

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.34

Related Version *Full Version*: https://arxiv.org/abs/2105.12262

1 Introduction

In the traditional population protocol model [5], we have a population of n agents, where every agent is a finite state machine with a small number of states. We refer to the cross product of all of the states of the agents in the population as the *configuration* of the population. Two agents can interact, whereupon their internal states may change as a *deterministic* function of their current states. While the transition function is deterministic, it need not be symmetrical. That is, the interaction is ordered, where one agent is called an *initiator* and the other is called a *responder*. A standard assumption is that the order of agents upon an interaction is chosen uniformly at random. This is equivalent to having a single random bit that can be used by the transition function.

The sequence of interactions that the population undergoes is called a *schedule*, and is decided by a *scheduler*. The standard scheduler used in this model is the *uniformly random* scheduler, which chooses all interactions uniformly at random. Agent states are mapped to outputs via a problem specific output function. Generally, a protocol aims to take any legal initial configuration (legal input) and, after a sufficient number of interactions, turn it into one of a desired set of configurations (legal output). After the population reaches a legal output configuration, every following configuration is also a legal output. The running time of the protocol is an upper bound on the number of interactions (steps) required to map any

35th International Symposium on Distributed Computing (DISC 2021).
Editor: Seth Gilbert; Article No. 34; pp. 34:1–34:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

legal input to a legal output. The model assumes agent interactions happen sequentially, but another common term is *parallel time*, which is the running time of a protocol divided by n. A formal definition of population protocols is given in Section 2.

In this paper, we focus on the *leader election* problem. In this problem, every agent is initially marked as either a *leader* or a *follower*. We are guaranteed that initially there exists at least one leader in the population. The goal is to design a protocol, such that, after a sufficient number of interactions, the population always converges to a configuration with a unique leader. The leader election problem has received a large amount of attention in the population protocol literature [5, 4, 3, 1, 2, 26, 27, 38, 37, 39, 19, 20, 40, 42, 25, 7, 13, 14, 35] and thus makes the perfect case study for our model.

Motivation for our model. Population protocols aim to model the computational power of a population of many weak computational entities. Initially introduced to model animal populations [5] (a flock of birds, each attached with a sensor), the model has found use in a wide range of fields. For example: wireless sensor networks [31, 24], molecular computation (e.g. DNA computing) [21, 18, 16]. The assumption of completely uniform interactions in these models is a reasonable *approximation* to the true nature of the interactions. That is, a flock of birds does not interact uniformly at random, the interaction probability of molecules in a fluid can depend on their size and shape, and sensors in a sensor network may experience delays, malfunctions, or even adversarial attacks. The common thread among all of these scenarios is that, while the uniformity assumption is too strong, these systems still contain some amount of randomness. There is a rich literature on designing population protocols for the uniformly random scheduler, and it would be very disheartening if these results do not generalize if we slightly weaken the scheduler. In this paper we try to model these environments, which are "somewhat noisy", and answer the question: Are current population protocol algorithms robust or fragile?

Smoothed analysis. To this end we consider *smoothed analysis* of population protocols. Smoothed analysis was first introduced by Spielman and Teng [34, 33], in an attempt to explain the fact that some problems admit strong theoretical lower bounds, but in practice are solved on a daily basis. The explanation smoothed analysis suggests for this gap, is that lower bounds are proved using very specific, pathological instances of a problem, which are highly unlikely to happen in practice. They support this idea by showing that some lower bound instances are extremely *fragile*, i.e., a small random perturbation turns a hard instance into an easy one. Spielman and Teng applied this idea to the simplex algorithm, and showed that, while requiring an exponential time in the *worst case*, if we apply a small random noise to our instance before executing the simplex algorithm on it, the running time becomes polynomial in expectation.

While in classical algorithm analysis worst-case analysis currently reigns supreme, the opposite is true regarding population protocols. The vast majority of population protocols assume the uniformly random scheduler. This is due to the fact that under the adversarial scheduler most problems of interest are pathological. This reliance on the uniformly random scheduler leads us to ask the following questions: Is the assumption regarding a uniformly random scheduler too strong? Will the algorithms developed under this assumption fail in the real world? We use smoothed analysis to show that indeed it is possible to design *robust* algorithms for the much-studied leader election problem in population protocols. That is, an algorithm that can provide convergence guarantees even if only a *tiny* percentage of the interactions is random. In doing so we smoothly bridge the gap between the adversarial scheduler and the uniformly random scheduler.

Our model and results. It is easy to see that if *all* of the interactions are chosen adversarially, no problem of interest can be solved. In this paper, we present a model which smoothly bridges the gap between the adversarial scheduler and the uniformly random scheduler. In our model, we have an adaptive adversary which chooses the $(i+1)$-th interaction for the population after the completion of the i-th interaction. This choice can be based on *any* past information of the system (interactions, configurations, random bits flipped). With probability $1-p$ the next interaction is the one chosen by the adversary, and with probability p it is an interaction between two agents chosen uniformly at random. We call p the smoothing parameter. In our analysis we allow protocols to access randomization directly as in [15, 29, 11]. Specifically, we assume that each time two agents interact, they get one (unbiased) random bit. In the appendix, we show how to extend our results even if we only assume that the order of initiator-responder is random (and no random bit is given). This results in a slight slowdown in our convergence time by a $O(p^{-1}\log n)$ multiplicative factor. If we assume a random bit is flipped for every interaction, then the adversary cannot decide the outcome of the random bit (but may decide the initiator-responder order). While if we assume no random bit is flipped, then the adversary cannot decide the initiator-responder order, and it is taken to be random. This model is meant to model an environment that is mostly adversarial, but contains a small amount of randomness.

Throughout this work, we assume that agents know some lower bound for the smoothing parameter p (see section 2 for more details). This might seem like a strong assumption at first glance. Let us provide two examples to motivate this assumption.

- Population protocols are often motivated by biological systems, for example, viruses interacting in a fluid. These biological systems undergo an evolutionary process that allows them to learn the value p as they evolve. Imagine several populations of viruses, each with some different estimate of p. The populations that underestimate or overestimate p will die out, while those with a reasonable estimate of p will remain.
- Consider the case of small artificial agents, such as nanobots or sensors mounted on birds. As these agents are deployed to a physical environment, it is possible to measure the environment beforehand and get some estimate of p. If a direct measurement is impossible, a trial and error approach of guessing p might be sufficient (recall that we only need a reasonable lower bound).

We consider the fundamental problem of leader election in this model and show that we can design a protocol for our model which uses $O((\log^2 n)\cdot(\log(n/p)))$ states per agent, and elects a unique leader in $O(p^{-2}n\log^3 n)$ steps with high probability. Although our result does not match the best known running time of $O(n\log n)$, using $O(\log\log n)$ states, for the uniformly random scheduler ($p=1$), we are able to present a *smooth transition* between a running time of $O(n\,\mathrm{polylog}\,n)$, using $O(\mathrm{polylog}\,n)$ states, for $p=1$ and an infinite running time for the adversarial scheduler ($p=0$), where the problem cannot be solved, regardless the number of states.

Furthermore, this shows that *any* amount of noise in the system is sufficient to guarantee that the leader election problem can be solved if we allow for a sufficient number of states per agent. We also note that because the number of states required is $O((\log^2 n)\cdot(\log(n/p)))$, even for an extremely minuscule amount of noise, $p=1/poly(n)$, the leader election problem can be solved by agents using $\mathrm{polylog}(n)$ states. This is important because we would like our agents to be very simple computational units, so we would like to avoid agents with a super-polylogarithmic number of states.

The key building block in our leader election algorithm is the *phase clock* primitive (see section 3 for a formal definition). This is a weak synchronization primitive, which is at the heart of many state of the art algorithms for fundamental problems like leader election and majority in population protocols [6, 2, 26, 27, 11, 38, 35, 9]. The analysis for all current *phase clock* implementations fails for any constant smoothing parameter $p < 1$, assuming an $O(\text{polylog}(n))$ number of states. Roughly speaking, existing phase clocks break when the adversary chooses two agents and repeatedly forces them to interact (a detailed explanation is given in Section 3.1).

We present a novel phase clock design that is robust even when all but a tiny fraction of the interactions are adversarial. Our phase clock relies heavily on the fact that the random bits flipped per interaction are not chosen in an adversarial fashion. To overcome the shortcomings of existing phase clock algorithms, we base our phase clock on a stochastic process whose correctness is *indifferent* to adversarial interactions. Finally, we show that using our phase clock in a simplified (and slower) version of the leader election algorithm of [38] achieves the desired running time. Although we provide a complete (and simplified) proof of the leader election protocol with our phase clock for completeness, the original analysis [38] still goes through unchanged. That is, our phase clock is basically plug-and-play. Thus, we believe this primitive can be used directly for more complex population protocols such as the complete leader election algorithm of [38], or the majority algorithm of [2, 9]. However, properly presenting and analyzing these algorithms is beyond the scope of this paper, and we leave it for future work.

1.1 Related Work

Smoothed analysis was introduced by Spielman and Teng [34, 33]. Since then, it has received much attention in sequential algorithm design (see the survey in [34]). Recently, smoothed analysis has also received some attention in the distributed setting. The first such application is due to Dinitz et al. [22], who apply it to various well-studied problems in dynamic networks. Since then, different smoothing models [28] and problems [17, 30] were considered. To the best of our knowledge, we are the first to consider smoothed analysis of population protocols.

Leader election has been extensively studied in the population protocol model. The problem was first considered in [5], where a simple protocol was presented. 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 uses only two states per agent and elects a unique leader in $O(n^2)$ steps in expectation. This protocol is time-optimal: Doty and Soloveichik [23] showed that any constant space protocol requires $\Omega(n^2)$ expected steps to elect a unique leader. In a breakthrough result, Alistarh and Gelashvili [3] designed a leader election protocol that converges in $O(n \log^3 n)$ expected steps and uses $O(\log^3 n)$ states per agent. Thereafter, a number of papers have been devoted to fast leader election [2, 26, 27, 38, 11]. Gąsieniec, Staehowiak, and Uznanski [27] gave an algorithm that converges in $O(n \log n \log\log n)$ expected steps 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 a $O(n^2/\text{polylog}(n))$ convergence time requires $\Omega(\log\log n)$ states [1]. Sudo et al. [38] gave a protocol that elects a unique leader within $O(n \log n)$ expected steps and uses $O(\log n)$ states per agent. This is time-optimal because any leader election protocol requires $\Omega(n \log n)$ expected steps even if it uses an arbitrarily large number of states and the agents know the exact size of the population [36]. These two protocols were the state-of-the-art until recently, when Berenbrink et al. [11] gave a time and space optimal protocol. In all of the above literature, the stabilization time (i.e., the number of steps it takes to elect a unique leader) is evaluated under the uniformly random scheduler.

Self-stabilizing leader election has also been well studied [4, 37, 39, 19, 20, 40, 42, 25, 7, 13, 14, 35]. In the self-stabilizing setting, we do not assume that all agents are initialized at the beginning of an execution. That is, we must guarantee that a single leader is elected eventually and maintained thereafter even if the population begins an execution from an *arbitrary* configuration. Typically, the population must create a new leader if there is no leader initially, while the population must decrease the number of leaders to one if there are two or more leaders initially. Unfortunately, the self-stabilizing leader election cannot be solved in the standard model [4]. Thus, this problem has been considered (i) by assuming that the agents have global knowledge such as the exact number of agents [14, 13, 40], (ii) by assuming the existence of oracles [25, 7], (iii) by slightly relaxing the requirement of self-stabilization [37, 39, 35], or (iv) by assuming a specific topology of the population such as rings [4, 19, 20, 42].

Several papers on population protocols assume the globally fair scheduler [5, 4, 25, 7, 19]. Intuitively, this scheduler cannot avoid a possible step forever. Formally, the scheduler guarantees that in an infinite execution, a configuration appears infinitely often if it is reachable from a configuration that appears infinitely often. Assuming the fairness condition is very helpful in designing protocols that solve some problem *eventually*, however, it is not helpful in bounding the stabilization time. Thus, the uniformly random scheduler is often assumed to evaluate the time complexities of protocols, as mentioned above. Actually, the uniformly random scheduler is a special case of the globally fair scheduler.

Several papers considered population protocols with some form of noise. In [41] a random scheduler with non-uniform interaction probabilities is proposed, and the problem of data collection is analyzed for this model. While their model generalizes the standard random scheduler, it still does not allow adversarial interactions, and thus is quite different from our model. Sadano et al. [32] introduced and considered a stronger model than the original population protocols under the uniformly random scheduler. In their model, agents can control their moving speeds. Faster agents have a higher probability to be selected by the scheduler at each step. They show that some protocols have a much smaller stabilization time by changing the speeds of agents.

Similarly to us, the authors of [8] also try to answer the question of whether population protocols can function under imperfect randomness. They take a very general approach, which is somewhat different than ours. The main difference is that the randomness of a schedule (a sequence of interactions) is measured as its *Kolmogorov complexity* (the size of the shortest Turing machine which outputs the schedule). Intuitively, the schedule is random if its Kolmogorov complexity is (almost) equal to the length of the schedule. They parameterize the "randomness" of the schedule by a parameter T, where $T = 1$ means that the schedule is completely random, while the randomness decreases as the value of T goes to 0. An *adversary* with parameter T is a scheduler that only generates schedules with parameter T.

They show that any problem which can be solved for $T = 1$ can also be solved for $T < 1$ (imperfect randomness). They also consider the leader election problem, and give upper and lower bounds for the value of T required to solve the problem. Their bounds are not explicit, but are presented as a function of the largest root of a certain polynomial. Apart from a different notion of randomness, their work differs from ours in that it assumes an oblivious adversary (while we consider an adaptive adversary). This makes a direct comparison between our results and those of [8] somewhat tricky. It might be said that we take a somewhat more pragmatic approach, showing a very natural augmentation to the popular random scheduler. This allows us to *explicitly* express the running time and number of states for all values of p (showing that for reasonable values, the performance is very close to that of the random scheduler), while in [8] only existence results are presented.

2 Preliminaries

Population Protocols. A *population* is a network consisting of *agents*. We denote the set of all agents by V and let $n = |V|$. We assume that a population is a complete graph, thus every pair of agents (u, v) can interact, where u serves as the *initiator* and v serves as the *responder* of the interaction. Throughout this paper, we use the phrase "with high probability" to denote a probability of $1 - O(n^{-\alpha})$ for an arbitrarily large constant α.

A *protocol* $P(Q, T, X, Y, \pi_{\text{in}}, \pi_{\text{out}})$ consists of a finite set Q of states, a transition function $T : Q \times Q \times \{0, 1\} \to Q \times Q$, a finite set X of input symbols, a finite set Y of output symbols, an input function $\pi_{\text{in}} : X \to Q$, and an output function $\pi_{\text{out}} : Q \to Y$. The agents are given (possibly different) inputs $x \in X$. The input function π_{in} determines their initial states $\pi_{\text{in}}(x)$. When two agents interact, T determines their next states according to their current states and one bit. The *output* of an agent is determined by π_{out}: the output of an agent in state q is $\pi_{\text{out}}(q)$.

A *configuration* is a mapping $C : V \to Q$ that specifies the states of all the agents. We say that a configuration C changes to C' by the interaction $e = (u, v)$ and a bit b, denoted by $C \overset{(e,b)}{\to} C'$, if $(C'(u), C'(v)) = T(C(u), C(v), b)$ and $C'(w) = C(w)$ for all $w \in V \setminus \{u, v\}$.

Thus, given a configuration C, a sequence of interactions (or ordered pairs of agents) $\{\gamma_i\}_{i=0}^{\infty}$, and a sequence of bits $\{b_i\}_{i=0}^{\infty}$, the *execution* starting from C under $\{\gamma_i\}_{i=0}^{\infty}$ and $\{b_i\}_{i=0}^{\infty}$ is defined as the sequence of configurations $\{C_i\}_{i=0}^{\infty}$ such that $C_i \overset{(\gamma_i, b_i)}{\to} C_{i+1}$. A sequence of interactions is called a *scheule* and will be explained in detailed in the next subsection. We assume that each $b_i \in \{0, 1\}$ is a random variable such that $\Pr[b_i = 1] = 1/2$ and these random bits $b_0, b_1, \ldots$ are independent of each other. That is, upon each interaction the two agents have access to a bit of randomness to decide their new states. In the appendix, we show how to extend our results for the more standard population protocol model where only the order of initiator-responder is random, and no additional randomness is available.

Schedulers. A schedule $\gamma = \{\gamma_i\}_{i=0}^{\infty} = \{(u_i, v_i)\}_{i=0}^{\infty}$ is a sequence of ordered pairs which determines the interactions the population of agents undergoes. Note that although γ is ordered, we use a set notation for simplicity. The schedule is determined by a *scheduler*. In the classical population protocol model, a uniformly random scheduler is used. That is, every pair in γ is chosen uniformly at random. Let us denote this scheduler by Γ_u. One can also consider an *adversarial* scheduler. Such a scheduler creates γ in an adversarial fashion. Let us denote this scheduler by Γ_a. We would like to note that while the sequence of interactions is chosen adversarially by Γ_a, it does not determine the coin flips observed by the agents. This type of scheduler can either be *adaptive* or *oblivious* (non-adaptive). In both cases the adversary has complete knowledge of the initial state of the population and the algorithm executed by the agents. However, for the oblivious case the sequence of interactions, γ, must be chosen *before* the execution of the protocol, while for the adaptive case the interaction γ_i is chosen by the adversary after the execution of the $(i-1)$-th step, with full knowledge of the current state of the population. The difference between an oblivious and an adaptive adversary can also be stated in term of knowledge of the randomness in the population. An adaptive adversary has full knowledge regarding the population, including the random coins used in the past. While the oblivious adversary does not have access to the randomness of the system.

Our model. We consider a smoothed scheduler Γ_s which is a combination of Γ_u and Γ_a. Specifically, let $\gamma^a = \{\gamma_i^a\}_{i=0}^{\infty}, \gamma^u = \{\gamma_i^u\}_{i=0}^{\infty}$ be the schedules chosen by Γ_a, Γ_u. We define the smoothed schedule $\gamma_s = \{\gamma_i^s\}_{i=0}^{\infty}$ of Γ_s as $\gamma_i^s = \gamma_i^u$ with probability p and $\gamma_i^s = \gamma_i^a$ with probability $1-p$, where $p \in [0,1]$ is the *smoothing parameter*. We note that if Γ_a is adaptive, then so is Γ_s. For the rest of the paper we focus on an *adaptive* adversary.

In this paper, we assume that a rough knowledge of an upper bound of n, p^{-1} is available. Specifically, we assume that all agents know two common values n', p', such that $n \leq n' = O(n), p \geq p' = \Omega(p)$. Assuming such a rough knowledge about n is standard in the recent population protocol literature [3, 1, 2, 27, 29, 12, 11, 37, 38, 39, 35, 9], and we generalize this assumption for p. Due to the asymptotic equivalence between n, p and n', p', we only use only n, p in the definition and analysis of our algorithm.

Leader election. The leader election problem requires that every agent should output L or F ("leader" or "follower") respectively. We say that a configuration C of P is output-stable if no agent may change its output in an execution of P that starts from C, regardless of the choice of interactions. Let $\mathcal{S}_P$ be the set of the output-stable configurations such that, for any configuration $C \in \mathcal{S}_P$, exactly one agent outputs L (i.e., is a leader) in C. This problem does not require inputs for the agents. Hence, we assume $X = \{x\}$, thus all the agents begin an execution with a common state $s_{\text{init}} = \pi_{\text{in}}(x)$. We say that a protocol P is a leader election protocol for a scheduler Γ, if the execution of the protocol starting from the configuration where all agents are in state s_{init} reaches a configuration in $\mathcal{S}_P$ with probability 1 with respect to the scheduler Γ. We define the stabilization time of the execution as the number of steps until it reaches a configuration in $\mathcal{S}_P$ for the first time.

One-way Epidemic. In the proposed protocol, we often use the *one-way epidemic* protocol[6]. This is a population protocol where every agent has two states $\{0,1\}$ and the transition function is given as $(x,y) \to (x, \max\{x,y\})$. All nodes with value 1 are *infected*, while all nodes with value 0 are *susceptible*. Initially, we assume that a single node is infected. We say that the one-way epidemic finishes when all nodes are infected. This is an important primitive for spreading a piece of information among the population.

Angluin et al. [6] prove that one-way epidemic finishes within $\Theta(n \log n)$ interactions with high probability for Γ_u. It is easy to see that the one-way epidemic protocol finishes within $O(p^{-1} n \log n)$ steps for Γ_s with high probability. This is because within $O(p^{-1} n \log n)$ steps of Γ_s there must exist $\Omega(n \log n)$ *random* interactions with high probability. Due to the nature of the one-way epidemic protocol, we can just ignore all adversarial interactions, and the original analysis goes through.

Martingale concentration bounds. In our analysis we often encounter the following scenario: We have a series of *dependent* binary random variables $\{X_i\}_{i \geq 0}$ such that $\forall i, E[X_i \mid X_0, ..., X_{i-1}] \geq q$, for some constant q.[1] And we would like to bound the probability $Pr[\sum_{i=0}^{\lceil \alpha q^{-1} t \rceil} X_i \leq t]$ for some constant α. Note that if the variables were independent, we could have simply used a Chernoff type bound. As this is not the case, we use martingales for our analysis.

[1] Note that this condition is equivalent to $\forall i, Pr[X_i = 1 \mid X_0 = s_0, ..., X_{i-1} = s_{i-1}] \geq q$ for any binary string s of length $i-1$.

We say that a sequence of random variables, $\{Y_i\}_{i=0}$, is a sub-martingale with respect to another sequence of random variables $\{X_i\}_{i\geq 0}$ if it holds that $\forall i, E[Y_i \mid X_0, ..., X_{i-1}] \geq Y_{i-1}$. The following concentration equality holds for sub-martingales:

▶ **Theorem 1** (Azuma). *Suppose that $\{Y_i\}_{i\geq 0}$ is a sub-martingale with respect to $\{X_i\}_{i\geq 0}$, and that $|Y_i - Y_{i-1}| \leq c_i$. Then for all positive integers k and positive reals ϵ it holds that:*

$$Pr[Y_k - Y_0 < -\epsilon] \leq e^{\frac{-\epsilon^2}{2\sum_{j=1}^{k} c_j}}$$

Let us consider the sequence $\{X_i\}_{i\geq 0}$ from before. Recall that $\forall i, E[X_i \mid X_0, ..., X_{i-1}] \geq q$. Without loss of generality assume that $X_0 = 0$.

Let us define $Y_i = \sum_{j=0}^{i} X_j - q \cdot i$. Note that $Y_0 = 0$. Let us show that $\{Y_i\}_{i\geq 0}$ is a a submartingale with respect to $\{X_i\}_{i\geq 0}$. It holds that:

$$E[Y_{i+1} \mid X_0, ..., X_i] = E[X_{i+1} - q + Y_i \mid X_0, ..., X_i] = E[X_{i+1} \mid X_0, ..., X_i] - q + Y_i > Y_i$$

Where in the transitions we used the fact that the variables $X_0, .., X_i$ completely determine Y_i, thus $E[Y_i \mid X_0, ..., X_i] = Y_i$, and the fact that $E[X_i \mid X_0, ..., X_{i-1}] \geq q$. Next we apply Azuma's inequality for $Y_{\lceil 2q^{-1}t \rceil}$ by setting $\epsilon = t$, noting that $\forall i, |Y_{i+1} - Y_i| \leq 1$.

$$\begin{aligned}\Pr\left[\sum_{i=0}^{\lceil 2q^{-1}t \rceil} X_i < t\right] &\leq \Pr\left[\sum_{i=0}^{\lceil 2q^{-1}t \rceil} X_i - 2t \leq -t\right] \\ &\leq \Pr[Y_{\lceil 2q^{-1}t \rceil} < -t] \leq e^{-t^2/\lceil 2q^{-1}t \rceil} = e^{-\Theta(t)}\end{aligned}$$

We state the following theorem:

▶ **Theorem 2.** *Let $\{X_i\}_{i\geq 0}$ be a series of binary random variables such that $\forall i, E[X_i \mid X_0, ..., X_{i-1}] \geq q$ for some constant q. Then it holds that for every positive integer t:* $\Pr\left[\sum_{i=0}^{\lceil 2q^{-1}t \rceil} X_i < t\right] \leq e^{-\Theta(t)}$.

Specifically, when we set $t = \Theta(\log n)$ with a sufficiently large constant we get a high probability bound.

3 Phase clock implementation for Γ_s

A phase clock is a weak synchronization primitive used in population protocols. In a phase clock we would like all of the agents to have a variable, let's call it *hour*, with the following properties:

1. All agents simultaneously spend $\Omega(f(n))$ steps in the same hour.
2. For every agent an hour lasts $O(g(n))$,

Where the above holds with high probability for every value of hour. Ideally, we desire $f(n) = g(n)$.

We borrow some notation from [10], and define the above more formally. A *round* is a period of time during which all agents have the same *hour* value. Denote by $R_s(i), R_e(i)$ the start and end of round i. Formally, $R_s(i)$ is the interaction at which the last agent reaches hour i, while $R_e(i)$ is the interaction during which the first agent reaches hour $i+1$. We define the length of round i as $L(i) = \max\{0, R_e(i) - R_s(i)\}$. Note that it may be the case that $R_e(i) \leq R_s(i)$, thus a max is needed in the definition. Finally, during these $L(i)$ interactions, all agents have *hour* $= i$. Next we define the stretch of round i as $S(i) = R_e(i) - R_e(i-1)$.

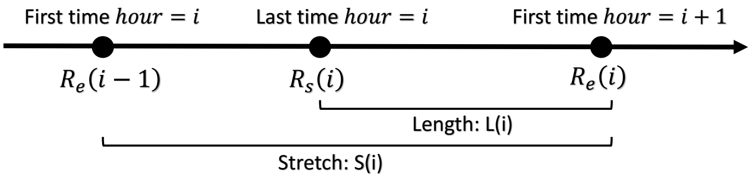

Figure 1 A visual representation of the length and stretch of a round.

This is the amount of time since $hour = i$ is reached for the first time until $hour = i + 1$ is reached for the first time. Note that $L(i) \leq S(i)$ always holds. For a visual representation, we refer the reader to Figure 1. Using the above notation, we define a phase clock.

▶ **Definition 3.** *We say that an algorithm is a phase clock with parameters $f(n)$ and $g(n)$ (or a $(f(n), g(n))$-phase clock) if it has the following guarantees with high probability for any $i \geq 0$:*

1. $L(i) \geq d_1 f(n)$
2. $S(i) \leq d_2 g(n)$

Where d_1 and d_2 are adjustable constants (taken to be sufficiently large). When $f(n) = g(n)$ we simply write a $f(n)$-phase clock.

We note that all current phase clock algorithms require the uniformly random scheduler, and do not extend to Γ_s, as we will see in the next subsection.

3.1 Why existing algorithms fail

In this subsection, we show why the existing phase clock algorithms fail in our model.

There are three kinds of phase clock algorithms in the field of population protocols: a phase clock with a unique leader [6], a phase clock with a junta [26, 27, 11, 9], and a leaderless phase clock [2, 38]. The first kind is essentially a special case of the second kind. The second kind is a $\log n$-phase clock that uses only a constant number of states. However, we require the assumption that there is a set $J \subset V$ of agents marked as members of a *junta*, such that $|J| = O(n^{1-\epsilon})$, where ϵ is a constant. The third kind is a $\log n$-phase clock that uses $O(\log n)$ states but does not require the existence of a junta.

In our notation, the second algorithm can be written as follows:

- Each agent has a variable $minute \in \mathbb{N}$.
- Each agent outputs $hour = \lfloor minute/M \rfloor$, where M is a (sufficiently large) constant.
- Suppose that an initiator u and a responder v interact. The initiator u sets its $minute$ to $\max(u.minute, v.minute + 1)$ if u is in the junta; otherwise to $\max(u.minute, v.minute)$.

By the definition of the algorithm, only an agent in the junta can increase $\max_{v \in V} v.minute$. This fact and the sublinear size of the junta guarantees that the length of each round is $\Omega(n \log n)$ with high probability. However, this guarantee depends on the uniformly random nature of the scheduler. In our model, every interaction is chosen adversarially with probability $1 - p$. The adversary can force two agents in the junta, say u and v, to interact so frequently that every round finishes within $O(1/(1-p))$ steps. Thus, unless $p = 1 - O(1/n)$, i.e., unless the adversary can only choose an extremely small fraction of the interactions, the adversary can always force each round to finish in $o(n)$ steps, i.e., $o(1)$ parallel time. In particular, if $p = 1 - \Omega(1)$, the adversary can always force each round to finish in a constant number of steps.

The third algorithm (the leaderless phase clock) can be written as follows[2]:

- Each agent has variables $hour \in \mathbb{N}$ and $minute \in \{0, 1, \ldots, M\}$, where $M = \Theta(\log n)$ with a sufficiently large hidden constant.
- Suppose that an initiator u and a responder v interact. The initiator u updates its $hour$ and $minute$ as follows:

$$(u.hour, u.minute) \leftarrow \begin{cases} (v.hour, 0) & \text{if } u.hour < v.hour \\ (u.hour + 1, 0) & \text{else if } u.minute = M \\ (u.hour, u.minute + 1) & \text{otherwise.} \end{cases}$$

In this phase clock, an agent resets its *minute* to zero each time it increases its *hour*. Once an agent resets its *minute* to zero, it must have no less than M interactions, or interact with an agent whose *hour* is larger than its *hour*, before it increases its *hour*. Thus, one can easily observe that the length of each round is $\Omega(n \log n)$ under the uniformly random scheduler. However, in our model, this does not hold. The adversary can pick two agents and force them to interact frequently, so that every round finishes within $O((\log n)/(1-p))$ steps, i.e., $O((\log n)/(n(1-p)))$ parallel time.

3.2 Our algorithm

We present a $((np^{-1} \log^2 n), (np^{-2} \log^2 n))$-phase clock using $O((\log n) \cdot \log(n/p))$ states per agent, where p is the smoothing parameter for Γ_s.

In our algorithm each agent v has three states: $second, minute, hour$. Where *hour* is the output variable. The domain of the variables is: $second \in \{0, ..., S\}, minute \in \{0, ..., M\}$ where $S = \log(n/p) + \log\log n + c, c = O(1)$ and $M = \Theta(\log n)$. For simplicity of notation and without loss of generality, we assume that $1/p, \log n, M, S, c$ are all integers. While the domain of *hour* is unbounded in our algorithm, it can easily be taken to be bounded (By using a simple modulo operation [10], or by stopping the counter once it reaches some upper limit [26, 27]). All variables are initialized to 0. For each interaction, we apply Algorithm 1, where u is the initiator and v is the responder. Roughly speaking, the *second* variable follows the following random walk pattern:

$$second \leftarrow \begin{cases} second + 1, & \text{with probability } 1/2 \\ 0, & \text{with probability } 1/2 \end{cases}$$

When it reaches S, the *minute* variable is incremented, and *second* is reset back to 0. When *minute* reaches M, the *hour* variable is incremented and both other variables are reset to 0. Finally, the *hour* and *minute* variables are spread via the one-way epidemic process. By doing so, every agent learns the maximum *hour* value in the system, and the maximum *minute* value for its current *hour*.

The main innovation in our algorithm is the increment pattern that the *second* variable undergoes. Our increment pattern guarantees that the *second* variable is robust to adversarial interactions. What dictates the speed of the increment is the *total number* of interactions in the system.

In the following section, we show that indeed our algorithm is a phase clock with round length $\Theta(np^{-1} \log^2 n)$ and stretch of $\Theta(np^{-2} \log^2 n)$.

[2] The implementation of this phase clock slightly differs between [2] and [38]. Here we describe the implementation presented in [38].

Algorithm 1 Phase clock.

```
$M \leftarrow \Theta(\log n), S \leftarrow \log(n/p) + \log\log n + O(1)$
$\forall v \in V, v.second \leftarrow 0, v.minute \leftarrow 0, v.hour \leftarrow 0$
foreach interaction $(u, v)$ do
    $u$ makes a fair coin flip
    if Heads then $u.second \leftarrow u.second + 1$
    else $u.second \leftarrow 0$
    if $u.second = S$ then
        $u.minute \leftarrow u.minute + 1$
        $u.second \leftarrow 0$
    if $u.minute = M$ then
        $u.hour \leftarrow u.hour + 1$
        $u.minute \leftarrow 0$
    //One-way epidemic
    if $u.hour < v.hour$ then
        $u.hour \leftarrow v.hour$
        $u.minute \leftarrow 0$
        $u.second \leftarrow 0$
    if $u.hour = v.hour$ and $u.minute < v.minute$ then
        $u.minute \leftarrow v.minute$
        $u.second \leftarrow 0$
```

3.3 Analysis

Lower bounding $L(i)$. Our first goal is to show that $L(i) \geq \Omega(np^{-1}\log^2 n)$. In order to achieve this, it enough to show that $S(i) \geq \Omega(np^{-1}\log^2 n)$. This is due to the fact at as soon as a new maximum value for *hour* appears in the population, it is spread to all agents via the one-way epidemic process within $O(np^{-1}\log n)$ steps. Let us formalize this claim. Assume that $S(i) = R_e(i) - R_e(i-1) \geq \Omega(np^{-1}\log^2 n)$. On the other hand, due to the one-way epidemic it holds that $R_s(i) - R_e(i-1) \leq O(p^{-1}n\log n)$. Combining these two facts we get that:

$$\begin{aligned} L(i) = R_e(i) - R_s(i) &\geq \Omega(np^{-1}\log^2 n) + R_e(i-1) - R_s(i) \\ &\geq \Omega(np^{-1}\log^2 n) - O(p^{-1}n\log n) = \Omega(np^{-1}\log^2 n) \end{aligned}$$

Thus, for the rest of this section we focus on lower bounding $S(i)$.

As we aim to bound $S(i) = R_e(i) - R_e(i-1)$, for every i, let us assume for the rest of the analysis that $R_e(i-1) = 0$. That is we assume that time 0 is when $v.hour = i$ holds for some agent for the first time. Let $m' = \max_{v \in V, v.hour = i} v.minute$ and let T_k be a random variable such that $m' = k$ holds in the T_k-th step for the first time. Note that $T_0 = 0$ holds with probability 1. We prove the following lemma:

▶ **Lemma 4.** *For $c > 2$, it holds that $Pr(T_{k+1} - T_k > cnp^{-1}\log n) > 1/2$ for any $k = 0, 1, ..., M-1$.*

Proof. For m' to increase by 1 starting from time T_k, at least one agent must observe S consecutive heads in its coin flips. Let us consider $cnp^{-1}\log n$ consecutive interactions starting from time 0. Let us denote by x_v the amount of interactions agent v took part in during this time as initiator. Note that $\sum_{v \in V} x_v = cnp^{-1}\log n$. Let us upper bound the probability of agent v seeing S consecutive heads during this time. The probability that agent v sees a sequence of S heads, starting exactly upon its j-th interaction as initiator, and ending upon its $(\min\{j+S, x_v\})$-th interaction as initiator, is upper bounded by 2^{-S}. Note that if $x_v - j < S$ the probability is 0, but the upper bound still holds. Now let us use a union bound over all values of j. This leads to an upper bound of $x_v \cdot 2^{-S}$ for the probability that agent i sees at least S consecutive heads. Recall that $S = \log(n/p) + \log\log n + c$. To finish the proof we apply a union bound over all agents to get an upper bound of

$$\sum_{v \in V} x_v \cdot 2^{-S} = \frac{cnp^{-1}\log n}{2^c np^{-1}\log n} = c \cdot 2^{-c}$$

for the probability of at least one agent seeing S consecutive heads over a period of $cnp^{-1}\log n$ interactions. Finally $Pr(T_{k+1} - T_k > cnp^{-1}\log n) > 1 - c \cdot 2^{-c}$. By setting $c > 2$ we complete the proof. ◀

The above shows that with constant probability the maximum value of *minute* among all agents does not increase too fast. This holds regardless of the value of *hour*. Next, we show that *with high probability*, for every value of $i \geq 0$, the stretch of round i is sufficiently large.

▶ **Lemma 5.** *For every $i \geq 0$, it holds with high probability that $L(i) = \Omega(np^{-1}\log^2 n)$.*

Proof. Fix some $hour = i$, and let X_k be the indicator variable for the event that $T_{k+1} - T_k > cnp^{-1}\log n$. According to Lemma 4, $E[X_k \mid X_0, ..., X_{k-1}] > 1/2$ holds for any $k = 0, 1, ..., M-1$. Let $X = \sum_{k=0}^{M-1} X_k$, and note that $S(i) \geq X \cdot cnp^{-1}\log n$. Ideally we would like to use a Chernoff bound to lower bound X, but unfortunately the $\{X_k\}$ variables are not independent. Thus, we apply Theorem 2 with parameters $q = 1/2, t = M/4$ and get that: $\Pr\left[\sum_{i=0}^{M} X_i < M/4\right] \leq e^{-\Theta(M)}$.

Recall that $M = \Theta(\log n)$, thus by setting M sufficiently large we get that with high probability $S(i) = \Omega(np^{-1}\log^2 n)$. As noted before, this implies that $L(i) = \Omega(np^{-1}\log^2 n)$, which completes the proof. ◀

We note that the lower bound holds even when all interactions are chosen adversarially ($p = 0$). The only reason p appears in the lower bound is due to the definition of S. We continue to prove our upper bound, for which the existence of random interactions is crucial. Specifically, we require the existence of random interactions in order to utilize one-way epidemics.

Upper bounding $S(i)$. In what follows we upper bound $S(i)$ directly. As before, we first consider m', the maximum value of the *minute* variable, and show that it increases sufficiently fast.

▶ **Lemma 6.** *From any configuration where $m' = j < M$ holds, m' increases to $j+1$ within $dnp^{-2}\log n$ steps with a constant probability, for a sufficiently large constant d.*

Proof. Without loss of generality, we assume that every agent satisfies $hour = i$ and $minute = j$ in the configuration because the one way epidemic propagates $\max_{v \in V} v.hour$ and $\max_{v \in V, v.hour=i} v.minute$ to all agents within $O(p^{-1}n\log n)$ steps with high probability.

In our algorithm, we can say that each agent $v \in V$ plays a *lottery game* repeatedly. Agent v starts one round of the game each time it sees a tail. If v sees S consecutive heads before the next tail, v *wins* the game in that round. Otherwise, (i.e., if v sees less than S heads before it sees the next tail), v *loses* the game in that round. When an agent sees the next tail (or wins the round), the next round of the game begins. At each round of the game, v wins the game with probability $2^{-S} = \Omega(p/(n \log n))$ (Recall that $S = \log(np^{-1}) + \log\log n + O(1)$). The goal of our proof is to show that with constant probability, some agent wins the game at least once within $dnp^{-2} \log n$ steps, for a sufficiently large constant d.

For an agent v, we denote by $W_i(v)$ the event that v wins it's i-th game. Note that the $W_i(v)$ events are independent of each other for all values of i and v. This is because for every interaction only the initiator flips a coin, and the coins used for every game don't overlap. Let us also denote by $Y_k(v) = \bigvee_{i=1}^{k} W_i(v)$, the event that agent v wins at least once in its first k games. Let $Y_k = \bigvee_{v \in V} Y_k(v)$, be the event that at least one agent wins at least one of its first k games.

Let d be a sufficiently large constant and let $\tau = (dp^{-1} \log n)/4$. Let us denote by $X(v)$, the event that agent v plays at least τ games in the first $dp^{-2} n \log n = 4np^{-1}\tau$ steps, and let $X = \bigwedge_{v \in V} X(v)$. Finally we are interested in lower bounding the probability of Z, the event that at least one agent wins a game within the first $4np^{-1}\tau$ steps. We note that it holds that $\Pr[Z] \geq \Pr[X \wedge Y_\tau]$. That is, if all agents play at least τ games within the first $4np^{-1}\tau$ steps, and at least one agents wins one of its first τ games then event Z occurs. Applying a union bound, we write $\Pr[Z] = 1 - \Pr[\neg X \vee \neg Y_\tau] \geq 1 - \Pr[\neg X] - \Pr[\neg Y_\tau]$. In order to conclude the proof we wish to show that $\Pr[\neg Y_\tau] < 1/3$ and $\Pr[\neg X] < 1/3$.

To bound $\Pr[\neg X]$ it is sufficient to show that within the first $4np^{-1}\tau$ steps every agent sees at least τ tails with high probability. As every interaction is chosen uniformly at random with probability p, and the initiator / responder order is also random, within the first $4np^{-1}\tau$ steps each agent will observe at least 2τ tails in expectation. Applying a Chernoff bound for a sufficiently large constant d, we get that every agent will observe τ or more tails w.h.p. Thus $\Pr[\neg X] = O(1/n) < 1/3$.

Next, we bound $\Pr[\neg Y_\tau]$. Expanding the expression, we get:

$$\begin{aligned}\Pr[\neg Y_\tau] &= Pr\left[\bigwedge_{v \in V} \neg Y_\tau(v)\right] = Pr\left[\bigwedge_{v \in V} \bigwedge_{i=1}^{\tau} \neg W_i(v)\right] \\ &= (1 - 2^{-S})^{n\tau} \leq e^{-\frac{n\tau}{2^S}} = e^{-\Omega(d)} \leq 1/3,\end{aligned}$$

where in the above we use the independence of the $\{W_i(v)\}$ events, and the fact that d is sufficiently large. This completes the proof. ◀

We are now ready to prove our upper bound.

▶ **Lemma 7.** *For every $i \geq 0$, it holds with high probability that $S(i) = O(np^{-2} \log^2 n)$.*

Proof. Let d be a sufficiently large constant that satisfies the conditions of Lemma 6 and $\tau = dnp^{-2} \log n$. Fix some *hour* $= i$, and let X_j be the indicator variable such that $X_j = 1$ holds if and only if m' increases at least by one from the $(j-1)\tau$ step to the $\tau j - 1$ step. By Lemma 6, it holds that $E[X_i \mid X_0, ..., X_{i-1}] \geq q$ for some constant q. We finish the proof by applying Theorem 2 with parameter $t = M$ and get that: $\Pr\left[\sum_{i=0}^{2q^{-1}M} X_i < M\right] \leq e^{-\Theta(M)}$.

Recall that $M = \Theta(\log n)$, thus setting M sufficiently large, implies that $S(i) \leq 2q^{-1}M \cdot dnp^{-2} \log n = O(np^{-2} \log^2 n)$ with high probability. ◀

Finally, we state our main theorem:

▶ **Theorem 8.** *Algorithm 1 is a $((np^{-1} \log^2 n), (np^{-2} \log^2 n))$-phase clock that uses $O((\log n) \cdot (\log(p^{-1}n)))$ states for Γ_s.*

4 Leader election

We analyze the following leader election protocol, as described in [38] (the module backup)[3]. In the protocol every agent is either a *a leader* or a *follower*. This is represented via a binary *leader* variable. We assume that initially there is at least one leader in the population. Every agent also has a *level* variable initiated to 0 and bounded by the value $\ell_{max} = \Omega(\log n)$. The protocol assumes the existence of a phase clock in the system. For every agent we call the time between two consecutive increases of the *hour* variable of the phase clock an *epoch for that agent* (this is a subjective value per agent, not to be confused with a *round* as was defined in the previous section). For our usage we can bound the range of the *hour* variable by a small constant. This can be easily implemented via a modulo operation (see [38] for a detailed implementation), where the duration of each round still has the same guarantees of Theorem 8. To simplify the pseudo-code we introduce a *tick* variable which is raised for an agent only in the first interaction it takes part in as an initiator once it enters a new epoch.

The algorithm consists of two parts, where the first part guarantees that we quickly converge to a single leader with high probability, while the second part guarantees that the population *always* reaches a state where there exists a single leader. Accordingly, the second part is very slow to converge, but is rarely required.

1. On the first interaction in each epoch, a leader makes a coin flip and increments the *level* variable if it observed heads (up to the limit ℓ_{max}). Thereafter, the maximum level in the population is shared among all the agents via one-way epidemic. A leader becomes a follower when it observes a higher level than it's own.
2. When two leaders interact, one remains a leader and the other one becomes a follower.

The pseudocode for the above is given in Algorithm 2.

Algorithm 2 Leader election.

```
$\ell_{max} \leftarrow \Theta(\log n)$
$\forall v \in V, v.level \leftarrow 0$
foreach interaction $(u, v)$ do
    //One way epidemic
    if $u.level < v.level$ then
        $u.leader \leftarrow false$
        $u.level \leftarrow v.level$
    //$u.tick \leftarrow true$ when $u$ enters a new epoch
    if $u.tick = true$ and $u.leader = true$ then
        $u.tick \leftarrow false$
        $u$ makes a fair coin flip
        if Heads then $u.level \leftarrow \min\{u.level + 1, \ell_{max}\}$

    if $v.leader = true$ and $u.leader = true$ then $u.leader \leftarrow false$
```

In [38] the correctness and running time are analyzed under Γ_u. First let us present the correctness analysis. That is, there is always at least one leader in the population. Roughly speaking, this is because the first part always keeps the leader with the highest level, while the second part only eliminates a leader if it interacts with another leader. So at any point in time when a leader is eliminated, it can "blame" a leader which currently exists.

[3] Essentially the same idea as [38] was previously presented in [2], however they differ in implementation. We follow the implementation of [38].

As the run-time analysis of [38] is for Γ_u it is not immediately clear what are the implications for Γ_s. Luckily, the analysis still goes through as long as we have a phase clock for Γ_s. Let us present a simplified analysis, which is somewhat different than the analysis presented in [38]. We aim to bound the time it takes to reduce the number of leaders to 1. First we note that because the length of a round is $\Omega(p^{-1}n\log^2 n)$ steps, then with high probability every leader has at least one interaction during the first half of the round, also the information of the interaction is guaranteed to spread to the entire population within the round with high probability. This guarantees that for every round, every leader flips a coin, and the maximum level is propagated throughout the population via the one-way epidemic within that round. For the rest of the analysis we assume that indeed every leader has at least one interaction per epoch and that the phase clock and one-way epidemic function correctly. As these events happen with high probability, we can guarantee that they hold throughout the execution of the first $\Theta(np^{-2}\log^3 n)$ steps with high probability via a simple union bound.[4]

Let us denote by L_i the random variable for the number of leaders remaining after round i, where $L_0 > 0$ is the initial number of leaders. Then it holds that $E[L_i] \leq L_{i-1}(1/2 + 2^{-L_{i-1}})$. This is because the distribution of L_i behaves exactly like $B(L_{i-1}, 1/2)$ (number of heads when tossing L_{i-1} fair coins), with the exception that if all coins are tails, we get L_{i-1} leaders remaining instead of 0. Thus, when computing the expectation we must add a $L_{i-1}2^{-L_{i-1}}$ term. Finally, note that $L_{i-1}(1/2 + 2^{-L_{i-1}}) \leq \frac{3}{4}L_{i-1}$ for all $L_{i-1} \geq 2$. Using Markov's inequality, it holds that $Pr[L_i \geq \frac{33}{40}L_{i-1}] = Pr[L_i \geq \frac{3}{4}L_{i-1} \cdot \frac{11}{10}] \leq \frac{10}{11}$.

Let us denote by X_i the indicator random variable for the event that $L_i < \frac{33}{40}L_{i-1}$. Then it holds that $E[X_i \mid X_1, ..., X_{i-1}] > q$ for some constant q. To complete the proof we apply Theorem 2 and get that within $O(\log n)$ epochs we remain with a single leader with high probability. As every epoch requires $O(p^{-2}n\log^2 n)$ steps, we get a unique leader with high probability in $O(p^{-2}n\log^3 n)$ steps. Combining this with the cost of executing the slower second phase of the algorithm up to convergence, we get an expected running time of $O(p^{-2}n\log^3 n)$. This is because the second part requires $O(p^{-1}n^2)$ steps to completes, but is only required if the first part fails, which happens with probability $O(n^{-2})$. Thus, the second part's contribution to the expected stabilization time is $O(1/p)$. We state the following theorem:

▶ **Theorem 9.** *For every $p < 1$, leader election can be solved under Γ_s in $O(p^{-2}n\log^3 n)$ steps with high probability and in expectation using $\Theta((\log^2 n) \cdot (\log(np^{-1})))$ states.*

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 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.

[4] Note that when the execution time goes to infinity, these guarantees (i.e., synchronization via a phase clock) eventually fail. But our protocol has long since converged by this time.

4 D. Angluin, J. Aspnes, M. J Fischer, and H. Jiang. Self-stabilizing population protocols. *ACM Transactions on Autonomous and Adaptive Systems*, 3(4):13, 2008.

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 J. Beauquier, P. Blanchard, and J. Burman. Self-stabilizing leader election in population protocols over arbitrary communication graphs. In *International Conference on Principles of Distributed Systems*, pages 38–52, 2013.

8 Joffroy Beauquier, Peva Blanchard, Janna Burman, and Rachid Guerraoui. The benefits of entropy in population protocols. In *OPODIS*, volume 46 of *LIPIcs*, pages 21:1–21:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

9 Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. Time-space trade-offs in population protocols for the majority problem. *Distributed Computing*, pages 1–21, 2020.

10 Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. Time-space trade-offs in population protocols for the majority problem. *Distributed Computing*, pages 1–21, 2020.

11 Petra Berenbrink, George Giakkoupis, and Peter Kling. Optimal time and space leader election in population protocols. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 119–129, 2020.

12 Andreas Bilke, Colin Cooper, Robert Elsässer, and Tomasz Radzik. Brief announcement: Population protocols for leader election and exact majority with $O(\log^2 n)$ states and $O(\log^2 n)$ convergence time. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing*, pages 451–453, 2017.

13 Janna Burman, David Doty, Thomas Nowak, Eric E Severson, and Chuan Xu. Efficient self-stabilizing leader election in population protocols. *arXiv preprint*, 2019. `arXiv:1907.06068`.

14 S. Cai, T. Izumi, and K. 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.

15 D. Canepa and M. G. Potop-Butucaru. Stabilizing leader election in population protocols, 2007. URL: `http://hal.inria.fr/inria-00166632`.

16 Luca Cardelli and Attila Csikász-Nagy. The cell cycle switch computes approximate majority. *Scientific reports*, 2(1):1–9, 2012.

17 Soumyottam Chatterjee, Gopal Pandurangan, and Nguyen Dinh Pham. Distributed MST: A smoothed analysis. In *ICDCN*, pages 15:1–15:10. ACM, 2020.

18 Ho-Lin Chen, Rachel Cummings, David Doty, and David Soloveichik. Speed faults in computation by chemical reaction networks. *Distributed Comput.*, 30(5):373–390, 2017.

19 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.

20 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.

21 Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from dna. *Nature nanotechnology*, 8(10):755–762, 2013.

22 Michael Dinitz, Jeremy T Fineman, Seth Gilbert, and Calvin Newport. Smoothed analysis of dynamic networks. *Distributed Computing*, 31(4):273–287, 2018.

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

24 Moez Draief and Milan Vojnovic. Convergence speed of binary interval consensus. *SIAM J. Control. Optim.*, 50(3):1087–1109, 2012.

25 M. J. Fischer and H. Jiang. Self-stabilizing leader election in networks of finite-state anonymous agents. In *International Conference on Principles of Distributed Systems*, pages 395–409, 2006. doi:10.1007/11945529_28.

26 Leszek Gąsieniec and Grzegorz Stachowiak. Enhanced phase clocks, population protocols, and fast space optimal leader election. *Journal of the ACM (JACM)*, 68(1):1–21, 2020.

27 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.

28 Uri Meir, Ami Paz, and Gregory Schwartzman. Models of smoothing in dynamic networks. In *DISC*, volume 179 of *LIPIcs*, pages 36:1–36:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

29 Othon Michail, Paul G Spirakis, and Michail Theofilatos. Simple and fast approximate counting and leader election in populations. In *Proceedings of the 20th International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 154–169, 2018.

30 Anisur Rahaman Molla and Disha Shur. Smoothed analysis of leader election in distributed networks. In *SSS*, volume 12514 of *Lecture Notes in Computer Science*, pages 183–198. Springer, 2020.

31 Etienne Perron, Dinkar Vasudevan, and Milan Vojnovic. Using three states for binary consensus on complete graphs. In *INFOCOM*, pages 2527–2535. IEEE, 2009.

32 Ryoya Sadano, Yuichi Sudo, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. A population protocol model with interaction probability considering speeds of agents. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 2113–2122. IEEE, 2019.

33 Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.

34 Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Commun. ACM*, 52(10):76–84, 2009.

35 Yuichi Sudo, Ryota Eguchi, Taisuke Izumi, and Toshimitsu Masuzawa. Time-optimal loosely-stabilizing leader election in population protocols. *arXiv preprint*, 2020. arXiv:2005.09944.

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

37 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.

38 Yuichi Sudo, Fukuhito Ooshita, Taisuke Izumi, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Time-optimal leader election in population protocols. *IEEE Trans. Parallel Distributed Syst.*, 31(11):2620–2632, 2020.

39 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.

40 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*, (Early Access):1–13, 2021. doi:10.1109/TPDS.2021.3076769.

41 Chuan Xu, Joffroy Beauquier, Janna Burman, Shay Kutten, and Thomas Nowak. Data collection in population protocols with non-uniformly random scheduler. *Theor. Comput. Sci.*, 806:516–530, 2020.

42 Daisuke Yokota, Yuichi Sudo, and Toshimitsu Masuzawa. Time-optimal self-stabilizing leader election on rings in population protocols. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 301–316. Springer, 2020.

A Random initiator-responder order

We show how to extend our proof for the case where the initiator-responder order is random, and no random coins are available. First we change our phase clock algorithm to work without coin flips. The pseudo-code is given as Algorithm 3. It is essentially the same algorithm, but now the initiator increases its *second* value, and the responder sets it to 0. Throughout the rest of the analysis we still refer to "coin flips" made by the agents, where we mean that an agent flips heads if it is an initiator, and tails otherwise. The most important difference to keep in mind is that, while the coin flips for each agent are independent of each other, coin flips between *different agents* are no longer independent.

Algorithm 3 Phase clock.

```
$M \leftarrow \Theta(\log n), S \leftarrow \log(n/p) + \log\log n + O(1)$
$\forall v \in V, v.second \leftarrow 0, v.minute \leftarrow 0, v.hour \leftarrow 0$
foreach interaction $(u, v)$ do
    $u.second \leftarrow u.second + 1$
    $v.second \leftarrow 0$
    if $u.second = S$ then $u.minute \leftarrow u.minute + 1$ and $u.second \leftarrow 0$
    if $u.minute = M$ then $u.hour \leftarrow u.hour + 1$ and $u.minute \leftarrow 0$
    //One-way epidemic
    if $u.hour < v.hour$ then $u.hour \leftarrow v.hour$ and $u.minute \leftarrow 0$ and $u.second \leftarrow 0$
    if $u.hour = v.hour$ and $u.minute < v.minute$ then $u.minute \leftarrow v.minute$ and $u.second \leftarrow 0$
```

We now restate the relevant Lemmas. First note that the lower bound on $L(i)$ still holds as we did not assume independence between coin flips made by different agents in the proof of Lemma 5 and Lemma 4. As for the upper bound we did assume independence in Lemma 6, but not in Lemma 7. We state the following alternative for Lemma 6:

▶ **Lemma 10.** *For Algorithm 3, from any configuration where $m' = j < M$ holds, m' increases to $j + 1$ within $O(np^{-2}\log^2 n)$ steps with a constant probability.*

Proof. We maintain the same notations as the proof of Lemma 6, with the exception that we choose $\tau = (Sdp^{-1}\log n)/4$ (larger by an S factor than originally). The proof remains unchanged until we need to bound $Pr[\neg X]$ and $Pr[\neg Y]$. Now there exist dependencies between the coin flips of different agents (but not the coin flips of a single agent).

The proof that $Pr[\neg X]$, remains unchanged. That is, we used a Chernoff bound to state that $Pr[\neg X(v)] < 1/n^2$. Now we have dependencies between coin flips of different agents, however the coin flips of a single agent are still independent. Finally, we note that: $Pr[\neg X] = Pr\left[\bigvee_{v\in V} \neg X(v)\right] \le 1/n$. Where the last transition is due to a union bound.

Next we bound $Pr[\neg Y]$. Again, we expand the expression:

$$\Pr[\neg Y_\tau] = Pr\left[\bigwedge_{v\in V} \neg Y_\tau(v)\right] = Pr\left[\bigwedge_{v\in V}\bigwedge_{i=1}^{\tau} \neg W_i(v)\right] \le Pr\left[\bigwedge_{(v,i)\in U} \neg W_i(v)\right]$$

Our goal is to bound $Pr\left[\bigwedge_{v\in V} \neg Y_\tau(v)\right]$, however there are dependencies between the events. However, it is sufficient if we can find a subset $U \subseteq V \times [\tau]$, such that the events $\{W_i(v)\}_{(v,i)\in U}$ are independent. Note that that every $W_i(v)$ can depend on at most S other events. This is because every round can have length at most S (at which point the round is won). Let us now construct a set U corresponding to *independent* events $\{W_i(u)\}_{(u,i)\in V}$.

This can be constructed greedily, starting with $U=\emptyset, V'=V\times[\tau]$, we add some $(u,i)\in V'$ to U and remove from V' all (v,j) such that event $W_i(u)$ depends on $W_j(v)$. We continue this construction until $V'=\emptyset$. As for every element added to U at most S elements were removed from V', we get that $|U|\geq n\tau/S$. Now we can write:

$$Pr\left[\bigwedge_{(v,i)\in U}\neg W_i(v)\right]\leq(1-2^{-S})^{n\tau/S}$$

$$\leq e^{-\frac{(ndp^{-1}\log n)/4}{2^S}}=e^{-\frac{(ndp^{-1}\log n)/4}{2^c np^{-1}\log n}}=e^{-\Omega(d)}\leq 1/3,$$

which completes the proof. ◀

We now state the main theorem for our phase clock.

▶ **Theorem 11.** *Algorithm 1 is a $(np^{-1}\log^2 n,\ np^{-2}\log^3 n)$-phase clock that uses $O((\log n)\cdot(\log(p^{-1}n)))$ states under Γ_s when the initiator-responder order is random.*

Finally let us restate our leader election algorithm for the random initiator-responder order case (Algorithm 4). As before, we can still see this algorithm as flipping coins, but we lose the independence. A slight detail we must notice is that according to our original definition the tick is raised when the agent enters a new epoch. This is now problematic, as when an agent enters a new epoch it is always an initiator, and thus will always increase its level. To overcome this obstacle we can assume that tick is raised one interaction *after* the *hour* variable was increased. This now gives an equal probability for increasing and not increasing the level variable. Our analysis presented in Section 4 does not require independence between coin flips and it goes through unchanged. Thus, we have the following theorem.

Algorithm 4 Leader election.

```
$\ell_{max} \leftarrow \Theta(\log n)$
$\forall v \in V, v.level \leftarrow 0$
foreach interaction $(u, v)$ do
    if $u.level < v.level$ then $u.leader \leftarrow false$ and $u.level \leftarrow v.level$
    $//u.tick \leftarrow true$ one interaction after $u$ enters a new epoch
    if $u.tick = true$ and $u.leader = true$ then
        $u.tick \leftarrow false$
        $u.level \leftarrow \min\{u.level + 1, \ell_{max}\}$
    if $v.tick = true$ then $u.tick \leftarrow false$
    if $v.leader = true$ and $u.leader = true$ then $v.leader \leftarrow false$
```

▶ **Theorem 12.** *For every $p<1$, Algorithm 4 solves leader election under Γ_s in $O(np^{-2}\log^4 n)$ steps with high probability and in expectation, using $\Theta((\log^2 n)\cdot(\log(n/p)))$ states, when the initiator-responder order is random.*