Dynamic Size Counting in the Population Protocol Model

Counting (on) agents to drive a phase clock

Dominik Kaaser
dominik.kaaser@tuhh.de
TU Hamburg
Hamburg, Germany

Maximilian Lohmann
maximilian.lohmann@tuhh.de
Christian Doppler Laboratory for Blockchain Technologies
for the Internet of Things, TU Hamburg
Hamburg, Germany

ABSTRACT

The population protocol model describes collections of distributed agents that interact in pairs to solve a common task. We consider a *dynamic* variant of this prominent model, where we assume that an adversary may change the population size at an arbitrary point in time. In this model we tackle the problem of *counting the population size:* in the dynamic size counting problem the goal is to design an algorithm that computes an approximation of $\log n$. This estimate can be used to turn static, non-uniform population protocols, i.e., protocols that depend on the population size n, into dynamic and loosely-stabilizing protocols.

Our contributions in this paper are three-fold. Starting from an arbitrary initial configuration, we first prove that the agents converge quickly to a valid configuration where each agent has a constant-factor approximation of $\log n$, and once the agents reach such a valid configuration, they stay in it for a polynomial number of time steps. Second, we show how to use our protocol to define a uniform and loosely-stabilizing phase clock for the population protocol model. Finally, we support our theoretical findings by empirical simulations that show that our protocols work well in practice.

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**; • **Mathematics of computing → Stochastic processes**.

KEYWORDS

Population Protocols, Size Counting, Loose Stabilization, Phase Clocks

ACM Reference Format:
Dominik Kaaser and Maximilian Lohmann. 2024. Dynamic Size Counting in the Population Protocol Model: Counting (on) agents to drive a phase clock. In *ACM Symposium on Principles of Distributed Computing (PODC '24), June 17–21, 2024, Nantes, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3662158.3662825

PODC '24, June 17–21, 2024, Nantes, France

ACM ISBN 979-8-4007-0668-4/24/06.
https://doi.org/10.1145/3662158.3662825

1 INTRODUCTION

Population protocols [4] are a prominent model for computation among simple and anonymous distributed *agents*. Agents have limited memory and are commonly modeled as finite state machines. They interact with each other in pairs where they observe each others' states and update their states according to a common transition function. In each time step a pair of agents is chosen uniformly at random to interact. The computation does not halt but the agents rather converge to a desired configuration or output. Population protocols have a multitude of applications. They are closely related to chemical reaction networks [32] and model interacting particle systems [30], programmable chemical controllers at the level of DNA [18], or biochemical regulatory processes in living cells [17].

There are two prohibitive factors that limit the application of classical population protocols in realistic scenarios: all agents start in a predefined state, and the population size n is fixed over time. At the same time, in their original paper Angluin, Aspnes, Diamadi, Fischer, and Peralta [4] motivate the model with a flock of birds that are equipped with temperature sensors. Clearly, the number of birds in a flock changes over time [19]. Even worse, throughout hunting season there is a looming threat that a poaching adversary selectively targets certain types of birds in the flock. In this paper we therefore design a *uniform dynamic size counting* protocol that can be used to transform modern (non-uniform) protocols into dynamic protocols. In a uniform population protocol the state space and the transition function do not depend on the population size n.

We prove that our protocol is *loosely-stabilizing* according to the definition by Sudo, Nakamura, Yamauchi, Ooshita, Kakugawa, and Masuzawa [34] even if the population size changes over time: when started in any arbitrary state, our protocol quickly reaches a state with correct output and remains in such a state for a polynomial number of interactions. Our protocol improves upon the state space complexity of the best known protocol by Doty and Eftekhari [21] at the expense of a slightly increased running time.

A plenitude of recent works (e.g., [1, 2, 10, 12, 16, 24]) have proposed efficient but non-uniform population protocols where the required number of states slowly grows with the population size, and the transition functions encode a function of n. Unfortunately, in the context of biologically-inspired computing, determining the exact population size or even a polynomial approximation may not be feasible. Size counting protocols [15, 21, 23] address precisely this issue: they provide an approximation of the number of agents which can then be used to execute non-uniform payload protocols.

1.1 Results in a Nutshell

The problem of dynamic size counting is equivalent to the problem of loosely-stabilizing size counting [22]. In this paper we therefore present the first protocol using an asymptotically optimal number of memory bits for the loosely-stabilizing size counting problem. Our size counting protocol doubles as a loosely-stabilizing uniform phase clock that can be used to synchronize populations.

We remark that the clock by Berenbrink, Biermeier, Hahn, and Kaaser [9] is also loosely-stabilizing. However, their clock encodes $\log n$ in its transition function and thus is not uniform. Clearly, once a protocol encodes the population size it cannot be applied in our dynamic setting where the population size may change over time.

Assuming $\log \hat{n}$ to be the largest initial estimate among the population, our protocol converges in $O(\log \hat{n} + \log n)$ parallel time w.h.p. All agents then continue holding correct estimates for polynomial parallel time w.h.p. When the largest value initially stored in any agents' state is s, our protocol requires $O(\log s + \log \log n)$ bits of memory w.h.p. The expression *with high probability (w.h.p.)* denotes a probability of at least $1 - n^{-\Omega(1)}$.

1.2 Background and Related Work

Approximate Counting. Alistarh, Aspnes, Eisenstat, Gelashvili, and Rivest [1] introduce a protocol for approximate size counting using coin flips. Each agent flips a coin until it lands on heads. The resulting number of trials has (almost) a geometric distribution with parameter 1/2. The largest value sampled by one of the n agents is a constant-factor approximation of $\log n$. In the original population protocol model, however, agents do not have access to a source of randomness. Thus, so-called synthetic coins are used [1, 14]. This technique does not require an external source of randomness but instead it extracts randomness from the random scheduler.

To simplify the analysis of size counting protocols using random coins, Doty and Eftekhari [23] assume that agents have access to random bits. They present an approximate size counting protocol that computes an estimate of $\log n \pm 5.7$. This is accomplished by taking the average of $O(\log n)$ maxima of n geometrically distributed random variables each.

Sudo, Ooshita, Izumi, Kakugawa, and Masuzawa [35] define a protocol that provides truly uniform synthetic random coins by splitting the population into two groups, *leaders* and *followers*. Leaders generate random coins by observing whether they are initiator or responder in an interaction. Unfortunately, this approach cannot be used in our dynamic setting since it may occur that all leaders are removed from the population.

Berenbrink, Kaaser, and Radzik [15] present an approximate size counting protocol that computes $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$. Here, the agents elect a leader that generates M tokens which are balanced using a load-balancing algorithm. If some agents do not have a token after balancing, M must have been smaller than n. In this case, M is doubled and the load balancing restarts. When the process terminates, the agents learn $\log n \pm 1$ from the leader. As before, this protocol is also not suitable in the dynamic setting where the single leader agent may be removed from the population.

For further details we refer to the survey of size counting in population protocols by Doty and Eftekhari [21]. There, the authors also introduce a mechanism to compose uniform counting protocols with non-uniform protocols. Nevertheless, their compound protocols are not applicable in the dynamic setting where the population size may change over time.

Dynamic Size Counting. Dynamic population protocols are one of the open problems described in the seminal paper by Angluin, Aspnes, Diamadi, Fischer, and Peralta [4]. In this setting Doty and Eftekhari [22] introduce an *adversary* that is capable of adding and removing agents from the population. All agents are added in some predefined state, but agents can be removed arbitrarily. With regards to counting the authors prove that dynamic size counting is equivalent to loosely-stabilizing size counting, and they present a loosely-stabilizing size counting protocol that solves the dynamic size counting problem.

In their protocol, Doty and Eftekhari [22] use successive random coin flips to approximate $\log n$ w.h.p. However, the naive approach of always spreading the largest estimate breaks as soon as the population shrinks. Instead, they use the robust detection algorithm introduced in [3] to detect when the estimate is inaccurate. To this end, each agent stores a list of length $O(\log n)$ which can be reduced to $O(\log \log n)$ at the expense of a reduced sensitivity to population changes. The idea is to not only use the maximum of n geometrically distributed random variables but to use the detection algorithm from [3] on their first missing value. As a result, every agent uses $O(\log^2 s + \log n \log \log n)$ bits or $O(\log^2 s + (\log \log n)^2)$ bits of memory w.h.p. in the optimized version. Here, s describes the maximum value stored in any variable of the agent's memory. For this protocol Doty and Eftekhari prove loose-stabilization: starting from an arbitrary configuration their protocol converges in $O(\log n + \log \log \hat{n})$ time, and once the protocol has converged all agents hold an estimate of $\Theta(\log n)$ for a polynomial time. Here, $\log \hat{n}$ is the initial maximum estimate among the population.

Phase Clocks. Phase clocks have widespread use in distributed systems. They can be used for mode changes, triggering different behavior, periodic resets to a predefined state, or periodic synchronization of a *time* variable [6]. In population protocols, phase clocks can be categorized into leader driven, junta driven (i.e., by a group of leaders), and leaderless. An agent's *time* can be visualized as a hand on a clock face. All agents' hands should roughly point in the same direction, i.e., lie within a specific interval. The phase clock is divided into hours, each requiring $\Theta(\log n)$ time to pass.

In the context of population protocols, phase clocks are a fundamental building block many efficient population protocols rely on to orchestrate their distributed computation. Intuitively, phase clocks divide time into blocks of $\Theta(n \log n)$ interactions each. These blocks are then sufficiently long such that a simple epidemic spreading process succeeds in transmitting some information to all agents. This allows block-synchronization of the population and, ultimately, efficient algorithms for the population protocol model.

The first phase clocks were introduced and analyzed by Angluin, Aspnes, and Eisenstat [5]. Together with a *junta-election* mechanism [26] they have been employed to solve leader election [27], majority [9, 11], plurality consensus [7, 8], and size counting [15]. Simple phase clocks are implemented by counters modulo some large value m (see, e.g., [2, 5, 9, 11, 20, 27]). Whenever the counter of some agent u crosses zero, the agent receives a *signal* indicating that

a new phase starts. Following the nomenclature by Berenbrink, Biermeier, Hahn, and Kaaser [9], the signals divide time into alternating intervals: so-called *burst-intervals* and so-called *overlap-intervals*. During a burst-interval, every agent gets exactly one signal. An overlap-interval consists of those interactions between two burst-intervals where no agent gets a signal. A burst-interval together with the subsequent overlap-interval forms a *phase*.

Berenbrink, Biermeier, Hahn, and Kaaser [9] present a loosely-stabilizing phase clock protocol that quickly synchronizes arbitrary configurations. This leaderless phase clock requires $O(\log n)$ states and runs forever. After taking at most $O(\log n)$ time to enter a synchronous configuration, the population stays synchronized at any time w.h.p. However, their protocol is non-uniform: it requires an approximation of $\log n$ and is thus unsuitable to generate an approximation of $\log n$ in the dynamic setting. Nevertheless, while their exact approach is not applicable in our setting, their three-phase division of a phase clock inspired our loosely-stabilizing protocol.

Detection. Alistarh, Dudek, Kosowski, Soloveichik, and Uznanski [3] propose a protocol called *detection* that allows all agents to learn whether a so-called *source* agent is present in the population. The idea is to use transitions of the form $(u, v) \to (\min\{u+1, v+1\}, \min\{u+1, v+1\})$ except for source agents which do not change their state but stay at zero. If the agents reach some value in $\Omega(\log n)$, no source is present w.h.p. In [3] the authors prove lower bounds for the minimum value of any agent if no source agent is present. Sudo, Nakamura, Yamauchi, Ooshita, Kakugawa, and Masuzawa [34] prove corresponding upper bounds. Sudo, Eguchi, Izumi, and Masuzawa [33] analyze a related transition function $(u, v) \to (\max\{u-1, v-1\}, \max\{u-1, v-1\})$ for a protocol called Countdown with Higher Value Propagation (CHVP). In this paper, we use the CHVP protocol to synchronize the population.

2 MODEL AND RESULTS

Throughout this paper, we use V to denote the set of all agents and $|V| = n$ to denote the number of agents. The *state space* Q consists of tuples, where each entry is called a *variable*. A *configuration* $C\colon V \to Q$ maps each agent to a state in Q. The *execution* of a protocol $\Xi = \{C_0, C_1, \dots\}$ is a sequence of configurations. The configuration C_0 is called the *initial configuration*. Each configuration C_{i+1} is generated from configuration C_i by selecting two agents uniformly at random and updating their states according to the transition function defined by our algorithm. Thus C_i describes the configuration after i interactions. We use pseudocode to describe how two interacting agents change their variables in an interaction.

A protocol is $(t_c(n), t_h(n))$*-loosely-stabilizing* if, from any configuration, it converges in $t_c(n)$ time to a correct configuration and holds a correct configuration for $t_h(n)$ time w.h.p. Time is measured in parallel time, where one unit of parallel time consists of n interactions. Accordingly, C_n describes the configuration after one parallel time step. For the sake of comparability we use the same metric as in [22] and measure space complexity in bits rather than in the number of states.

Algorithm 1 SimplifiedDynamicSizeCounting(u, v)

```
1 if u.time ≤ 0                                              ▷ wrap-around
2   or (u ∈ I_reset and v ∈ I_exchange)                      ▷ reset → exchange
3   or (u ∉ I_exchange and u.max ≠ v.max)                    ▷ hold → exchange
4 then
5     grv ← Geom(1/2)
6     (u.time, u.max) ← (τ1 · max{u.max, grv}, grv)
7 if u, v ∈ I_exchange and u.max < v.max                     ▷ exchange maximum
8     (u.time, u.max) ← (τ1 · v.max, v.max)
9 u.time ← max{u.time, v.time} − 1                           ▷ update time
```

2.1 Simplified Algorithm

To give an intuitive overview of our algorithm we first present a simplified version (see Algorithm 1). Our protocol uses geometrically distributed random variables (GRVs) to estimate the population size. These variables can be obtained by repeatedly flipping a coin and counting how many flips are needed until it lands on heads. (We later explain how agents could generate such random variables from the inherent randomness of the scheduler.) The underlying idea is that the maximum of n independent random variables with distribution $\mathrm{Geom}(1/2)$ is in $\Theta(\log n)$ w.h.p. Thus, the agents generate a linear number of GRVs and spread their maximum via epidemic spreading. To adapt to changing population sizes, this process is repeated cyclically akin to a countdown timer.

In the simplified case, each agent u stores two variables, `max` and `time`. The variable `max` stores the current maximum value of GRVs that agent u has encountered. The variable `time` tracks when the maximum GRV was last adopted by any agent. It is a simple countdown that starts at a multiple of `max` and uses CHVP to ensure little deviation among the population. Once the `time` reaches zero, it wraps around to a multiple of either the current maximum value `max` or a newly sampled GRV, if the latter happens to be larger. This wrap-around is called a *reset*.

Similarly to [9], the state space for the `time` variable is divided into three phases which we call *exchange*, *hold*, and *reset*. The phases are defined based on an agent's current maximum estimate `max`. To this end we use three constants $\tau_1 > \tau_2 > \tau_3$ and define that an agent u is in the exchange phase if its u.`time` variable is in $(\tau_1 \cdot u.\mathtt{max}, \tau_2 \cdot u.\mathtt{max}]$, in the hold phase if its u.`time` variable is in $(\tau_2 \cdot u.\mathtt{max}, \tau_3 \cdot u.\mathtt{max}]$, and in the reset phase if otherwise its u.`time` variable is in $(\tau_3 \cdot u.\mathtt{max}, 0]$.

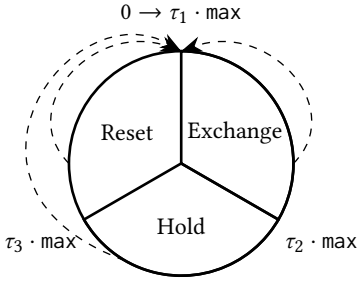

Figure 1: Phase transitions

In the first phase, all agents exchange their maximum value via epidemic spreading. When agents adopt a new larger maximum value, their time is rewound to $\tau_1 \cdot \texttt{max}$. Agents use the detection protocol [3] to synchronize their time by detecting when the maximum was last adopted. This detection protocol is based on epidemic spreading and it concludes before any agent leaves the exchange phase. At this point, all agents have the same maximum value w.h.p.

After the exchange phase, the agents move to the hold phase. If, from this point on, an agent encounters a different maximum value (a low probability event), the premise of the exchange phase has failed. In this case the agent resets: it generates a new GRV and sets its time back to $\tau_1 \cdot \texttt{max}$. Otherwise, if all maximum values align, all agents eventually move to the reset phase. Here, they will immediately reset when interacting with any agent in the exchange phase. Thus, when the first agent leaves the reset phase for the exchange phase, all other agents quickly follow. This effectively starts the next exchange phase via epidemic spreading.

2.2 Our Results

While the algorithm presented in Algorithm 1 aims to convey the idea of our protocol, we do need to modify it to facilitate a theoretical analysis. We will describe the required additions in the next section. For the amended protocol we show the following theorem.

Theorem 2.1. *For any constant $k \geq 2$, and largest initial size estimate $\log \hat{n}$, our algorithm is a $\left(\mathrm{O}(\log \hat{n} + \log n), \Theta(n^{k-1} \log n)\right)$-loosely-stabilizing protocol solving the size counting problem w.h.p. When s denotes the largest value initially stored in any of the agents' variables, our protocol requires $\mathrm{O}(\log s + \log\log n)$ bits per agent w.h.p.*

We remark that the best known size counting protocols compute $\log n \pm 1$ in the static setting [15]. However, non-uniform protocols typically require only an estimate $\log \hat{n} = \Theta(\log n)$ and thus a constant-factor approximation is sufficient for their correctness and performance.

Since our protocol is uniform, it also solves the dynamic size counting problem. When considering the memory in bits, once our protocol is converged it requires an optimal $\mathrm{O}(\log\log n)$ bits per agent, improving upon the $\Omega((\log\log n)^2)$ bits required by [22]. However, this decrease in space complexity comes at the expense of a larger convergence time when compared to [22]. In particular, our convergence time depends on $\log \hat{n} + \log n$ (where $\log \hat{n}$ is the initial estimate) while their protocol converges in roughly $\log\log \hat{n} + \log n$ time. This means their protocol is faster when the initial population size is exponentially over-estimated.

In addition, our protocol doubles as a loosely-stabilizing uniform phase clock. We say that an agent receives a signal whenever the agent resets. Formally, we show the following theorem.

Theorem 2.2. *Let $k \geq 2$ be a constant. Assume that all agents have an estimate of $\Theta(\log n)$ at time t_0. Then there exists a constants c and a sequence of time steps $(t_i)_{i \geq n}$ such that every agent ticks exactly once in the interval $[t_i - c \cdot n \log n, t_i + c \cdot n \log n]$ ("burst") and $t_{i+1} - t_i = \Theta(n \log n)$ with $t_{i+1} - t_i \geq 3c \cdot n \log n$ ("overlap") for a polynomial number of intervals $i \leq n^k$.*

3 DYNAMIC SIZE COUNTING PROTOCOL

In this section we present the details of our protocol for the loosely-stabilizing size counting problem. Our protocol is formally specified in Algorithm 2. Every agent u stores four variables: u.max, u.lastMax, u.time, and u.interactions. For newly added agents these variables are initialized to max = lastMax = 1, time = τ_1, and interactions = 0. The agents use the variable max to spread the maximum of GRVs using epidemics. The time is synchronized via CHVP. In contrast, the interactions variable is not exchanged; it measures the interactions an agent has had since its last reset.

All agents progress through three phases, *exchange*, *hold*, and *reset*, in that order. For each agent u we define these phases by dividing u.time into intervals that depend on the agent's estimate. In a configuration C, agents are in at most one of the following sets corresponding to the respective phases:

$$I_{\text{exchange}} = \{v \in V : C(v).\texttt{time} \geq \tau_2 \cdot C(v).\texttt{max}\},$$
$$I_{\text{hold}} = \{v \in V : \tau_2 \cdot C(v).\texttt{max} > C(v).\texttt{time} \geq \tau_3 \cdot C(v).\texttt{max}\},$$
$$I_{\text{reset}} = \{v \in V : \tau_3 \cdot C(v).\texttt{max} > C(v).\texttt{time} \geq 0\}.$$

If all agents share the same maximum value M defined as $M = \max_{v \in V}(\max\{C(v).\texttt{max}, C(v).\texttt{lastMax})\}$, the length of the exchange phase is $|I_{\text{exchange}}| = (\tau_1 - \tau_2) \cdot M$, the length of the hold phase is $|I_{\text{hold}}| = (\tau_2 - \tau_3) \cdot M$, and the length of the reset phase is $|I_{\text{reset}}| = \tau_3 \cdot M$. Here, the τ_i denote large enough constants which we will specify later in Lemma 4.5.

When agents reset, they generate new GRVs. The goal of the exchange phase is to spread the maximum of these GRVs to all agents via epidemics. The hold phase functions as a separator between the exchange phase and the reset phase; its goal is to ensure that an agent that has just left the exchange phase does not immediately reset. The reset phase is used to launch the agents into the next round while minimizing the spread of the agents' interactions variables.

Lines 2 to 6 of Algorithm 2 implement the phase transition leading to a reset. A reset is formally defined as an interaction where the agent sets its max to another GRV, its time to a multiple of the previous maximum or the new GRV, and its interactions to zero. It follows that resetting agents always enter the exchange phase. During the exchange phase, agents will adopt larger max values without generating new GRVs (see Lines 11,12). In any other cases, resets always lead to new GRVs (see Lines 6,8).

If an agent adopts a new maximum, it resets its time. This is detected by other agents using CHVP (see Line 15). If the adoption happened recently and the agent is still in the exchange phase, it stays in the exchange phase. When agents have not detected a new maximum for a long time, they progress through the hold and reset phases. Eventually, the agents reset back to the exchange phase. At this point, resetting agents replace their max variable with a new GRV.

Most agents' newly sampled GRVs will be much smaller than $\log n$. To keep the population synchronized, the agents store a "trailing" estimate lastMax. If the new GRV is larger, they will use it to define the phase lengths. Otherwise, they will use the last rounds' max value (see Line 6). As we will see, this ensures that the phases of all agents stay sufficiently large.

Algorithm 2 DynamicSizeCounting(u, v)

```
1  function DynamicSizeCounting(u, v)
2    if u.time ≤ 0                                                          ▷ wrap-around
3      or (u ∈ I_reset and v ∈ I_exchange)                                  ▷ reset → exchange
4      or (u ∉ I_exchange and u.max ≠ v.max) then                           ▷ hold → exchange
5        grv ← 20(k + 1) · GRV(k)
6        (u.time, u.interactions, u.max, u.lastMax) ← (τ1 · max{u.max, grv}, 0, grv, u.max)
7    if u.interactions > τ' · max{u.max, u.lastMax}                         ▷ "backup" GRV generation
8        (u.interactions, grv) ← (0, GRV(k))
9        if grv > u.max                                                     ▷ reset if larger than overestimated max
10           (u.time, u.max) ← (τ1 · 20(k + 1) · grv, 20(k + 1) · grv)
11   if u, v ∈ I_exchange and u.max < v.max                                 ▷ exchange maximum
12       (u.time, u.max, u.lastMax) ← (τ1 · v.max, v.max, v.lastMax)
13   if u.max = v.max and (u × v) ∉ (I_exchange × I_reset)                  ▷ exchange last maximum
14       u.lastMax ← max{u.lastMax, v.lastMax}
15   (u.time, u.interactions) ← (max{u.time, v.time} − 1, u.interactions + 1)   ▷ CHVP
```

In CHVP agents always adopt larger `time` values of fellow agents (minus 1). This leads to a problem: One agent adopting a `max` might prevent many others storing the same `max` from resetting, and thus also from generating new GRVs. To prevent this and to guarantee $\Omega(n)$ new GRVs in $O(M + \log n)$ time, we use the `interactions` count to detect such situations. This effectively guarantees that every agent generates one GRV in $O(M)$ time. This is described in Lines 7 to 10 of Algorithm 2. To preserve synchronization, however, agents will only adopt this "backup-GRV" when it is an order of magnitudes larger than their current maximum GRV (see Line 9). To this end agents "overestimate" the saved `max` by $20(k + 1)$ (see Lines 6,10). We use the constant τ' to control the number of interactions any agent should have before generating a "backup-GRV". We define τ' in Lemma 4.5. Finally, Lines 11 to 15 describe the logic for exchanging `max`, `lastMax`, and `time` values.

The intuition for the recovery of our clock from an arbitrary state is the following. As the population starts with an arbitrary maximum $\log \hat{n}$ stored by some agents, the initial maximum might be quite large. Thus, to generate a new accurate maximum, $\log \hat{n}$ is *forgotten* in $O(\log \hat{n})$ time. Agents *forget* a maximum when they overwrite it with a different one. When the entire population forgets a maximum, no agents store it. Initially, a large $\log \hat{n}$ will dominate the epidemic and immediately replace all other estimates. Only after the population forgets $\log \hat{n}$, accurate estimates of $\log n$ can spread to the entire population. Once a new maximum is generated, the agents synchronize and move through the phases together. They then generate new GRVs, leading to a new valid estimate of $\log n$. At this point, the population has converged. This cycle is then repeated again and again, leading to a polynomial holding time w.h.p. Recall that the phases are defined by time metrics that each agent stores, combined with their current estimate. Once converged, the phases thus have length $\Theta(\log n)$.

Geometrically Distributed Random Variables. As in [22] we assume for the sake of our analysis that agents can generate GRVs. However, this is not a strong assumption. Indeed, the process of generating one GRV can be split up into multiple interactions, each consisting of one coin flip. This would allow us, after some warm-up phase, to use synthetic coins as introduced in [1].

To achieve high probability bounds on the maximum of m, Lines 2 to 6 of Algorithm 2 implement the previously defined phase transitions leading to a reset. A reset is defined as the agent setting their `max` to another GRV, their `time` to a multiple of the previous maximum or the new GRV, and their `interactions` to zero. Thus, resetting agents will always enter the exchange phase. Agents already in the exchange phase will adopt larger `max` values without generating new GRVs (see Lines 11, 12). In any other cases, resets always lead to new GRVs (see Lines 6,8). As we will see in Lemma 4.1, k is an arbitrary constant that ultimately allows us to control the error probability of our protocol. For the sake of simplicity, we will assume that each agent can generate these in one interaction instead of k subsequent interactions. As k is constant, this does not affect the asymptotic running time complexity. For completeness, an algorithm that generates the maximum of $k \cdot n$ GRVs is given in the full version [29].

4 ANALYSIS

In this section we prove Theorems 2.1 and 2.2. Note that we did not optimize the constants for our proofs, and some constants that we use in our analysis are quite substantial (see, e.g., Lemma 4.5). However, our empirical analysis in Section 5 shows that the protocol works well with much smaller constants. We start our analysis in Section 4.1 with basic observations and notation after which we present in Section 4.2 fundamental building blocks and tools that we use throughout the remainder of this paper. Then we analyze the convergence time in Section 4.3 and the holding time in Section 4.4.

4.1 Preliminaries

Global Maximum. Agents in the reset phase will reset when interacting with any agent in the exchange phase, no matter what their `max` is. So in the reset phase, `max` values do not impact the protocol. In contrast, agents in the wait phase will only reset when interacting with agents in the exchange phase if their `max` differs.

Thus, at any time we define a *global maximum* M as the largest `max` stored by any agent in the exchange or wait phases. We say a *new global maximum* M' is generated if any agent generates a new GRV such that $M' > M$. A maximum is *forgotten* if it is present only in agents in the reset phase or no agents at all.

Note that the `max` and `lastMax` values may differ. We define all phases using whichever is larger and in the following all formal statements refer to the maximum of `max` and `lastMax`. When using M in the context of time complexity, we mean the largest `max` or `lastMax` value present in agents with this maximum. The expression *relatively quickly* means a time complexity of $O(M)$. As we will see, this becomes $O(\log n)$ after some initial convergence time w.h.p.

Synchronized population. Our protocols resemble a clock divided into three phases. The goal of this clock is to synchronize agents around the clock face. As the clock uses all three variables, our definition of synchronicity also depends on all those variables. We define a *synchronized population* as a population in a configuration where every agent stores the same value

$$\texttt{max}, \texttt{lastMax} \in [0.5 \log n, 40(k+1)^2 \log n].$$

Furthermore, all agents have to be either in $I_{\text{exchange}} \cup I_{\text{wait}}$ or in $I_{\text{wait}} \cup I_{\text{reset}}$, and all agents must have $\texttt{time} < \tau_1 M$.

Lastly, to avoid new "backup" GRVs, the `interactions` count must not be too high. How high exactly depends on how far the population has progressed through the intervals. Before all agents store the same maximum, the infection process is akin to an epidemic. Let T denote the time required for an epidemic to finish w.h.p. according to Lemma 4.2. Afterward, the agents progress through the three phases. Let T' denote the time required until the maximum detection value would drop below zero according to Lemma 4.5. We can bound how much the `interactions` values may differ depending on these two variables: At time $t \in [0, T + T']$, no agent may have $\texttt{interactions} > 2t(1 + \sqrt{k/t}) \log n$. Thus, when an agent generates a sufficiently large new global maximum that infects all agents, the population synchronizes. This follows from the definition of Algorithm 2 and is formalized below.

Round. The transition from the reset phase back to the exchange phase is explicitly excluded from the synchronicity definition. When agents reset, they might generate new GRVs smaller than $0.5 \log n$. However, when the population is synchronized, they will re-synchronize without resetting a second time, in the time required by two epidemics w.h.p. We call this entire process one *round*. A round starts at the first synchronized configuration. It includes the last synchronized configuration and extends until the last interaction before the population synchronizes again. In this unsynchronized period, each agent must reset exactly once w.h.p. Thus, once the population is synchronized, agents will directly continue from one round to the next. This is formalized in Section 4.4 where we analyze the holding time of our dynamic size counting protocol.

4.2 Toolbox

In this section, we provide a collection of properties and tools, most of which are from related works. These properties are then used to define and analyze our protocol rigorously.

Maximum of Geometric Random Variables. Whenever we refer to GRVs we mean random variables with geometric distribution $\text{Geom}(1/2)$. It is folklore that the maximum of n independent and identically distributed GRVs is concentrated around $\Theta(\log n)$, see, e.g., Lemma D.7 in [23]. The following result is a straight-forward extension. We give the proof in the full version [29].

Lemma 4.1. *Let $k \geq 1$ be an arbitrary constant with $k \leq n$ and $n \geq 50$, and let $G = \{G_1, G_2, \ldots, G_{k \cdot n}\}$ be a set of $k \cdot n$ i.i.d. random variables with distribution* $\text{Geom}(1/2)$. *Define $M = \max\{G\}$. Then*

$$Pr[0.5 \log n \leq M \leq 2(k+1) \log n] \geq 1 - O(n^{-k}).$$

Epidemics. In epidemics, agents store a single value and adopt the maximum of any agent's value they encounter. Formally, the transition rule reads $(u, v) \to (\max\{u, v\}, v)$. Assume that initially one agent is in state 1 and $n - 1$ agents are in State 0. It is folklore that within $O(n \log n)$ interactions every agent has state 1 w.h.p., see, e.g., Lemma 2 in [5]. The following variant allows us to control the error probability by providing a high-probability bound in terms of k (see, e.g., Lemma 4 in [9]).

Lemma 4.2. *Let k be any positive constant and let M be defined as $M = \max_{v \in V}\{C_0(v)\}$. After at most $t \leq 4(k+1)n \log n$ interactions every agent is in state M with probability at least $1 - O(n^{-k})$.*

Detection Protocol. Sudo, Eguchi, Izumi, and Masuzawa [33] use the detection protocol from [3] by counting down from a starting value. The agents exchange the maximum of their values and decrement it by one. This happens until their values reach zero. In our protocol we use a one-sided version of this CHVP process. The transition rule is defined as

$$(u, v) \to (\max\{u, v\} - 1, v).$$

The following lemma is based on Lemma 3 from [33], which is in turn based on Lemma 1 from [3]. We give a detailed overview of one-sided CHVP in the full version [29].

Lemma 4.3. *Let $m = \max_{v \in V}\{C_0(v)\}$, Δ be an arbitrary positive integer, and k be a positive constant. Any execution of CHVP enters a configuration C_τ with $\tau < 7n(\Delta + k \log n)$, where $\max_{v \in V}\{C_\tau(v)\} \leq m - \Delta$ holds with probability at least $1 - n^{-k}$.*

In the following lemma we investigate the lower bound of the CHVP variable, which is based on Lemma 4 from [33]. Initially, the minimum value can be arbitrarily small. We now show that after $O(\log n)$ time, this minimum will be close to the initial maximum m. To do this, we model the CHVP process as an epidemic starting from m. All agents starting with m start off as infected, with the rest being uninfected. Once all agents have been infected, their minimum value will be bounded by the number of interactions each agent initiates.

Lemma 4.4. *Let $m = \max_{v \in V}\{C_0(v)\}$, Δ be an arbitrary positive integer, and $k \geq 2$ be a constant. Any execution of CHVP enters a configuration C_τ with $\tau = 7n(\Delta + k \log n)$, where $\min_{v \in V}\{C_\tau(v)\} \geq m - 12(\Delta + k \log n)$ holds with probability at least $1 - n^{-k}$.*

For a configuration C we define $C[V] = \{C(v) \mid v \in V\}$. Then a configuration C of CHVP lies in some interval I if $C[V] \subset I$. With slight abuse of notation, we will omit the $[V]$ and write $C \subset I$ meaning that all values of all agents in C lie in that interval. In

the following lemma, we provide the actual constants for which our analysis holds. Recall that we did not try to optimize these constants but they have been chosen for mere convenience.

Lemma 4.5. *Let $k \geq 2$ be a constant, $\tau_1 = 1140k$, $\tau_2 = 1119k$, $\tau_3 = 454k$, and $\tau' = 4350k$. Then for any $M \geq 0.5 \log n$ and $\max_{v \in V}\{C_0(v)\} = \tau_1 M$, there exist $i_1 < i_2 < i_3 = O(nM)$, such that $C_{i_1} \in (\tau_1 M, \tau_2 M], C_{i_2} \in (\tau_2 M, \tau_3 M], C_{i_3} \in (\tau_3 M, 0]$ before any agent initiated more that $\tau' M$ interactions with probability at least $1 - O(n^{-k})$.*

Proof. We define m such that $M = m \log n$ and $i_1 = 8n(k+1)m \log n$. According to Lemma 4.4, for $\Delta_1 = (8/7(k+1)m - k) \log n$, it holds that

$$\begin{aligned} C_{i_1} &\subset (\tau_1 m \log n, \tau_1 m \log n - 12(\Delta_1 + k \log n)] \\ &\subset (\tau_1 m \log n, (\tau_1 - 21k) m \log n] = (\tau_1 M, \tau_2 M]. \end{aligned}$$

Note that i_1 allows for two epidemics to complete w.h.p. (see Lemma 4.2). All agents have left the first interval after $i_2 = 400nkm \log n \geq i_1 + 7n(21km + k) \log n$ interactions (see Lemma 4.3). Thus, for $\Delta_2 = (400/7m - 1)k \log n$, it holds that

$$\begin{aligned} C_{i_2} &\subset ((\tau_1 - 21k) m \log n, \tau_1 m \log n - 12(\Delta_2 + k \log n)] \\ &\subset ((\tau_1 - 21k) m \log n, (\tau_1 - 686k) m \log n] = (\tau_2 M, \tau_3 M]. \end{aligned}$$

Lastly, all agents will have left this interval after $i_3 = 1065nkm \log n = 665nkm \log n + i_2$ interactions. For the lower bound, we now have a concrete desired value: $\min_{v \in V}\{C_{i_3}\} \geq 0$. During $i_3 - i_2$ interactions the minimum reaches at most $\tau_1 m \log n - 1140km \log n$. Thus, $\tau_1 = 1140k$, $\tau_2 = 1119k$, and $\tau_3 = 454k$. Then

$$C_{i_3} \subset (\tau_1 m \log n - 1140km \log n, 0] = (\tau_3 M, 0].$$

As the probability of failure for both Lemmas 4.3 and 4.4 is n^{-k}, by applying a union bound this holds with probability at least $1 - O(n^{-k})$. Lastly, all agents will reach time zero or reset beforehand after at most $i_3 + 7n(454km + k) \log n \leq 4257km \log n$ interactions w.h.p. (see Lemma 4.3), initiating at most

$$4257km \left(1 + \sqrt{1/(4257m)}\right) \log n \leq 4350km \log n = \tau' M$$

interactions with probability $1 - n^{-k}$ (see Lemma A.1 in the full version [29].) □

If $M \geq 0.5 \log n$ and $\max_{v \in V}\{C_0(v)\} = \tau_1 M$, this lemma implies that there exist τ_1, τ_2, τ_3 such that in $\Theta(M)$ time all agents traverse through intervals $(\tau_1 M, \tau_2 M], (\tau_2 M, \tau_3 M], (\tau_3 M, 0]$ together w.h.p. Additionally, $\tau_1 - \tau_2$ is long enough for two epidemics to complete before any agent exits the first interval. Thus, the `time` propagated via CHVP fulfills the requirements of our clock.

This concludes the presentation of the different tools that we use in our protocols. When applied accordingly, they provide the means to approximate the population size, exchange those estimates, and define synchronized clocks. Using these building blocks, we will now define our loosely-stabilizing, uniform phase clock to solve the dynamic size counting problem.

4.3 Convergence Time

One fundamental observation is that new global maxima lead to a synchronized population. When applying this to synchronized populations, all agents enter the exchange phase together, thereby effectively forgetting any old maxima. Thus, the first resetting agent generates a new global maximum. Similarly, in unsynchronized populations, agents will forget the initial maxima and generate new ones relatively quickly, thus synchronizing the population. This is formalized in the following lemma.

Lemma 4.6. *Let $k \geq 2$ be a constant. When any agent generates a new global maximum $M \geq 0.5 \log n$, it synchronizes the entire population in $O(M + \log n)$ time with probability $1 - O(n^{-k})$.*

Proof. As M is new, the generating agent is in the exchange phase with `time` $\geq \tau_1 M$. The new maximum will spread via epidemic. All infected agents will stay in the exchange phase until the entire population stores the same maximum, as required by the definition of τ_1. Once infected, the agents will stay in the exchange phase, with `max` $= M$ unless a new, even larger maximum is generated. In the worst case, the final infected agent generates a new, larger GRV M'. As this would represent the overestimated maximum of kn GRVs, $M' \in [10(k+1) \log n, 40(k+1)^2 \log n]$ holds w.h.p.

However, the `interactions` values might not be synchronized yet, leading to new "backup" GRVs. Initially, most agents might store `max` $\in O(1)$, leading to $O(n \log n)$ interactions while M still spreads. At most $4(k+1)kn \log n$ GRVs are generated during this time (see Lemma 4.2). Using Lemma 4.1, we can provide the upper bound of their maximum as

$$\begin{aligned} &2(k+1) \log(4(k+1)kn \log n) \\ &\quad = 2(k+1)(2 + \log(k+1) + \log(k) + \log n + \log\log n) \\ &\quad \leq 2(k+1)(2 + 4 \log n) < 10(k+1) \log n \end{aligned}$$

with probability at least $1 - O(n^{-k})$. After that point, each agent will store `max` $\in \Omega(M)$. Thus, by the definition of τ', each agent will generate at most k GRVs before the clock is traversed, unless a new maximum is generated. The maximum of these GRVs is at most $10(k+1) \log n$ with probability at least $1 - O(n^{-k})$. If the new maximum is indeed larger, it would spread to all agents. Since it would have to be larger than $0.5 \log n$ and thus be overestimated to at least $0.5 \cdot 20(k+1) \log n = 10(k+1) \log n$, no subsequent "backup" GRVs would be large enough to replace the new maximum and disrupt the synchronization again. Either way, all agents will store the same `max` value without a larger one being generated during the clock's traversal w.h.p.

As all agents now progress through all phases together (see Lemma 4.5), and have a `max` $\geq 0.5 \log n$, the reset $\rightarrow$ exchange phase transition spreads via epidemic, while no agent leaves the exchange phase. Thus, exactly kn GRVs are generated, with their overestimated maximum being in $[10(k+1) \log n, 40(k+1)^2 \log n]$, and spreading to the entire population. At this point, the `interactions` values will differ by at most $4(k+1) \log n(1 + \sqrt{k/(4(k+1)\log n)})$. Note that `lastMax` can still be $\omega(\log n)$ here. However, as no new "backup" GRVs will be generated during the traversal of the clock, all agents will traverse through it together. After another such round

taking $O(M)$ time, it holds that $\texttt{max}, \texttt{lastMax} \in \Theta(\log n)$. Thus, the clock is synchronous. □

In the following, we will show that agents will always generate a new global maximum in $O(M + \log n)$ time. Together with Lemma 4.6, this proves that agents will synchronize within this time. Starting with an M that is low, the agents will generate a new global maximum in polylogarithmic time.

Lemma 4.7. *Let C_0 be an arbitrary initialization with $M < 0.5 \log n$. Then a new global maximum $M' \geq 0.5 \log n$ will be generated in $O(\log n)$ time w.h.p.*

Proof. Observe that in $c \log n$ time, agents have $\Theta(c \log n)$ interactions w.h.p. (see the full version [29]). Thus, in $O(M) \leq O(\log n)$ time, the "backup" GRV generation is triggered for all agents, resulting in at least kn GRVs, with the maximum being at least $0.5 \log n$. □

In the following lemma, we show that starting from large M, the agents generate a new global maximum relatively quickly.

Lemma 4.8. *Let C_0 be an arbitrary initialization, with $M \geq 0.5 \log n$. Then a new global maximum will be generated in $O(M + \log n)$ time w.h.p.*

The idea is as follows. Agents storing M will infect other agents, thereby triggering an epidemic and almost synchronizing the population, except for their `interactions` values. The agents will then end up in the reset phase together and eventually reset, leading to the population forgetting M. If a new, larger maximum is generated at any time during this process, this will synchronize the population directly.

Proof. For the sake of simplicity, we will assume that at least one agent storing M infects another agent. While this is no prerequisite, M provides an upper bound in the time complexities for all maxima at least $0.5 \log n$.

We require this specific distinction because of one edge case. Assume a subset of agents are initialized with M, such that almost all are in the reset phase. However, the rest are $O(1)$ interactions away from entering it as well. Depending on how close those agents are to the reset phase and how many agents with M are in the reset phase already, the probability of them infecting any agents with $\texttt{max} < M$ diminishes. When no new agent is infected with such a `max`, all agents storing $\texttt{max} \geq 0.5 \log n$ will traverse through all phases in $O(M)$ time. Afterward, by Lemma 4.7, they will synchronize in $O(\log n)$ time.

As we assume one agent will be infected with M and thus have $\texttt{time} = \tau_1 M$, this will trigger an epidemic spreading of M. All agents will be infected in $O(\log n) \leq O(M)$ time (see Lemma 4.2) and then progress through the phases almost synchronously. We say almost because the `interactions` values are not yet synchronized. Lemma 4.6 holds if at any time agents generate GRVs larger than M. Then, the population synchronizes relatively quickly. Otherwise, the population will end up together in the reset phase, as required by the definition of τ_3 (see Lemma 4.5). Once the first agent resets, they will trigger the reset → exchange exchange phase transition epidemic. As all resetting agents will set their `time` to at least $\tau_1 M$, they will not exit this phase before all agents enter it and spread the new maximum. As n resetting agents generate k samples each, the new overestimated maximum of those GRVs is in $[10(k+1)\log n, 40(k+1)^2 \log n]$ w.h.p. (see Lemma 4.1). Thus, Lemma 4.6 holds, proving the clock synchronizes. □

We have shown that the agents will generate a new global maximum starting from any initial maxima. As new global maxima synchronize the population, any arbitrary initialization will become synchronized.

Lemma 4.9. *The population synchronizes in $O(M + \log n)$ time from an arbitrary initialization.*

Proof. Agents generate a new global maximum in $O(M + \log n)$ time. This follows directly from the combination of the time bounds from Lemmas 4.7 and 4.8. Either one of the two cases has to hold for any given M. By Lemma 4.6, such a new global maximum synchronizes the population in $O(M + \log n)$ time. □

4.4 Holding Time

For the holding time, we assume the first synchronized configuration of a round as the initialization. First, we show that starting from this configuration, one round directly continues to the next w.h.p. Then we apply the union bound to determine the holding time w.h.p.

Lemma 4.10. *Once synchronized, the population stays synchronized until the first agent resets w.h.p.*

Proof. As τ_1, τ_2, τ_3 are defined according to Lemma 4.5, the agents will gather in each interval w.h.p. Meanwhile, no agent will have more than $\tau' M$ interactions w.h.p. due to τ' being defined in accordance with Lemma 4.5. Thereby, no new "backup" GRVs will be generated. All agents will keep M until the first agent reaches $\texttt{time} = 0$ and thus resets. □

The following lemma shows that when a population is synchronized, agents will eventually reset exactly once before becoming synchronized again.

Lemma 4.11. *Let C_0 be a synchronized configuration. This synchronization implies a round and will continue directly into another round w.h.p.*

In the proof, we show that eventually no agents will be left in the exchange or hold phases. This means the first resetting agent generates a new global maximum. Then we can reduce this problem to Lemma 4.6, without any "backup" GRVs being generated.

Proof. It follows directly from Lemma 4.10 that all agents stay synchronized until the first one resets. Before the first agent resets, all agents are in the reset phase, as required by τ_3 (see Lemma 4.5). Thus, the first resetting agent will generate a new global maximum among the exchange and hold phases. While this might be smaller than $0.5 \log n$, the agent will still stay in the exchange phase for $\Theta(\log n)$ time due to the trailing estimate. During this time, they will trigger the reset → exchange epidemic while staying in the exchange phase themselves.

After every agent has entered the exchange phase, kn GRVs have been generated. The maximum of those is at least $0.5 \log n$ w.h.p. With this maximum, Lemma 4.6 holds. As all agents stay in the

exchange phase until this second epidemic has finished, no further samples are generated, but all agents will become synchronized again. Thus, the population synchronizes again w.h.p. □

The following lemma puts everything together and shows the holding time for the Adoption Detection protocol.

Lemma 4.12. *Starting from a synchronized configuration, the population will continue directly from round to round for* $\Theta(n^{k-1} \log n)$ *time w.h.p.*

Proof. From Lemma 4.11, it follows that starting from a synchronized configuration, the population continues from one round to the next with probability at least $1 - O(n^{-k})$. By the union bound over n^{k-1} rounds, the probability that all continue successively is at least $1 - O(n^{-1})$. As all agents reset exactly once, any new global maximum is $\Theta(\log n)$ w.h.p. by Lemma 4.1. Together with the requirements for the phase intervals (see Lemma 4.5), it follows that one round requires $\Theta(\log n)$ time w.h.p. Thus, n^{k-1} rounds require $\Theta(n^{k-1} \log n)$ time to complete w.h.p. □

4.5 Proof of the Theorems

In this section, we put everything together and prove our main result, Theorem 2.1. We start with the analysis of the space complexity.

Lemma 4.13. *Let* C_0 *be an arbitrary initialization, with* s *denoting the largest value stored in any of the agents' variables. Then Algorithm 2 requires* $O(\log s + \log \log n)$ *bits per agent w.h.p.*

Proof. All variables of agents store asymptotically equivalent values. Let M denote the maximum value generated by our protocol during any execution. The `max` and `lastMax` variables are thus bound by M. The `time` variable is bounded by $\tau_1 M = O(M)$. Similarly, the `interactions` variable resets back to 0 after $\tau' M = O(M)$.

As we assume a constant k, during each round, the largest maximum will be $O(\log n)$ w.h.p. according to Lemma 4.1. By the union bound, in polynomial time, the largest maximum will still be $O(\log n)$ w.h.p. As all these values can be stored in binary representation, each variable requires $O(\log s + \log \log n)$ bits. Thus, the space requirements are $4\,O(\log s + \log \log n) = O(\log s + \log \log n)$ bits w.h.p. □

Finally, we will now give the proofs of Theorems 2.1 and 2.2.

Proof of Theorem 2.1. The proof of Theorem 2.1 follows from Lemma 4.12 for the holding time, Lemma 4.9 for the convergence time, and Lemma 4.13 for the space complexity. □

Proof of Theorem 2.2. By assumption of the theorem, all agents are in a synchronized configuration at time t_0. From Lemma 4.11 it follows that the agents will perform one synchronized round. Observe that the reset is triggered by an epidemic that concludes in $O(n \log n)$ interactions while the constants that control the length of a round are set to a multiple of that time. The theorem therefore follows immediately from Lemma 4.12. □

5 EMPIRICAL ANALYSIS

In this section, we present empirical data and show that our protocol works well for practical instance sizes. In particular, our data confirm that our protocol requires only modest constant factors. Our protocol is parameterized by τ_1, τ_2, τ_3 to define the phase lengths. In our simulation we set the constants to $\tau_1 = 6$, $\tau_2 = 4$, and $\tau_3 = 2$. Additionally we set $\tau' = 20$ and $k = 16$, resulting in a theoretical holding time of $\Omega(n^{15})$. In our simulations the reported estimate of an agent u is $\max\{u.\texttt{max}, u.\texttt{lastMax}\}$ without the overestimation applied in Algorithm 2. Unfortunately, our protocol has an unbounded state space hence we cannot use ready-made simulators [13, 25]. We therefore implement a custom simulator software using the C++ programming language. As a source of randomness we use the pseudo-random number generator (PRNG) ranlux [31], a high quality PRNG with a period length of over 10^{100} [28]. To ensure independence among our simulations we seed the PRNG with a non-deterministic random number using the C++ random_device generator. Our simulator software is available from our public GitHub repository.[1]

We simulate populations of up to 10^6 agents for 5000 parallel time steps. Each data point is generated from 96 independent simulation runs. To ensure quick simulation times, we do not report the configuration after each interaction but instead we create a snapshot every n interactions. In our first plot in Section 5 we assume that the system is initially empty and show the minimum, median, and maximum values of all 96 estimates over 5000 parallel rounds for $n = 10^6$. Then in Fig. 3 we plot the relative error for varying population sizes $n = 10^1, 10^2, \ldots, 10^6$. Together, the first two plots confirm our theoretical results by showing that our algorithm has a long holding time and the resulting estimate approaches $\log n$ as n grows. Similarly to [22] we also present in Fig. 4 simulation data where an adversary changes the population size during the simulation. To this end, we reduce the population size to 500 after 1350 parallel time. Our data show the rapid adaption of the estimate to the new population size. Note that the estimates of the decimated populations deviate a lot, in particular the maximum values. This conforms to the findings regarding relative deviation from Fig. 3. Nevertheless, the drop is clearly visible in the estimates, particularly for larger values of n.

6 CONCLUSION

We present a new protocol that solves the dynamic size counting problem. In addition, our protocol constitutes a loosely-stabilizing uniform phase clock. The main reason why our protocol can be used to synchronize the agents into phases is its inherent oscillating behavior. Intuitively, our protocol uses similar phase transitions as the loosely-stabilizing phase clock by Berenbrink, Biermeier, Hahn, and Kaaser [9]. This is the main difference to the work by Doty and Eftekhari [22]; their protocol uses the detection protocol by Alistarh, Dudek, Kosowski, Soloveichik, and Uznanski [3] in a continuous fashion. We believe that our clocks are of independent interest: loosely-stabilizing phase clocks have applications as an underlying synchronization mechanism for further loosely-stabilizing

[1] https://github.com/dcmx/DynamicSizeCounting

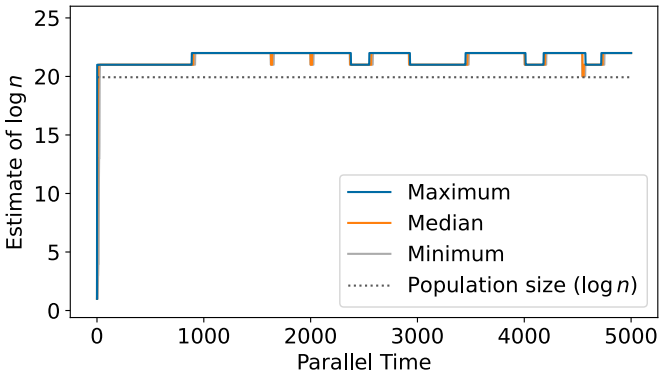

Figure 2: Size estimate in a system of 10^6 agents

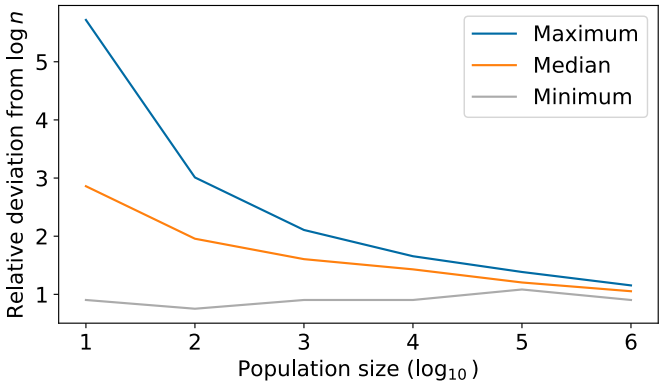

Figure 3: Relative error for different values of n

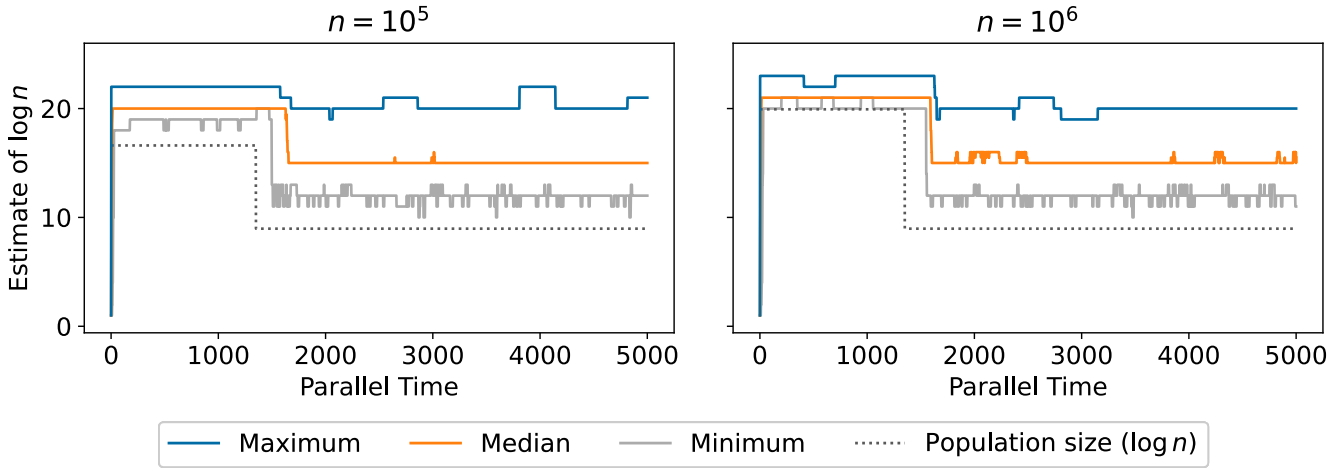

Figure 4: Size estimate for different population sizes. All but 500 agents are removed after 1350 parallel time.

and dynamic population protocols. We highlight that it is an intriguing open question to design a loosely-stabilizing phase clock that uses only $o(\log n)$ (or even only constantly many) states.

Our protocol has a significantly reduced space complexity compared to the best previously known protocol for this problem [22]. This improvement comes at the expense of possibly a larger convergence time. We remark, however, that from an asymptotic point of view this increase in the convergence time only "strikes" if the system is initialized with an estimate $\hat{n}$ that exponentially overestimates the true population size n. If $\hat{n}$ is within polynomial bounds of n, our convergence time remains asymptotically the same. It is an open problem to formally prove a non-trivial trade-off between the required number of states and the convergence time.

Regarding the quality of the approximation, Doty and Eftekhari [23] use in the static setting the average of $O(\log n)$ maxima of n GRVs each. This leads to an additive factor approximation of $\log n$. It is an open question whether a similar extension of our protocol could also provide agents with a more accurate estimate. Doty and Eftekhari [21] furthermore show that non-uniform protocols can be made uniform by composing them with a size counting protocol. They assume a fixed population size and thus use the size counting protocol only once. In dynamic populations, non-uniform protocols must be restarted every time the size changes. A formal analysis of a general framework that allows to compose dynamic size counting protocols with non-uniform protocols in the dynamic setting is an open problem.

ACKNOWLEDGMENTS

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, as well as the Christian Doppler Research Association is gratefully acknowledged.

REFERENCES

[1] D. Alistarh, J. Aspnes, D. Eisenstat, R. Gelashvili, and R. L. Rivest. 2017. Time-space trade-offs in population protocols. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2560–2579. DOI: 10.1137/1.9781611974782.169.

[2] D. Alistarh, J. Aspnes, and R. Gelashvili. 2018. Space-optimal majority in population protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2221–2239. DOI: 10.1137/1.9781611975031.144.

[3] D. Alistarh, B. Dudek, A. Kosowski, D. Soloveichik, and P. Uznanski. 2017. Robust detection in leak-prone population protocols. In *DNA Computing and Molecular Programming - 23rd International Conference, DNA*, 155–171. DOI: 10.1007/978-3-319-66799-7_11.

[4] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. 2006. Computation in networks of passively mobile finite-state sensors. *Distributed Comput.*, 18, 235–253. DOI: 10.1007/s00446-005-0138-3.

[5] D. Angluin, J. Aspnes, and D. Eisenstat. 2008. Fast computation by population protocols with a leader. *Distributed Comput.*, 21, 183–199. DOI: 10.1007/s00446-008-0067-z.

[6] A. Arora, S. Dolev, and M. G. Gouda. 1991. Maintaining digital clocks in step. *Parallel Process. Lett.*, 1, 11–18. DOI: 10.1142/S0129626491000161.

[7] G. Bankhamer, P. Berenbrink, F. Biermeier, R. Elsässer, H. Hosseinpour, D. Kaaser, and P. Kling. 2022. Fast consensus via the unconstrained undecided state dynamics. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA*, 3417–3429. DOI: 10.1137/1.9781611977073.135.

[8] G. Bankhamer, P. Berenbrink, F. Biermeier, R. Elsässer, H. Hosseinpour, D. Kaaser, and P. Kling. 2022. Population protocols for exact plurality consensus: how a small chance of failure helps to eliminate insignificant opinions. In *PODC '22: ACM Symposium on Principles of Distributed Computing*, 224–234. DOI: 10.1145/3519270.3538447.

[9] P. Berenbrink, F. Biermeier, C. Hahn, and D. Kaaser. 2022. Loosely-stabilizing phase clocks and the adaptive majority problem. In *1st Symposium on Algorithmic Foundations of Dynamic Networks, SAND*, 7:1–7:17. DOI: 10.4230/LIPIcs.SAND.2022.7.

[10] P. Berenbrink, R. Elsässer, T. Friedetzky, D. Kaaser, P. Kling, and T. Radzik. 2018. A population protocol for exact majority with o(log5/3 n) stabilization time and theta(log n) states. In *32nd International Symposium on Distributed Computing, DISC*, 10:1–10:18. DOI: 10.4230/LIPICS.DISC.2018.10.

[11] P. Berenbrink, R. Elsässer, T. Friedetzky, D. Kaaser, P. Kling, and T. Radzik. 2021. Time-space trade-offs in population protocols for the majority problem. *Distributed Comput.*, 34, 2, 91–111. DOI: 10.1007/S00446-020-00385-0.

[12] P. Berenbrink, G. Giakkoupis, and P. Kling. 2020. Optimal time and space leader election in population protocols. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, 119–129. DOI: 10.1145/3357713.3384312.

[13] P. Berenbrink, D. Hammer, D. Kaaser, U. Meyer, M. Penschuck, and H. Tran. 2020. Simulating population protocols in sub-constant time per interaction. In *28th Annual European Symposium on Algorithms, ESA*, 16:1–16:22. DOI: 10.4230/LIPIcs.ESA.2020.16.

[14] P. Berenbrink, D. Kaaser, P. Kling, and L. Otterbach. 2018. Simple and efficient leader election. In *1st Symposium on Simplicity in Algorithms, SOSA*, 9:1–9:11. DOI: 10.4230/OASIcs.SOSA.2018.9.

[15] P. Berenbrink, D. Kaaser, and T. Radzik. 2019. On counting the population size. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC*, 43–52. DOI: 10.1145/3293611.3331631.

[16] A. Bilke, C. Cooper, R. Elsässer, and T. Radzik. 2017. 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 ACM Symposium on Principles of Distributed Computing, PODC*, 451–453. DOI: 10.1145/3087801.3087858.

[17] L. Cardelli and A. Csikász-Nagy. 2012. The cell cycle switch computes approximate majority. *Scientific reports*, 2, 656. DOI: 10.1038/srep00656.

[18] Y.-J. Chen, N. Dalchau, N. Srinivas, A. Phillips, L. Cardelli, D. Soloveichik, and G. Seelig. 2013. Programmable chemical controllers made from dna. *Nature nanotechnology*, 8, 10, 755. DOI: 10.1038/nnano.2013.189.

[19] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. 2006. When birds die: making population protocols fault-tolerant. In *Distributed Computing in Sensor Systems, Second IEEE International Conference, DCOSS*, 51–66. DOI: 10.1007/11776178_4.

[20] S. Dolev and J. L. Welch. 2004. Self-stabilizing clock synchronization in the presence of byzantine faults. *J. ACM*, 51, 5, 780–799. DOI: 10.1145/1017460.1017463.

[21] D. Doty and M. Eftekhari. 2021. A survey of size counting in population protocols. *Theor. Comput. Sci.*, 894, 91–102. DOI: 10.1016/j.tcs.2021.08.038.

[22] D. Doty and M. Eftekhari. 2022. Dynamic size counting in population protocols. In *1st Symposium on Algorithmic Foundations of Dynamic Networks, SAND*, 13:1–13:18. DOI: 10.4230/LIPIcs.SAND.2022.13.

[23] D. Doty and M. Eftekhari. 2019. Efficient size estimation and impossibility of termination in uniform dense population protocols. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC*, 34–42. DOI: 10.1145/3293611.3331627.

[24] D. Doty, M. Eftekhari, L. Gasieniec, E. E. Severson, P. Uznanski, and G. Stachowiak. 2021. A time and space optimal stable population protocol solving exact majority. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, 1044–1055. DOI: 10.1109/FOCS52979.2021.00104.

[25] D. Doty and E. E. Severson. 2021. Ppsim: A software package for efficiently simulating and visualizing population protocols. In *Computational Methods in Systems Biology - 19th International Conference, CMSB*, 245–253. DOI: 10.1007/978-3-030-85633-5_16.

[26] L. Gasieniec and G. Stachowiak. 2021. Enhanced phase clocks, population protocols, and fast space optimal leader election. *J. ACM*, 68, 1, 2:1–2:21. DOI: 10.1145/3424659.

[27] L. Gasieniec and G. Stachowiak. 2018. Fast space optimal leader election in population protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2653–2667. DOI: 10.1137/1.9781611975031.169.

[28] F. James and L. Moneta. 2020. Review of high-quality random number generators. *Computing and Software for Big Science*, 4, 2. DOI: 10.1007/s41781-019-0034-3.

[29] D. Kaaser and M. Lohmann. 2024. Dynamic size counting in the population protocol model. Full version of this paper. arXiv: 2405.05137 [cs.DC].

[30] T. M. Liggett. 1985. *Interacting Particle Systems*. DOI: 10.1007/b138374.

[31] M. Lüscher. 1994. A portable high-quality random number generator for lattice field theory simulations. *Computer Physics Communications*, 79, 100–110. DOI: 10.1016/0010-4655(94)90232-1.

[32] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. 2008. Computation with finite stochastic chemical reaction networks. *Nat. Comput.*, 7, 4, 615–633. DOI: 10.1007/S11047-008-9067-Y.

[33] Y. Sudo, R. Eguchi, T. Izumi, and T. Masuzawa. 2021. Time-optimal loosely-stabilizing leader election in population protocols. In *35th International Symposium on Distributed Computing, DISC*, 40:1–40:17. DOI: 10.4230/LIPIcs.DISC.2021.40.

[34] Y. Sudo, J. Nakamura, Y. Yamauchi, F. Ooshita, H. Kakugawa, and T. Masuzawa. 2012. Loosely-stabilizing leader election in a population protocol model. *Theor. Comput. Sci.*, 444, 100–112. DOI: 10.1016/J.TCS.2012.01.007.

[35] Y. Sudo, F. Ooshita, T. Izumi, H. Kakugawa, and T. Masuzawa. 2019. Logarithmic expected-time leader election in population protocol model. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC*, 60–62. DOI: 10.1145/3293611.3331585.