An $O(\log^{3/2} n)$ Parallel Time Population Protocol for Majority with $O(\log n)$ States[*]

Stav Ben-Nun
Bar-Ilan University
Ramat Gan, Israel
stav.bennun@live.biu.ac.il

Tsvi Kopelowitz
Bar-Ilan University
Ramat Gan, Israel
kopelot@gmail.com

Matan Kraus
Bar-Ilan University
Ramat Gan, Israel
matan3@gmail.com

Ely Porat
Bar-Ilan University
Ramat Gan, Israel
porately@cs.biu.ac.il

ABSTRACT

In population protocols, the underlying distributed network consists of n nodes (or agents), denoted by V, and a *scheduler* that continuously selects uniformly random pairs of nodes to *interact*. When two nodes interact, their states are updated by applying a state transition function that depends only on the states of the two nodes prior to the interaction. The efficiency of a population protocol is measured in terms of both time (which is the number of interactions until the nodes collectively have a valid output) and the number of possible states of nodes used by the protocol. By convention, we consider the parallel time cost, which is the time divided by n.

In this paper we consider the *majority* problem, where each node receives as input a color that is either black or white, and the goal is to have all of the nodes output the color that is the majority of the input colors. We design a population protocol that solves the majority problem in $O(\log^{3/2} n)$ parallel time, both with high probability and in expectation, while using $O(\log n)$ states. Our protocol improves on a recent protocol of Berenbrink et al. that runs in $O(\log^{5/3} n)$ parallel time, both with high probability and in expectation, using $O(\log n)$ states.

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**.

KEYWORDS

population protocols, majority

[*]Stav Ben-Nun, Tsvi Kopelowitz, Matan kraus and Ely Porat were supported by ISF grants no. 1278/16 and 1926/19, by a BSF grant 2018364, and by an ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (grant no. 683064).

ACM Reference Format:
Stav Ben-Nun, Tsvi Kopelowitz, Matan Kraus, and Ely Porat. 2020. An $O(\log^{3/2} n)$ Parallel Time Population Protocol for Majority with $O(\log n)$ States. In *Symposium on Principles of Distributed Computing (PODC '20), August 3–7, 2020, Virtual Event, Italy.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3382734.3405747

1 INTRODUCTION

Population protocols, introduced by Angluin et al. [6], have been extensively studied in recent years [1–5, 7, 8, 11–17]. In these protocols, the underlying distributed network consists of n nodes (or agents), denoted by V, and a *scheduler* that continuously selects pairs of nodes to *interact*. Each node u stores its own local state $s_u \in S$, and when two nodes $u, v \in V$ interact, the states of u and v after the interaction are determined solely by the states of u and v prior to the interaction. Following previous work, we assume that any two nodes are allowed to interact, and that each selection of a pair of nodes for interaction is uniformly random from the pairs of nodes in V. The objective of population protocols for a given problem is to extract an output from s_u for each $u \in V$ so that the collective configuration of outputs defines a *valid* solution to the problem, and the valid solution will never change, regardless of future interactions. When the system reaches a valid output configuration that never changes, the system is said to be *stable*. The number of interactions until the system is stable is called the *stabilization* time of the protocol. By convention, we consider the parallel stabilization time, which is the stabilization time divided by n.

The efficiency of a population protocol is measured in terms of both the parallel stabilization time, either with high probability or in expectation, and $|S|$, which directly affects the number of bits that each node must invest in the protocol. Thus, if a protocol runs for $f(n)$ parallel time and has $|S| = g(n)$, then we define the efficiency of the protocol by $<f(n), g(n)>$. Typically, one is interested in protocols where $|S| = O(\text{poly-}\log(n))$, in which case each node stores only $O(\log \log n)$ bits.

Majority. In this paper we consider the *majority* problem in population protocols, where each node u receives an input color from $\{b, w\}$, where b should be interpreted as the color *black* and w should be interpreted as the color *white*. The output of a node is also a color from $\{b, w\}$. An output configuration is valid if all of the nodes output the same color, and more than half of the nodes have this color as their input (the input is assumed to have a well defined majority).

Previous work. Mertzios et al. [17] and Draief and Vojnović [14] designed $<O(n \log n), O(1)>$ protocols for majority. Mertzios et al. [17] named their protocol the *ambassador* protocol, and proved that the parallel stabilization time is $O(n \log n)$. Alistarh et al. [1] recently refined ideas from Alistrah et al. [5], in order to design an $<O(\log^3 n), O(\log^2 n)>$ majority protocol.

Angluin et al. [7] designed a monte carlo majority protocol that relies on the existence of a unique leader in V, but has a low probability of failing. The complexity of their protocol is, w.h.p.[1], $<O(\log^2 n), O(1)>$. Their protocol introduces the idea of *cancellations* and *duplications* or *splitting*. The idea behind cancellations is to introduce an empty color e so that when two nodes with different non-empty colors interact, the nodes change their colors to e, which does not affect the majority, but the ratio between the number of majority colored nodes and the number of minority colored nodes increases. The intuitive idea behind splitting is that when two nodes interact where one has a non-empty color and the other has an empty color, then the color from the colored node is duplicated to also be the color of the non-empty node. Since interactions are chosen uniformly at random, intuitively, the duplications do not change the ratio between the number of majority colored nodes and the number of minority colored nodes, but the difference between these numbers increase over time.

Bilke et al. [12] showed how to implement the protocol of Angluin et al. [7] to work without the existence of a unique leader. The complexity of their protocol is $<O(\log^2 n), O(\log^2 n)>$. Since our protocol uses ideas from the protocol of Bilke et al. [12], we provide details of the protocol in Section 2. Alistarh et al. [2] improved the complexity to $<O(\log^2 n), O(\log n)>$. The state-of-the-art population protocol for the majority problem is the $<O(\log^{5/3} n), O(\log n)>$ protocol of Berenbrink at el. [11], which builds upon the protocol of Bilke et al. [12]; see Section 3.

For lower bounds, Alistarh et al. [2] proved that under the conditions of monotonicity and output dominance, any majority protocol that stabilizes in expected $n^{1-\Omega(1)}$ parallel time must use at least $\Omega(\log n)$ states.

Our results and techniques. In this paper, we describe a new $<O(\log^{3/2} n), O(\log n)>$ protocol for majority, thereby improving the parallel stabilization time of the state-of-the-art protocol of Berenbrink at el. [11]. Notice that, by the lower bound of Alistarh et al. [2], for our runtime, the number of states is optimal.

Our protocol is based on the protocols of Berenbrink at el. [11] and Bilke et al. [12]. In particular, the time cost of the protocol of Berenbrink at el. [11] is $O(\log^{2-a} n + \log^{1+a} n)$, which is minimized when $a = 1/2$. However, in order to guarantee the high probability bounds, Berenbrink at el. [11] require $a \le 1/3$, and so the runtime becomes $O(\log^{2-a} n + \log^{1+a} n) = O(\log^{5/3} n)$.

In order to describe our protocol, we first describe in Section 2 the protocol of Bilke et al [12], which is the basis of the protocol of Berenbrink at el. [11]. Then, in Section 3 we describe the protocol of Berenbrink at el. [11], and in particular we detail the reason for why Berenbrink at el. [11] require $a \le 1/3$. Intuitively, the main challenge is due to maintaining counters in nodes that are incremented based on their interactions, and since the interactions are random, it is likely that values of counters deviate significantly from the average counter value.

Our protocol is designed with the goal of removing the requirement that $a \le 1/3$. In particular, we employ a power-of-two choices strategy ([9]) to reduce the deviation from the average counter value. The main challenge in implementing a power-of-two choices strategy is that we assume that interactions are symmetric, so if two nodes with the same state interact, their states after the interactions are necessarily the same. However, existing power-of-two choices strategies assume that ties can somehow be broken. In order to overcome this challenge, we partition the nodes that are responsible for counters to two equal sized sets, called L and R, and use the `Left` power-of-two choices strategy, which was inspired by Vöcking [18] and analyzed by Berenbrink et al [10], to control the deviation. We remark that the proof of correctness of the protocol of Berenbrink et al [11] becomes significantly simpler when implementing the `Left` power-of-two choices strategy, as we describe in Section 4.

Moreover, in order to guarantee that there are enough nodes that are responsible for the counters (which the algorithm requires in order to progress fast enough), we show in Section 5 that if our strategy for assigning nodes to L and R fails, then running a short version of the protocol of Bilke et al [12] suffices for solving majority.

1.1 Preliminaries

In order to differentiate between the protocol of Bilke et al. [12], the protocol of Berenbrink el al. [11], and our new protocol, while avoiding clutter and redundancy, we use the following names for the three protocols, respectively: the 2-protocol, the 5/3-protocol, and the 3/2-protocol.

Our protocols often uses the *broadcast* protocol, in which a node u broadcasts some information to all of the other nodes in V. The complexity of a broadcast is $<O(\log n), O(1)>$, both w.h.p. and in expectation; see [4, 7, 12].

2 AN $<O(\log^2 n), O(\log^2 n)>$ TIME PROTOCOL

Bilke et al. in [12] introduced an $<O(\log^2 n), O(\log^2 n)>$ majority protocol. For our purposes, we provide an overview of the 2-protocol in [12] while focusing only on the parallel time cost, and ignoring the number of states. Their protocol uses a notion of an *empty* color, which we denote by e, together with a method of *cancellation* and *splitting*. An execution of the protocol has a sequence of at most $\log n + 2$ *phases*, and each phase is composed of four stages: a cancellation stage, a first buffer stage, a splitting stage and second buffer stage. In the 2-protocol, a node $u \in V$ stays in the first three stages of a phase for $\Theta(\log n)$ interactions and in the last stage for $O(\log n)$ (some nodes may be *pulled out* of the last stage to the next phase).

States. Each node u stores the following information:

- The color of u, $c_u \in \{b, w, e\}$.
- A counter α_u, storing the phase number.

[1]In this paper, w.h.p. stands for *with high probability*. An event $\mathcal{E}$ is said to happen w.h.p. if $\Pr[\mathcal{E}] \ge 1 - n^{-\Omega(1)}$.

- A counter β_u, storing the interaction number in the current phase.
- A flag $\texttt{Done}_u$, indicating that u is broadcasting c_u as the majority color.
- A flag $\texttt{Split}_u$, indicating whether or not u has already participated in a split during this phase (the split operation is defined below).
- A flag $\texttt{Fail}_u$, indicating whether or not u has failed.

At the initialization, all of the flags and counters are set to 0. Notice that it is straightforward to derive the stage of u from β_u.

The state transition function. A node u with $\texttt{Fail}_u = 1$ broadcasts $\texttt{Fail}_u$. Otherwise, if $\texttt{Done}_u = 1$ then u broadcasts $\texttt{Done}_u$ and c_u. However, if during the broadcasting of $\texttt{Done}_u$ and c_u, u interacts with a node v where $c_v \neq c_u$, then u sets $\texttt{Fail}_u \leftarrow 1$.

Whenever a node u interacts with node v, if $\texttt{Done}_u = \texttt{Done}_v = 0$, and $\texttt{Fail}_u = \texttt{Fail}_v = 0$, then the following rules define the state transition function (the rules all happen in parallel):

- The counters α_u, β_u are updated as follows: If u and v are not in the same stage and are also not in consecutive stages, then set $\texttt{Fail}_u \leftarrow 1$. Thus, for the rest, assume that u and v are either in the same stage or in consecutive stages. Increment β_u, and if u reached the end of a second buffer stage then set $\beta_u \leftarrow 0$ and increment α_u. If $\alpha_u = \log n + 2$ then set $\texttt{Fail}_u \leftarrow 1$. If u is in a second buffer stage and v is in a cancellation stage, then set $\alpha_u \leftarrow \alpha_v$ (thereby pulling u into the next phase), and $\beta_u \leftarrow 0$. If u enters the second buffer stage:
  - If $\texttt{Split}_u = 0$, then set $\texttt{Done}_u \leftarrow 1$.
  - If $\texttt{Split}_u = 1$, then set $\texttt{Split}_u \leftarrow 0$.
- If u and v are both in the canceling stage, if either $c_u = w$ and $c_v = b$, or $c_u = b$ and $c_v = w$, then set $c_u \leftarrow e$.
- If u and v are both in the splitting stage, $c_u = e$, $c_v \neq e$, and $\texttt{Split}_v = 0$, then perform a *split* operation on v and therefore set $c_u \leftarrow c_v$ and $\texttt{Split}_u \leftarrow 1$.
- If u and v are both in the splitting stage, $c_u \neq e$, $c_v = e$, and $\texttt{Split}_u = 0$, then perform a *split* operation on u by setting $\texttt{Split}_u \leftarrow 1$.

Notice that the protocol uses the $\texttt{Split}$ flags in order to guarantee that each node does not participate in more than 1 split operation per phase.

Analysis. Let w_i, b_i be the number of white nodes and black nodes, respectively, entering phase i and let $d_i = |w_i - b_i|$. Notice that d_i does not change during a cancellation stage.

Inspired by Bilke et al. [12], Berenbrink et al. [11] proved the following.

Lemma 1. *Consider the ith phase of the 2-protocol. Then, w.h.p., if $|w_i - b_i| < n/3$, then after the cancellation stage of the phase, at least $6n/10$ of the nodes have an empty color, and in the splitting stage of the phase, for every node u that enters the splitting stage with a non-empty color, u participates in a split.*

An immediate conclusion from Lemma 1 is that if $|w_i - b_i| < n/3$, then, w.h.p., $d_{i+1} = 2d_i$. Moreover, if for all $i \leq j$, $|w_i - b_i| < n/3$, then, w.h.p., $d_j = 2^j \cdot d_0$, and so after at most $\log(n/d_0) \leq \log n$ phases, the algorithm reaches a phase k where $|w_k - b_k| \geq n/3$. The first such phase is called a *critical* phase. Bilke et al. [12] proved the following lemma regarding critical phases.

Lemma 2. *Once the 2-protocol reaches a critical phase, then, w.h.p., the protocol undergoes at most two more phases until there exists a node u with $c_u \neq e$ that does not split in the splitting stage. Moreover, w.h.p., c_u is the only color left in the system.*

Notice that when there exists a node u with $c_u \neq e$ that does not split in the splitting stage, then $\texttt{Done}_u$ is set to 1 and u broadcasts c_u, which costs another $O(\log n)$ parallel time. Finally, Bilke et al. [12] show that throughout the algorithm, w.h.p., there is never an interaction between two nodes that are at least two stages apart. Thus, the probability that a flag $\texttt{Fail}_u$ for some node u is set to 1 throughout the execution of the protocol is polynomially small in n.

In order to guarantee that the protocol is always correct, the protocol executes the ambassador protocol [17] in the background, which uses $O(1)$ states and $O(n \log n)$ parallel time (or $O(n^2 \log n)$ interactions), both w.h.p. and in expectation. In order to determine the output of a node u, if $\texttt{Fail}_u$ is set to 1, then the output is taken from the output of the ambassador protocol, and otherwise, the output is c_u.

Complexity. Each node participates in $O(\log n)$ interactions per phase, and the number of phases is $O(\log(n/d_0)) = O(\log n)$. Thus, if the algorithm does not set a $\texttt{Fail}$ flag to 1, which happens w.h.p., then the time cost of the algorithm is $O(\log^2 n)$. Moreover, due to the ambassador algorithm running in the background, the algorithm always reaches a valid output configuration, even when some node u sets $\texttt{Fail}_u \leftarrow 1$, and the expected time cost is also $O(\log^2 n)$.

3 AN <$O(\log^{5/3} n), O(\log n)$> TIME PROTOCOL

In the 2-protocol, the $O(\log^2 n)$ time bound is due to each node participating in $O(\log n)$ phases, and having $O(\log n)$ interactions in each phase. Berenbrink el al. [11] designed a protocol that builds upon the 2-protocol, and runs in $O(\log^{5/3} n)$ time. To do so, they reduce the number of interactions in each phase to be $O(\log^{1-a} n)$ for a specific constant $0 < a < 1$, to be chosen later. However, reducing the number of interactions in a phase affects the probability guarantee of Lemma 1. In particular, it is no longer guaranteed that, w.h.p., after a non-critical phase is done, then all of the colored nodes that enter a splitting stage manage to participate in a split. Thus, Berenbrink el al. [11] designed a mechanism that addresses the missing splits: the sequence of phases is partitioned into $\log^{1-a} n$ *epochs* where each epoch is a sequence of $\log^a n$ phases, and at the end of each epoch the protocol enters a special *catch-up phase.* Intuitively, the role of the catch-up phase is to allow for colored nodes that did not split in some phase of the epoch to catch up on their missing splits. Finally, phases no longer have buffers, and the mechanism of *pulling* nodes from one phase to a next is no longer used, and instead, Berenbrink el al. [11] use the same mechanism for pulling nodes from a catch-up phase of one epoch to the next epoch.

A node u is said to be *successful* in a splitting stage if either u enters the stage without a color, or u participates in a split during the splitting stage. Throughout an epoch, a node u is said to be *synchronized* after a splitting stage if so far u was successful in all of the splitting stages of the epoch. If u is not synchronized (that is,

u entered some splitting stage with a color and did not participate in a split), then u is said to be *out-of-sync*. Each node u stores a flag Sync_u, initially set to 1, indicating whether a node is synchronized or not. Throughout the execution of the protocol, if $\mathsf{Sync}_u = 0$, which means that u is out-of-sync, then the protocol ignores the parts of the transition function defined in the 2-protocol during interactions of u with other nodes, except for updating the counters α_u and β_u, and participating in the broadcasts. Each node u also stores a value ϕ_u so that if u is out-of-sync, then ϕ_u is the phase number within the current epoch in which u went out-of-sync. Thus, an out-of-sync node u did not split in $\log^a n - \phi_u$ splitting stages within the current epoch, and so, intuitively, unless this situation is directly addressed, at the end of the epoch there are $val(u) = 2^{\log^a n - \phi_u}$ uncolored nodes that would have been colored with c_u in the 2-protocol. Notice, however, that if the epoch contains a phase that in the 2-protocol would have been a critical phase, then this intuition is not necessarily true, since the 2-protocol in this case may not even reach the end of the epoch.

In order to catch up on coloring the $2^{\log^a n - \phi_u}$ nodes with potentially missing colors, whenever an out-of-sync node u interacts with an uncolored node v and $\phi_u < \log^a n$, then u and v participate in a special split: Let x be ϕ_u prior to the interaction. Then u sets $\phi_u \leftarrow x + 1$, while v sets $c_v \leftarrow c_u$, $\phi_v \leftarrow x + 1$, and $\mathsf{Sync}_v \leftarrow 0$. Notice that, after the split, $val(u)+val(v) = 2\cdot 2^{\log^a n-(x+1)} = 2^{\log^a n - x}$ which is exactly $val(u)$ prior to the interaction. Thus, intuitively, a split operation splits the value of u evenly between u and v. Once u reaches the end of a catch-up phase, if $\phi_u = \log^a n$, then u sets $\mathsf{Sync}_u \leftarrow 1$, and is now synchronized again. The role of the catch-up phase is to guarantee that at the end of the epoch, w.h.p., all of the out-of-sync nodes become synchronized again since they have enough interactions to continue splitting.

Finally, when an out-of-sync node u leaves a catch-up phase of the jth epoch while still being out-of-sync, then the following lemma relates j to the critical phase.

Lemma 3. *[Rephrased from [11]] Suppose that the first time that there exists some node u that finishes an epoch while being out-of-sync happens for epoch j. If $a \le \frac{1}{3}$, then, w.h.p., there exists a positive integer k where $(j-2)\log^a n < k \le j\log^a n$ such that the kth phase is a critical phase.*

Thus, once an out-of-sync node u leaves the jth epoch, node u begins a special broadcast which initiates the 2-protocol, but starting with the colors that were stored at nodes at the beginning of the $(j-1)$th epoch, which is the beginning of phase number $(j-2)\log^a n + 1$. In order for this protocol to run, the nodes store their colors from the beginning of the last two epochs. Since $k - (j-2)\log^a n \le 2\log^a n$, $d_k > n/3$ and, w.h.p., for all $i \le k$ we have $d_{i+1} = 2d_i$, then, w.h.p.,

$$n/3 < d_k = 2^{k-(j-2)\log^a n} d_{(j-2)\log^a n} \le 2^{2\log^a n} d_{(j-2)\log^a n},$$

and so $d_{(j-2)\log^a n} > \frac{n}{3\cdot 2^{2\log^a n}}$. Thus, the 2-protocol starting at phase $(j-2)\log^a n$, runs for an additional $O(\log \frac{n}{d_{(j-2)\log^a n}}) = O(\log^a n)$ phases, with a parallel time of $O(\log^{1+a} n)$. Notice that in order to reduce the number of states, when a node u moves to the 2-protocol, the protocol reuses the bits that were used for the epoch based protocol.

For the total time complexity, there are at most $O(\log n)$ phases until a node u ends an epoch while being out-of-sync, and in each such phase every node has $O(\log^{1-a} n)$ interactions, for a total of $O(\log^{2-a} n)$ time. In addition, there are $\log^{1-a} n$ epochs, and each epoch has a catch-up phase that runs for another $O(\log n)$ interactions, for a total of $O(\log^{2-a} n)$ time. Finally, executing the 2-protocol after some node finishes an epoch while being out-of-sync costs another $O(\log^{1+a} n)$ time. Thus, the total time is $O(\log^{2-a} n + \log^{1+a} n)$. This runtime, is minimized when $a = \frac{1}{2}$. Unfortunately, due to Lemma 3, the protocol requires $a \le \frac{1}{3}$, and so the runtime is minimized when $a = \frac{1}{3}$ and becomes $O(\log^{5/3} n)$.

3.1 Barrier for Increasing a

Our new protocol, described in Section 4, focuses on adapting the algorithm of Berenbrink et al. [11] in a way that allows to increase a to $\frac{1}{2}$. In order to explain the design choices behind our new protocol, we first provide an intuitive explanation as to what are the challenges in making Lemma 3 work for $a > \frac{1}{3}$.

In general, there are two reasons for why there may exist a node that is supposed to participate in a split during a particular phase but that node does not manage to do so. The first reason is that every node has only $O(\log^{1-a} n)$ interactions in a phase (compared to the $O(\log n)$ interactions per phase in the 2-protocol). We call this reason the *short-phase* reason. Berenbrink et al. [11] proved that, w.h.p., although the phases are shorter, a colored node u during a splitting stage has a constant probability of interacting with a non-colored node. Thus, the probability that in a given phase a particular colored node u does not split is at most $\frac{1}{2^{c\log^{1-a} n}}$ for some constant $c > 1$.

The second reason is that some nodes may have participated in a number of interactions that is far from the average number of interactions. In particular, nodes in a splitting stage whose counters are not concentrated near the average counter are likely to have a *small* number of interactions with other nodes that are also in the splitting stage. We call this reason the *deviation* reason. Berenbrink el al. [11] prove that the probability of a counter being at distance at least $\Omega(\log^{1-a} n)$ from the average counter value is at most $\frac{1}{2^{c'\log^{1-2a} n}}$, for some constant $c' > 1$.

Since in the worst-case a node can become out-of-sync during the first phase of an epoch, and there are $\log^a n$ phases in an epoch, an out-of-sync node u in the beginning of the catch-up phase necessitates at most $2^{\log^a n}$ splits in order to guarantee that there are no out-of-sync nodes that originated from u at the end of the catch-up phase. Moreover, since the probability of a node becoming out-of-sync during an epoch is at most $\frac{\log^a n}{2^{c\log^{1-a} n}} + \frac{\log^a n}{2^{c'\log^{1-2a} n}}$, then, w.h.p., by applying a Chernoff bound, the number of out-of-sync nodes at the beginning of a catch-up phase is $\frac{n\log^a n}{2^{\hat{c}\log^{1-a} n}} + \frac{n\log^a n}{2^{\hat{c}'\log^{1-2a} n}}$, for some constants $\hat{c}, \hat{c}' > 1$. For the jth epoch, let U_j denote the total number of splits needed to guarantee that there are no out-of-sync nodes at the end of the catch-up phase of epoch j. Notice that $U_j = \sum_{\text{out-of-sync } u} val(u)$. Thus, w.h.p., $U_j \le \frac{2^{\log^a n} n\log^a n}{2^{\hat{c}\log^{1-a} n}} + \frac{2^{\log^a n} n\log^a n}{2^{\hat{c}'\log^{1-2a} n}}$. Berenbrink el al. [11] proved that, w.h.p., the number of uncolored nodes at the beginning of a catch-up phase is $\Omega(n)$, and so if $U_j = o(n)$ then, w.h.p., the number of splits that take place in order to complete the missing splits until the end of the catch-up phase is U_j.

In order for the first term in the upper bound of U_j to be $o(n)$, we set $a \leq \frac{1}{2}$. However, In order for the second term in the upper bound of U_j to be $o(n)$, we set $a \leq \frac{1}{3}$. Thus, the second term in the upper bound on U_j, which is due to the second reason for having out-of-sync nodes, is the barrier for making the 5/3-protocol to run in $O(\log^{3/2} n)$ time.

4 A <$O(\log^{3/2} n), O(\log^2 n)$> PROTOCOL

Power of two choices. In order to reduce the number of missing splits due to deviations, we change the 5/3-protocol as follows: when two nodes interact, instead of both nodes incrementing their counters, the nodes implement a variation of the *power-of-two choices* strategy which increments the counter only for the node with the smaller counter. However, since we assume that the interactions are symmetric, it is not clear what to do in case of a tie. For now, we assume that the interactions are asymmetric, and so in the case of a tie, the protocol is able to choose just one of the nodes to increment its counter. In Section 5 we explain how to remove the requirement of asymmetric interactions.

The motivation for using a power-of-two choices strategy is that this strategy has the property that counters do not deviate too much from the average counter. This property is summarized in the following invariant and theorem, from Berenbrink et al. [10], which we rephrase here to be expressed in our terms.

Invariant 4. *Let α_i be the fraction of counters whose value is at least i less than the average counter value. If m interactions that increment counters have taken place so far, then, there is a constant c_1 such that for $1 \leq i \leq c_1 \log n$, $\alpha_i \leq 1.3 \cdot 2.8^{-i}$, w.h.p. Moreover, there is a constant c_2 such that for $i \geq c_2 \log n$, $\alpha_i = 0$, w.h.p.*

Theorem 5. *If m interactions that increment counters have taken place so far, then, w.h.p., the value of the maximum counter is $\frac{m}{n} + O(\log \log n)$.*

By Invariant 4, for $i = \hat{c} \log^{1-a} n$ the number of nodes whose counters are at least i away from the average counter is, w.h.p., at most $\frac{n}{2^i} = \frac{n}{2^{\hat{c}\log^{1-a} n}}$. Thus, the probability of a node becoming out-of-sync during an epoch is at most $\frac{\log^a n}{2^{c\log^{1-a} n}} + \frac{\log^a n}{2^{\hat{c}\log^{1-a} n}}$, and so, w.h.p., the number of out-of-sync nodes at the beginning of a catch-up phase is $\frac{n\log^a n}{2^{c''\log^{1-a} n}} + \frac{n\log^a n}{2^{c'''\log^{1-a} n}}$, for some constants $c'', c''' > 1$. Moreover, w.h.p., now $U_j \leq \frac{2^{\log^a n} n \log^a n}{2^{c''\log^{1-a} n}} + \frac{2^{\log^a n} n \log^a n}{2^{c'''\log^{1-a} n}}$, and in order for $U_j = o(n)$ we can choose $a \leq 1/2$.

Finally, the runtime of the algorithm is still $O(\log^{2-a} n + \log^{1+a} n)$, however, now we can choose $a = 1/2$ to get a total runtime of $O(\log^{3/2} n)$.

4.1 Simpler Proofs

As a side effect of using the power-of-two choices strategy, we are able to provide simpler proofs for correctness. The simplifications are a byproduct of the maximum value of a counter being close to the average value.

We define the *beginning* of a stage to be the interaction at which $n - n/2^{\Theta(\log^{1-a} n)}$ nodes have already entered the stage, and the *end* of the stage is when the first node leaves the stage. Similar definitions are made for beginning and end of a phase. Recall that every node has $O(\log^{1-a} n)$ interactions in each stage.

Lemma 6. *At the end of a stage, w.h.p., at least $n - n/2^{\Omega(\log^{1-a} n)}$ of the nodes have had $\Omega(\log^{1-a} n)$ of interactions between the beginning and the end of the stage.*

Proof. When the first node exits the stage, by Theorem 5, w.h.p., the maximum counter value is at most $O(\log \log n)$ away from the average counter value, and, by Invariant 4, w.h.p., there are at most $1.3 \cdot 2.8^{-\Theta(\log^{1-a} n)} n \leq n/2^{\Omega(\log^{1-a} n)}$ nodes whose counter values are at least $\Omega(\log^{1-a} n)$ away from the average counter value. The lemma follows. □

at most $i \cdot \frac{n}{2^{C\sqrt{\log n}}}$ out-of-sync nodes), w.h.p.

The following two lemmas are the basis for the inductive structure of the correctness. Lemma 7 states that, as long as the behaviour so far is as expected, then at the end of a cancellation stage there are many nodes with an empty color. Lemma 8 states that, as long as the behaviour so far is as expected, then the number of nodes that become out-of-sync due to the splitting phase is small. We then conclude, in Corollary 9 and Corollary 10, that, as long as the behaviour so far is as expected, after each phase, the number of out-of-sync nodes is $o(n)$. Thus, the number of out-of-sync nodes entering a catch-up phase is $o(n)$, and in Lemma 11, we use this assumption to show that, unless an epoch has a critical phase, there are no more out-of-sync nodes at the end of the epoch.

Lemma 7. *For a phase i, suppose that just before the phase begins, there are at most $o(n)$ out-of-sync nodes. If i happens before the critical phase, then, w.h.p., after the end of the cancellation stage in phase i at least $6n/10$ nodes have an empty color.*

Proof. By Lemma 6, w.h.p. there are $\Theta(n \log^{1-a} n)$ interactions between the beginning of the cancelling stage and its end. Moreover, there are at least $n - o(n)$ nodes in the cancelling stage. Let W be the set of synchronized nodes that are in the cancelling stage. Thus, $|W| = \Omega(n)$.

Partition the $\Theta(n \log^{1-a} n)$ interactions of the canceling stage into $\Theta(\log^{1-a} n)$ periods, each period consists of n interactions. Let z_i be the number of nodes in W at the beginning of the i'th period that have the minority color with respect to *only* W; notice that it could be that the global minority color is different from the minority color with respect to W. Let i^* denote the index of the last period of the stage. We will prove that $z_{i^*} < n/40$. We say that an interaction has been successful if either there was a cancellation or the number of nodes in W before the interaction that have the minority color with respect to *only* W is (already) at most $41z_i/42$.

Suppose the number of nodes in W before an interaction that have the minority color with respect to *only* W is more than $41z_i/42$. Then the probability that the interaction is a cancellation is at least $2 \cdot (41z_i/42)^2/n^2$, and so if the number of nodes in W at the end of a period that have the minority color with respect to *only* W is more than $n/40$, then the expected number of cancellations within the period is $2 \cdot (41z_i/42)^2/n \geq 2z_i/42$. Let X be the random variable counting the number of cancellations in such a period. By a Chernoff bound,

$$Pr[|X - 2z_i/42| > \frac{1}{2} \cdot 2z_i/42] < \exp\left(\frac{(1/2)^2 \cdot 2z_i/42}{4}\right) = \exp(-z_i/336) < \exp(-n/13440).$$

Therefore, in each such period, w.h.p., at least $z_i/42$ nodes from the minority colors are being canceled, and so $z_{i+1} \leq 41z_i/42$. After a sufficiently large constant number of periods, there are at most $n/40$ nodes from W of the minority color. By the assumption of the lemma, the number of nodes with the majority color is at most $n/3 + n/40$. The difference between the number of nodes with the majority color and the number of nodes with the minority color might be higher because of the out-of-sync nodes, but by no more than $o(n)$ by assumption. Thus, the number of nodes that are still colored after the last period is at most $n/40 + n/3 + n/40 + o(n) < 4n/10$. □

Lemma 8. *Suppose that at the beginning of a phase, there are at least $\frac{6n}{10}$ uncolored nodes, and assume that the phase is before the critical phase. Then, w.h.p., at most $\frac{n}{2^{\Omega(\log^{1-a} n)}}$ nodes become out-of-sync during this phase.*

Proof. Recall that the two reasons for a synchronized colored node u not participating in a split during the splitting stage are the short-phase reason and the deviation reason. Lemma 6 states that at most $n/2^{\Omega(\log^{1-a} n)}$ nodes did not split due to the deviation reason. Here we focus on the short-phase reason.

By the definition of a beginning of a stage, there are at most $o(n)$ nodes that did not enter the stage when the stage begins. Moreover, by assumption, there are at least $\frac{6n}{10}$ nodes with an empty color when the stage begins. Thus, during a stage, the probability for a colored node u to participate in a split during any interaction of u in the stage is at least $2/10 - o(1) \geq 2/11$. For a node u that had at least $\Omega(\log^{1-a} n)$ interactions in the splitting stage, the probability that u does not participate in a split in any of the interactions is at most $2/11^{\Omega(\log^{1-a} n)} = 1/2^{\Omega(\log^{1-a} n)}$. So, the expected number of nodes that did not split is at most $n/2^{\Omega(\log^{1-a} n)}$. By a Chernoff bound, w.h.p., the number of nodes that did not split is at most $n/2^{\Omega(\log^{1-a} n)}$.

□

Corollary 9. *Suppose that before a phase that appears before a critical phase, there are at most $(i-1)\frac{n}{2^{\Omega(\log^{1-a} n)}}$ nodes that are out-of-sync. Then, w.h.p., after the phase, at most $i\frac{n}{2^{\Omega(\log^{1-a} n)}}$ nodes are out-of-sync.*

Corollary 10. *Suppose that epoch j happens before the critical phase. Then $U_j = o(n)$.*

Lemma 11. *Consider an epoch j that appears before a critical phase. If $U_j = o(n)$, then, w.h.p., at the end of the catch-up phase of epoch j all of the nodes are synchronized.*

Proof. Recall that $U_j = \sum_{\text{out-of-sync } u} val(u)$. We partition the missing splits that are counted by U_j to singleton splits. When an interaction takes place between an out-of-sync node u and an non-colored node v, u passes $val(u)/2$ of its singletons to v. Thus, each singleton in u needs to interact with $\log val(u) \leq \log^a n$ non-colored nodes before the end of the catch-up phase in order to guarantee that after the catch-up phase there are no more out-of-sync nodes.

By Lemma 7, there are at most $\frac{4n}{10}$ synchronized colored nodes after the last cancellation stage in the epoch, and each one of these nodes could be split at most once during the last splitting stage. Thus, there are at most $\frac{8n}{10}$ synchronized colored nodes at the beginning of the catch-up phase. In addition, there are at most $o(n)$ out-of-sync nodes at the beginning of the catch-up phase, and so the number of synchronized nodes with an empty color at the beginning of the catch-up phase is more than $\frac{n}{10}$.

By Invariant 4 and Theorem 5, and similar to the proof of Lemma 6, w.h.p., there are $\Omega(n \log n)$ interactions during a catch-up phase, and so, w.h.p., every singleton interacts with at least $\Omega(\log^a n)$ uncolored nodes. Thus, w.h.p., there are no out-of-sync nodes by the end of the catch-up phase. □

Lemma 12. *If an epoch contains a critical phase, then after either this epoch or the next epoch, there is at least one out-of-sync node.*

Proof. Assume by contradiction that the lemma is false. This means that in both epochs all of the colored nodes managed to split. Since there are at least $n/3$ more nodes with the majority color than nodes with the minority color at the beginning of the last phase of the first epoch, then right before the second epoch, the difference is at least $2n/3$. However, this means that after the second epoch there are at least $4n/3$ colored nodes, which is a contradiction. □

5 IMPROVEMENT TO AN <$O(\log^{3/2} n), O(\log n)$> PROTOCOL

In this section, we describe how to reduce the number of states to $O(\log n)$. The main idea, inspired by Angluin et al. [2] and Berenbrink et al [11], is to partition the nodes to two types of nodes: *workers* and *clocks*. Moreover, the protocols no longer use the counters α_u and β_u from the 2-protocol. Instead, we introduce a new counter γ_u that counts until $O(\log n)$. The counter γ_u is used differently, depending on whether u is a clock or a worker.

Intuitively, a worker node u runs the protocol from Section 4, while counting the phase number using γ_u. We also add another $O(1)$ bits per node to indicate the current stage. Notice that the number of the current epoch can be derived from the phase counters.

Clock nodes do not participate in the protocol from Section 4. Instead, they are responsible for the information needed to move a worker node from one phase to the next, and from one stage to the next. Intuitively, clock nodes keep track of progress within an epoch. Each clock node u uses γ_u as a counter that is incremented with applying a variant of the power-of-two choices strategy whenever two clock nodes interact; the details of this strategy are given below. When a clock increments its counter and the counter reaches $\Theta(\log n)$, the clock sets the counter back to 0, indicating that a new epoch has begun.

A worker node u moves from one stage to the next stage when u interacts with a clock whose counter indicates that the phase has progressed to the next stage. Similarly, u moves from one phase to the next phase when u interacts with a clock whose counter indicates that the phase has progressed to the next phase. Notice that the same ideas allow for moving into and out of a catch-up phase.

Left-bias power-of-two choices. The power-of-two choices strategy that we apply is the `Left` variation: the clock nodes are partitioned into two equal-sized sets, the *Left* set and the *Right* set, and counters are incremented only during interactions of two clock nodes from different sets. In such a case, when the counters of the two clock nodes are equal, the clock from the Left set increments its counter. Otherwise, the clock with the smaller counter increments its counter. We show in Lemma 19 that Invariant 4 holds for `Left`. Notice that [10] proved that Theorem 5 holds for `Left`.

Creating workers and clocks. We now describe the procedure for creating workers and clocks. The very first interaction of a node u is called the *initial interaction of* u. When two nodes u and v have an interaction that is an initial interaction for both nodes, where u is a black node and v is a white node, then u becomes a Right-clock and v becomes a Left-clock. such an interaction is called a *first cancellation.* Any other type of initial interaction makes the node a worker node. After the initial interactions take place, it is straightforward to see that the number of Right-clocks is exactly the number of Left-clocks. Moreover, the majority color within the working nodes is the same as the majority color for all of the nodes.

In order to show that our protocol works, w.h.p., we show that the number of clocks and the number of workers is $\Omega(n)$. However, if the number of nodes with the minority color is $o(n)$, and so d_0 (which is the difference between the number of nodes with the majority color and the number of nodes with the minority color) is very large, then it is impossible to guarantee that there are $\Omega(n)$ clocks. However, in such a case, if the workers execute the 2-protocol (Section 2) then by Lemma 2, w.h.p., the workers find the majority color within two phases of the 2-protocol.

Thus, a worker node u right after the initial interaction simulates two phases of the 2-protocol, while using the bits of γ_u to implement α_u and β_u; notice that the bits in γ_u suffice, since there are only $O(\log n)$ interactions during the first two phases of the 2-protocol. During these two phases, interactions between u and clock nodes are ignored. If u does not split during one of the phases, then u initiates a broadcast protocol with the color c_u as the majority. Otherwise, u reverts to the 3/2-protocol, but now using the clock nodes in order to keep track of phases. Notice that when u moves from the two phases of the 2-protocol to the new protocol, the information that was in α_u is overwritten, since now γ_u is used for the new protocol.

At the same time, a clock node v counts until some value $t = \Theta(\log n)$, using the power-of-two choices strategy, where t is chosen to be large enough so that, w.h.p., all of the worker nodes have completed their two phases of the 2-protocol. After the value t is reached, the next increment on the counter in v resets the counter to 0. Notice that a worker node u that finished the two phases of the 2-protocol begins executing the protocol from Section 4 after interacting with a clock node v whose counter has been reset to 0 at least once.

Correctness. If $d_0 > n/3$, then there will be $\Omega(n)$ workers, and so, w.h.p., by Lemma 2, after at most two more phases of the 2-protocol, there exists a node broadcasting the majority color. Thus, for the following assume that $d_0 \le n/3$, and so there are more than $n/3$ nodes of the minority color. The following lemma shows that we have $\Theta(n)$ clocks and $\Theta(n)$ workers.

Lemma 13. *If $d_0 \le n/3$ then, w.h.p., the protocol produces $\Theta(n)$ clocks and $\Theta(n)$ workers.*

Proof. Since $d_0 \le n/3$, there are at least $n/3$ nodes with the minority color. In each of the first $n/12$ interactions, there is a constant probability of at least $1/24$ to have a first cancellation. Thus, there are at least $n/24$ clocks in expectation after $n/12$ interactions, and by a Chernoff bound, w.h.p., at least $n/30$ clocks.

In each of the first $n/12$ interactions, there is a constant probability of at least $1/24$ to *not* have a first cancellation. Similarly, w.h.p., there are at least $n/30$ worker nodes. □

Let n_w be the number of worker nodes and n_c be the number of clock nodes. Next we show that most of the worker nodes are highly synchronized in their progress. Notice that Lemma 6 still holds, as now there are less increments of counters, and so, w.h.p., almost all the nodes have at least $\Omega(\log^{1-a} n)$ interactions between the beginning and the end of each phase. In the following lemma we show that most of the workers have enough interactions *in* every phase.

Lemma 14. *At the end of the ith phase, w.h.p., all but at most $i \cdot n_w/2^{\Theta(\log^{1-a} n)}$ of the worker nodes have $\Theta(\log^{1-a} n)$ interactions in phase i since the beginning of the phase.*

Proof. Let t_s, t_e be the largest counter value of a clock node when phases i and $i+1$ begin, respectfully, and let $t = t_e - t_s = \Theta(\log^{1-a} n)$. We say that a phase *is in time* τ at the first time a clock node in the phase has a counter value of $t_s + \tau$. By Invariant 4 and Theorem 5, when the phase is in time $t/3$, w.h.p., all but at most $n_c/2^{\Omega(\log^{1-a} n)}$ of the clock nodes are in the phase.

Now consider the moment when the phase is in time $2t/3$. Let W_k be the set of nodes that are updated to be in phase k. By induction, prior to the ith phase, there were $|W_{i-1}| = n_w(1 - (i-1) \cdot n_w/2^{\Theta(\log^{1-a} n)})$ worker nodes that were in phase $i-1$. By Lemma 6, all but $n_w/2^{\Omega(\log^{1-a} n)}$ of the worker nodes have $\Theta(\log^{1-a} n)$ interactions between time $t/3$ and $2t/3$. Since there are $\Theta(n)$ clock nodes in the system and w.h.p., at time $t/3$ most of clock nodes are in the phase, then the probability that an interaction between time $t/3$ and time $2t/3$ involving a worker node in phase $i-1$ is with a clock node in phase i is constant. Thus, the probability that a worker node is not in phase i by time $2t/3$ is at most $1/2^{\Omega(\log^{1-a} n)}$. Therefore, the expected number of worker nodes that are not in phase i by time $2t/3$ is at most $|W_{i-1}|/2^{\Omega(\log^{1-a} n)}$, and so by a Chernoff bound, w.h.p., the number of worker nodes that are not in phase i by time $2t/3$ is at most $|W_{i-1}|/2^{\Omega(\log^{1-a} n)}$.

Finally, by Lemma 6, between time $2t/3$ and time t, all but $n_w/2^{\Omega(\log^{1-a} n)}$ of the worker nodes have had $\Theta(\log^{1-a} n)$ interactions.

In conclusion, the number of worker nodes that did not have $\Omega(\log^{1-a} n)$ interactions in phase i is, w.h.p., at most $n_w/2^{\Omega(\log^{1-a} n)}$. □

By Lemma 14 and Lemma 8, in each epoch that is before a critical phase, w.h.p., there are at most $o(n)$ worker nodes that are out-of-sync entering the catch-up phase of the epoch, as desired by Lemma 11. Thus, we conclude the following:

Lemma 15. *After $O(\log^{1.5} n)$ parallel time, w.h.p., the protocol reaches a stable correct configuration.*

In order to achieve the desired running time in expectation, our protocol runs the ambassador protocol in the background. If our protocol succeeds, then the output is determined by our protocol, and otherwise, the output is determined by the ambassador protocol, which is a low probability event. Thus, the expected time is still $O(\log^{1.5} n)$.

Theorem 16. *The exact majority can be computed in $O(\log^{3/2} n)$ parallel time and $O(\log n)$ states per node, w.h.p. and in expectation.*

6 SYMMETRIC INTERACTIONS

In this section we show that Invariant 4 holds for algorithm `Left`. We prove the invariant for a corresponding version of m balls and n bins, following the steps of [10].

Define a *batch* as n consecutive balls, and define *time t* to be the time after t batches. Notice that t is the average number of balls per bin at time t, i.e., after allocating tn balls. The *height* of a ball i in a bin u is the number of balls in u after inserting ball i. Let l_t^u be the number of balls in bin u at time t.

Lemma 17. *Let $a_l \cdot n$ and $b_l \cdot n$ be an upper bound on the number of bins with at most l balls on the left side and the right side, respectively. The probability that a bin u on the* right *side with exactly l balls will receive a ball in the next interaction is at least $\frac{2-2a_l}{n}$. The probability that a bin v on the* left *side with exactly l balls will receive a ball in the next interaction is at least $\frac{2-2b_l}{n}$.*

Proof. Let u be a bin with exactly l balls on the right side. In order for u to receive a ball, u must interact with a bin on the left side that has more than l balls. There are at least $(1-a_l)n$ bins on the left side with more than l balls, and there are $(b_l - b_{l-1})n$ bins on the right side with exactly l balls. The probability that in the next interaction, a bin on the right side with exactly l balls will receive a ball is at least $2(b_l - b_{l-1})(1-a_l)$. Each of the bins with exactly l balls are equally likely to receive a ball, so the probability that a specific bin u receives a ball is at least $2(b_l - b_{l-1})(1-a_l)/(b_l - b_{l-1})n = \frac{2-2a_l}{n}$. Similarly, the probability that a node v of the left side with exactly l balls receives a ball is at least $\frac{2-2b_l}{n}$ (the probability is actually slightly larger). □

Assuming Invariant 4 holds on time $t-1$, and applying Invariant 4 on Lemma 17, we have the following.

Corollary 18. *The probability of a bin u with at most $t-5$ balls after batch $t-1$ to receive a ball in the next allocation is at least $1.9/n$.*

The following lemma corresponds to Invariant 4:

Lemma 19. *Let $t \geq 0$. Assume that there is no batch $\tau < t$ such that Invariant 4 failed in the beginning of batch τ. Then, w.h.p., Invariant 4 holds after batch t.*

Proof. Consider a bin u. If $l_t^u < t$, then we say that u has $t - l_t^u$ *holes*. Let $q_t = t - l_t^u$. Assume $q_t = 5 + i$. Let $t' < t$ be the last batch such that $q_{t'} = 5$. Notice that the number of balls that u receives throughout batches t' to t is at most $t - t' + i$. Moreover, by Lemma 17 and since we assume Invariant 4 holds until time $t-1$, for each ball i in these batches, the probability that i is allocated to u is at least $1.9/n$. The number of balls that are allocated into bin u is asymptotically dominated by a binomial random variable $B((t-t')n, 1.9/n)$. We obtain:

$$\Pr[q_t \geq i+5] \leq \sum_{t'=0}^{t-1} \Pr[B((t-t')n, 1.9/n) \leq t - t' - i]$$
$$\leq \sum_{\tau=1}^{\inf} \sum_{k=i}^{\tau} \Pr[B((t-t')n, 1.9/n) \leq t - t' - i].$$

Berenbrink et al. [10] showed that for every $0 \leq k \leq \tau$,

$$\Pr[B((t-t')n, 1.9/n) \leq t - t' - i] \leq 3.4^{-k} \cdot 3.4^{-0.1\tau}.$$

Hence, $\Pr[q_t \geq i+5] \leq 13.5 \cdot 3.4^{-i}$.

Let Q_t denote the number of holes in the bin with the least number of balls at time t. Thus,

$$\Pr[Q_t \geq i+5] \leq n \cdot 13.5 \cdot 3.4^{-i}.$$

Thus, $Q_t = O(\log n)$ w.h.p., and we have the last argument of Invariant 4. For the first part, we have:

$$\Pr[q_t \geq i] \leq \cdot 13.5 \cdot 3.4^{-i+5} \leq 0.65 \cdot 2.8^{-i},$$

where the second inequality holds for $i > 41$. By the union bound, there are at most $1 + 0.65 \cdot 2.8^{-i}$ bins with at most $t - i$ balls. By Chernoff, w.h.p., there are at most $1.3 \cdot 2.8^{-i} + 2$ bins with at most $t - i$ balls. □

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, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2560–2579, 2017. doi:10.1137/1.9781611974782.169.

[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, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2221–2239, 2018. doi:10.1137/1.9781611975031.144.

[3] Dan Alistarh, Bartlomiej Dudek, Adrian Kosowski, David Soloveichik, and Przemyslaw Uznanski. Robust detection in leak-prone population protocols. In Robert Brijder and Lulu Qian, editors, *DNA Computing and Molecular Programming - 23rd International Conference, DNA 23, Austin, TX, USA, September 24-28, 2017, Proceedings*, volume 10467 of *Lecture Notes in Computer Science*, pages 155–171. Springer, 2017.

[4] Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, pages 479–491, 2015. doi:10.1007/978-3-662-47666-6_38.

[5] Dan Alistarh, Rati Gelashvili, and Milan Vojnovic. Fast and exact majority in population protocols. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 47–56, 2015. doi:10.1145/2767386.2767429.

[6] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 290–299, 2004. doi:10.1145/1011767.1011810.

[7] Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, 2008. doi:10.1007/s00446-008-0067-z.

[8] James Aspnes and Eric Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. 2009. doi:10.1007/978-3-540-89707-1_5.

[9] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999. doi:10.1137/S0097539795288490.

[10] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: The heavily loaded case. *SIAM J. Comput.*, 35(6):1350–1385, 2006. doi:10.1137/S009753970444435X.

[11] Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. 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 2018, New Orleans, LA, USA, October 15-19, 2018*, pages 10:1–10:18, 2018. doi:10.4230/LIPIcs.DISC.2018.10.

[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 ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 451–453, 2017. doi:10.1145/3087801.3087858.

[13] Jurek Czyzowicz, Leszek Gasieniec, Adrian Kosowski, Evangelos Kranakis, Paul G. Spirakis, and Przemyslaw Uznanski. On convergence and threshold properties of discrete lotka-volterra population protocols. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 393–405. Springer, 2015.

[14] Moez Draief and Milan Vojnovic. Convergence speed of binary interval consensus. In *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, pages 1792–1800, 2010. doi:10.1109/INFCOM.2010.5461999.

[15] Leszek Gasieniec, Grzegorz Stachowiak, and Przemyslaw Uznanski. Almost logarithmic-time space optimal leader election in population protocols. In Christian Scheideler and Petra Berenbrink, editors, *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22-24, 2019*, pages 93–102. ACM, 2019.

[16] Adrian Kosowski and Przemyslaw Uznanski. Brief announcement: Population protocols are fast. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 475–477, 2018. URL: https://dl.acm.org/citation.cfm?id=3212788.

[17] George B. Mertzios, Sotiris E. Nikoletseas, Christoforos L. Raptopoulos, and Paul G. Spirakis. Determining majority in networks with local interactions and very small local memory. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 871–882, 2014. doi:10.1007/978-3-662-43948-7_72.

[18] Berthold Vöcking. How asymmetry helps load balancing. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 131–141, 1999. doi:10.1109/SFFCS.1999.814585.