Byzantine-Tolerant Phase Clock

Costas Busch
Department of Computer Science, Augusta University, GA, USA

Paweł Garncarek
Institute of Computer Science, University of Wrocław, Poland

Dariusz R. Kowalski
Department of Computer Science, Augusta University, GA, USA

Abstract

A phase clock is a basic synchronization mechanism that keeps distributed nodes closely synchronized to execute the same phase of a distributed algorithm. A phase clock is typically implemented with a local logical counter that keeps track of the current phase count. Phase clocks are particularly useful in population protocols for implementing leader election and majority selection. We study phase clocks that tolerate Byzantine faults. We show that there is a phase clock that tolerates up to $f < n/3$ faulty nodes, where n is the number of nodes, such that the gap of the local counter values is $O(n^2 \log n)$. The gap can be further lowered to $O(\log n)$ when $f \leq n/8$. We also show that if $f > n/3$, then the gap grows to infinity as time increases. While analyzing phase clock we introduce novel techniques and bounds for balls into bins processes, which might be of independent interest. Using the phase clock, we obtain a majority selection population protocol that tolerates up to f faults and decides on the majority value in $O(\log^2 n)$ parallel time using poly-log states per node.

2012 ACM Subject Classification Computing methodologies → Distributed algorithms; Theory of computation → Adversary models

Keywords and phrases phase clock, Byzantine nodes, population protocols, balls into bins

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2025.30

Funding NSF Grant 2131538

1 Introduction

A phase clock, a basic synchronization mechanism for distributed processes, has been widely studied and implemented in various models. Most of its implementations use logical counters to track the number of steps, and the challenge is to keep them as close as possible across processes. In this work, we address this problem in a stochastic setting, in which interactions that may affect the phase clock occur by a randomized scheduler, e.g., population protocols. We consider phase clocks in distributed systems whose processing nodes could be Byzantine [9, 42, 33, 38, 20].

In order to implement efficiently Fault-Tolerant Phase Clock (FTPC), we study a generalization of the classical Power of Two Choices (PoTC) online principle, in which each ball chooses the bin with a smaller number of balls among two randomly selected, to a fault-prone setting – a faulty bin could present the ball with malicious information to enforce the decision on any of the two bins. This generalization allows using the PoTC paradigm in the design and analysis of distributed protocols, for example, population protocols, multiprocessor job scheduling, and network routing, under the presence of Byzantine failures [41].

1.1 Models and Problems

Population Protocols and the Majority Consensus Problem. In population protocols, a *randomized scheduler* selects random pairs of nodes (out of n nodes) to perform an *exchange* between them uniformly at random [5, 17]. Each exchange alters the states of the two

29th International Conference on Principles of Distributed Systems (OPODIS 2025).
Editors: Andrei Arusoaie, Emanuel Onica, Michael Spear, and Sara Tucci-Piergiovanni; Article No. 30; pp. 30:1–30:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

involved nodes, as specified by the transition function of the population protocol. Such protocols model interactions in natural systems (e.g., biological or chemical), where nodes meet randomly to perform basic tasks [3]. Typically, the nodes are assumed to have a small number of states.

A fundamental task is the *binary majority consensus problem*, where each node has an initial preference value in set $\{A, B\}$ and every node must decide the same value which is the majority in the preference tally. The performance is measured as the *parallel time* steps needed to converge to the decision, where each parallel time step is n exchanges. There is an $\Omega(\log n)$ lower bound to reach consensus [4]. This majority consensus problem has been extensively studied in the literature [2, 4, 17, 36].

Fault tolerance. Recently, population protocols have been studied in the context of a Byzantine adversary with up to a constant fraction $f < n/c_f$ (for some constant c_f) of *faulty nodes* [19]; the other nodes are called *honest nodes*. The Byzantine adversary could cause the minority value to be decided by the honest nodes instead of the majority. More precisely, a faulty node can only affect the state of an honest node during an exchange, by forcing certain transitions to the state machine of the honest node with the hope to derail the majority consensus process. However, the honest nodes are still able to operate after the exchanges with the faulty nodes, and with appropriate algorithm they could even compute a majority value correctly.

In [19] majority consensus population protocols are given that reach the correct decision for the honest nodes in $O(\log^3 n)$ parallel time with $O(\log^3 n)$ states per node. These bounds hold for an absolute difference d between the number of nodes that have initial preferences A and B, such that $d = \Omega(\min\{f + \sqrt{n \log n}, f \log^2 n + 1, n\})$. Here, using a FTPC we improve these bounds by a logarithmic factor for the same d.

Fault-Tolerant Phase Clock. A phase clock is a mechanism used to synchronize distributed processes. In particular, a phase clock in population protocols allows nodes to have an approximate common notion of time [3].

We extend the notion of phase clocks to settings with faults. Consider n nodes and m exchanges. There are h honest nodes and f faulty nodes, $h + f = n$. Each node has a counter which will be used to track approximate time. Whenever an exchange between two nodes occurs, one counter is incremented. If both nodes in the exchange are honest, then the smaller counter of the two is incremented; ties are resolved arbitrarily. Otherwise, the worst-case Byzantine adversary decides which of the two selected nodes increments its counter. The whole process is called a *Fault-Tolerant Phase Clock* (FTPC).

One can note that our FTPC is closely related to balls-into-bins processes, in particular to Power of Two Choices.

Balls-into-Bins and the Power of Two Choices (PoTC). In the classic balls-into-bins problem, m balls are thrown into n bins uniformly at random [31, 37]. Let $l_b(t)$, called a *load*, refer to the number of balls in bin b at time t. When we refer to *time* t, we mean that exactly t balls were thrown into bins so far. For $m = n$ the bin load is $\log n / \log \log n$ balls with high probability $(1 - o(1))$ [28]. In PoTC, in which each ball is proposed with two randomly selected bins and chooses one with smaller load, the maximum load is $\log \log n + \Theta(1)$ with high probability [10, 16]. PoTC is useful in network routing, where packets are sent along the path that currently has the lowest congestion [21]; other applications are in fog computing [15], queuing theory analysis [18], and distributed voting [22].

Table 1 Fault-Tolerant Phase Clock results. (Parameter $c > 0$.)

Theorem	f	m	Gap
1	$> n/3$	any	$\Omega(m/n^2)$
2	$\geq n/3 \cdot (1 + 1/c)$	$\Omega(c^2 n \log n)$	$\Omega(m/(cn))$
3	$< n/3$	any	$O(n^2 \log n)$
4	$\leq n/8$	any	$O(\log n)$

Byzantine faults model many problems in adverse computing environments, where processing nodes and communication links are prone to failures [1, 35, 40, 14, 33]. Recently, PoTC has been examined for $f < n$ faulty bins controlled by a Byzantine adversary [26], where it is shown that for $f < n^2/(2em)$ and $m = O(n \log n)$, the maximum loaded honest bin receives only a logarithmic number of balls. However, a question was left open about the impact of the adversary for larger values of m. The analysis that we present here answers this open question.

As mentioned earlier, FTPC and PoTC are closely related. Bins in PoTC correspond to nodes in FTPC. A ball in PoTC lands in the smaller of two selected bins in the same way that an exchange in FTPC results in incrementing the smaller counter of the two selected nodes, unless faulty bins/nodes are involved. In the remainder of the paper, we will use balls and bins terminology when talking about FTPC to underscore the fact that the results also hold for PoTC with faulty bins.

Adversaries. We consider the following worst-case adversaries. Whenever a ball chooses between an honest or a faulty bin, a worst-case adversary will decide which bin the ball lands into[1]. The adversary knows which two bins are chosen, knows the current loads of all bins and the current time t.

1.2 Our Contributions

For an overview of our Fault-Tolerant Phase Clock results, see Table 1.

We start with two lower bound results. The first lower bound applies to arbitrary number of balls m, while the second result gives better lower bounds for number of balls $m = o(n^2 \log n)$. In particular, the latter result is necessary for fast algorithms such as our Majority Population Protocol, since the number of "balls" during the execution of the algorithm is bounded by $O(n \log^2 n)$.

▶ **Theorem 1** (Phase Clock Gap: Lower Bound 1). *Let n be the number of bins and m denote the number of balls. There exists an adversarial strategy such that for $f > n/3$ faulty bins the expected gap between the most loaded and the least loaded honest bin is at least $\Omega(m/n^2)$.*

▶ **Theorem 2** (Phase Clock Gap: Lower Bound 2). *There exists an adversarial strategy, such that for any $c > 0$, $f \geq n/3 \cdot (1 + 1/c)$, and $m = \Omega(c^2 n \log n)$, with probability at least $1 - O(1/n)$, the gap between the most loaded and the least loaded honest bin is at least $\Omega(m/(cn))$.*

As a consequence of Theorem 2, when c is a large constant then f is arbitrarily close to $n/3$ and the gap is $\Omega(\log n)$ for $m = \Omega(n \log n)$. For $c = \log^{1/2} n$ the gap is $\Omega(\log^{3/2} n)$ for $m = \Omega(n \log^2 n)$, where $f \geq n/3 \cdot (1 + 1/\log^{1/2} n)$. The proof idea is as follows. We split the

[1] Loads of faulty bins never matter, but we say that balls land in faulty bins for consistency.

Table 2 Results and comparison with previous works on majority consensus in population protocols. * The protocol in [7] is designed for a static adversary, while our population protocol is designed and analyzed for the stronger and more general dynamic adversary. In [19] is shown that the lower bound on the tally difference is $d = \Omega(f + 1)$.

Population Protoc.	f	**Tally Difference** d	**Time**	**States**
[25]	0	$\Omega(1)$	$O(\log n)$	$O(\log n)$
[7] *	$o(\sqrt{n})$	$\omega(\sqrt{n}\log n)$	$O(\log n)$	$O(1)$
[19]	$O(n)$	$\Omega(\min\{f + \sqrt{n\log n}, f\log^2 n + 1, n\})$	$O(\log^3 n)$	$O(\log^3 n)$
Theorem 5	$O(n)$	$\Omega(\min\{f + \sqrt{n\log n}, f\log^2 n + 1, n\})$	$O(\log^2 n)$	$O(\log^2 n)$

honest bins into two groups: most loaded bins and least loaded bins. We treat each group as a whole rather than analyzing each bin independently. We estimate the total number of balls that each group is expected to receive and use Chernoff Bounds to obtain a bound on the number of balls in each group with high probability. We calculate the average number of balls per bin in each group. Finally, the gap between these averages will be smaller than the gap between some individual bins.

The dependence of the gap on the number of balls contrasts with the Phase Clock results without faults where it was said that Phase Clock has short memory, i.e., the gap did not depend on the number of balls.

For the FTPC upper bounds, we assume an adversary that decides to put a ball in the faulty bin whenever a faulty bin interacts with one of the k least loaded honest bins. Otherwise, i.e., whenever a faulty bin interacts with one of the $h - k$ most loaded bins, the adversary puts the ball in the honest bin. We call such adversaries *k-adversaries.* The following two results hold for k-adversaries.

▶ **Theorem 3** (Phase Clock Gap: Upper Bound 1). *Consider FTPC process with n bins, including $f < n/3$ faulty bins. At any time t for all pairs of honest bins b_1 and b_2 we have $|l_{b_1}(t) - l_{b_2}(t)| = O(n^2 \log n)$ with probability at least $1 - O(1/n)$.*

▶ **Theorem 4** (Phase Clock Gap: Upper Bound 2). *Consider FTPC process with n bins, including $f \leq n/8$ faulty bins. At any time t for all pairs of honest bins b_1 and b_2 we have $|l_{b_1}(t) - l_{b_2}(t)| = O(\log n)$ with probability at least $1 - O(1/n)$.*

In order to show Theorems 3 and 4, we developed a Time-Dependent Majorization Lemma (Lemma 13), which is a generalization of a known majorization lemma for balls and bins problems. Unlike the original, our version takes into account the fact that the probability vectors may be dynamically changing with time. This technical result may be of independent interest.

It is worth noting that the gap in the lower bound for $f > n/3$ increases with the number of balls m (see Lemma 6), unlike the upper bound for $f < n/3$. This shows a clear separation in the gap around $f = n/3$. (The result below assumes the nodes know a constant factor upper bound approximation of f represented in $O(\log\log n)$ bits.)

▶ **Theorem 5** (Population Protocol Majority Upper Bound). *Let d denote the absolute difference between the number of nodes that have initial preferences A and B. For sufficiently large n, and $f \leq n/c_f$, for some constant c_f, if $d = \Omega(\min\{f + \sqrt{n\log n}, f\log^2 n + 1, n\})$ then there is an algorithm in which all honest nodes decide the majority value in $O(\log^2 n)$ parallel time steps using $O(\log^2 n)$ states per node with probability at least $1 - O(\log^2 n)/n$.*

Table 2 compares our contribution in Theorem 5 with existing work on population protocols. As we mentioned earlier, Theorem 5 improves the result in [19] by a logarithmic factor in time and number of states. In [7] a Byzantine-resilient 3-state population protocol is presented that tolerates up to $f = o(\sqrt{n})$ faulty nodes, and requires $d = \omega(\sqrt{n}\log n)$ and converges in $O(\log n)$ parallel time to the majority value. The protocol in [7] was analyzed against a static adversary, where the faulty nodes are chosen before the execution, while ours is for the stronger and more general dynamic adversary (similar to [19]) where up to f nodes get corrupt during execution.

If no faults can be tolerated, the best-known protocol for time performance converges to the majority decision in $O(\log n)$ time using $O(\log n)$ states per node [25]. However, that work is not Byzantine-resilient like ours. Our analysis is based on the approach in [19], which was further based on the techniques in [17] (which assumed no faults). In [19] it is also shown that the initial difference must be $d \geq 2f$ (for $f > 0$) to guarantee a majority decision (under a weaker adversary).

1.3 Related Work

Phase Clocks and Population Protocols. Phase clocks have been used is self stabilizing systems in [30], in the context of transient faults. Such faults are not Byzantine in nature, and a phase clock deals with stabilizing the system in $O(k)$ time after a fault corrupts at most k processes. Phase clocks have been studied in the broader context of clock synchronization [8, 24, 23]. In population protocols, phase clocks have been used with the help of leaders who coordinate the actions of nodes in the phases [6]. Phase clocks were also used in the context of leader election problems in population protocols [27]. However, it is not straightforward to implement leader based phase clocks since the leader(s) may be corrupt. For this reason our phase clocks are leaderless. Phase clocks (without faults) have been used for the majority consensus problem in population protocols [3, 17, 13]. The phase clock based protocol in [17] uses $O(\log n)$ states and converges to the majority value in $O(\log^{5/3} n)$ parallel time, and the time bound was further improved to $O(\log^{3/2} n)$, again using phase clocks [13] (which do not tolerate faults). The best known bounds for no faulty nodes are given in [25], with $O(\log n)$ states and $O(\log n)$ expected parallel time.

Fault-tolerant population protocols have been considered in [7, 19, 29]. Such protocols model weak devices, such as battery-operated sensor networks, which can easily crash fail, or be compromised and act maliciously. Other applications include reactions to virus attacks or cell deterioration in biological systems. Our work on population protocols builds upon the concepts in [17, 13, 19].

Power of Two Choices. PoTC has been studied with $\ell \geq 2$ choices [10, 16], and there is a significant benefit if there is an asymmetry between the choices [43]. PoTC has also been studied with balls of different weights [39], and considering deletions [12]. There are generalizations to graphical allocations [11, 32], where bins are nodes in a graph and balls are randomly picked edges. The problem with graphical allocations has also been studied as a $(1+\beta)$-choice process [40], where each ball goes to a random bin with probability $1-\beta$ and to a less loaded of two random bins with probability β. The $(1+\beta)$-choice process can be used to model faults, but it does not capture properly the Byzantine adversary that we consider here. We focus on worst-case Byzantine failures, while the $(1+\beta)$-choice process assumes the failures are random. Fault-tolerance of PoTC has also been studied recently in [26], which presents several tradeoffs between the number of bins m, faults f and nodes n. But the number of considered balls is too limited to apply the results to our population protocol analysis here; which is why we provide a novel phase clock analysis. The work in [34] studies load balancing applications and allows incorrect results (similar to faults).

Paper Outline

In Section 2 we give some common notations that are used throughout the analysis. In Section 3, we give the analysis of the phase clock gap lower bounds in Theorems 1 and 2. In Section 4, we present the analysis of the phase clock gap upper bound and the proof of Theorems 3 and 4. The application of our phase clock to population protocols, and the proof of Theorem 5, appears in Section 5. Finally, we give our conclusions and open problems in Section 6. Some of the results and detail model definitions appear in the Appendix.

2 Notations

In the Fault-Tolerant Phase Clock, we consider n bins and m balls. There are h honest bins and f faulty bins, $h + f = n$. At the start of the process, all bins are empty. At each time $1 \leq t \leq m$, one ball lands in one of the bins. We use $l_b(t)$ to denote the load (i.e., number of balls) of bin b at time t.

At each time t, we partition the honest bins into groups of bins $G_{max}(t)$ and $G_{min}(t)$, which contain h_1 of the most loaded honest bins and h_2 of the least loaded bins respectively, $h_1 + h_2 = h$. If the parameter t is clear from the context, we may omit it and write simply G_{max} or G_{min}. Which bins belong to each group may change over time. It may happen that some of the bins in G_{max} have the same load as some of the bins in G_{min}; in those cases ties are solved arbitrarily, so that G_{max} contains exactly h_1 bins.

3 Lower bound on the gap of the fault-tolerant phase clock

In this section we present the proofs of Theorems 1 and 2.

3.1 Proof of Theorem 1

Proof of Theorem 1. Consider the following adversarial scenario: The adversary chooses one honest bin, say v, in advance and whenever v interacts with the faulty bins, no ball lands in v, while whenever other honest bins interact with the faulty bins, the ball goes to the honest bin.

We consider m interactions. Notice that for every $n(n-1)/2$ interactions there is on average 1 interaction per each pair of bins, i.e., during m interactions there are on average $A = \frac{2m}{n(n-1)}$ interactions per each pair of bins.

The expected number of balls that bin v receives from interactions with other honest bins is at most $(h-1)A$ balls (the expected value is exactly $(h-1)A$ if v is the unique least loaded bin the entire time) and no balls from faulty bins.

Every other honest bin interacts with $h-2$ honest bins other than v. Together, all honest bins other than v are expected to receive $(h-2)(h-1)A/2$ balls in interactions with each other, $(h-1)fA$ balls from the faulty bins and possibly some balls from interactions with v at times when v is not the least loaded bin. The expected number of balls that an average honest bin other than v receives is at least $(h-2)A/2 + fA$.

For completeness, each faulty bin receives on average $(f-1)A/2$ balls from other faulty bins and A balls from v.

It follows that for $f > n/3$, i.e., $h < 2n/3$, there exists an honest bin w other than v that is expected to receive at least $[(h-2)/2 + f]A = (f + h - h/2 - 1)A = (n - h/2 - 1)A > (2n/3 - 1)A > (h-1)A$ balls, which means that it receives on average $[(h-2)/2+f]A - (h-1)A = (f - h/2)A$ more balls than v. Given that $f, h \in \mathbb{N}$ and $f > n/3$, and $h < 2n/3$ we get that $f - h/2 \geq 1/3$ and thus w is expected to receive $\Omega(A) = \Omega(m/n^2)$ more balls than v.

This means that as more balls are thrown in, the gap between bin v and the average loaded honest bin will tend to increase linearly with the number of balls. This gives a lower bound on the gap between the least loaded and the most loaded honest bin $\Omega(m/n^2)$. ◀

3.2 Proof of Theorem 2

▶ **Lemma 6.** *For any $c > 0$ where $m \geq 144 \cdot c^2 n \ln n$ and*

$$2f \geq h \text{ and } h_1(2f-h)^2 \geq (n/c)^2 \text{ and } h_2(2f-h)^2 \geq (n/c)^2 \text{ and } \frac{h_2 m(2f-h)}{8n(n-1)} \geq \ln n \ , \quad (1)$$

with probability at least $1 - O(1/n)$, for total number of balls a_m in $G_{\max}(m)$ and b_m is $G_{\min}(m)$ we have

$$\frac{a_m}{h_1} - \frac{b_m}{h_2} \geq \frac{m(2f-h)}{2n(n-1)} \ .$$

The proof of Lemma 6 will appear in the full version.

From Lemma 6, when setting $2f - h \geq n/c$, and $h_1, h_2 \geq 1$, then all Inequalities 1 are satisfied. Since $n = h + f$, we get $2f - (n-f) \geq n/c$ which implies $f \geq n/3 \cdot (1+1/c)$. Hence, for $m \geq 144 \cdot c^2 n \ln n$, the gap between the most loaded bin and least loaded bin is at least

$$\frac{a_m}{h_1} - \frac{b_m}{h_2} \geq \frac{m(2f-h)}{2n(n-1)} \geq \frac{mn/c}{2n(n-1)} = \frac{m}{2c(n-1)} \ .$$

This completes the proof of Theorem 2.

4 Upper bound on the gap of FTPC

In this section we present the proof of an upper bound on the gap of FTPC, stated in Thm. 3.

We assume an adversary that decides to put a ball in the faulty bin whenever a faulty bin interacts with one of the k least loaded honest bins. Otherwise, i.e., whenever a faulty bin interacts with one of the $h-k$ most loaded bins, the adversary puts the ball in the honest bin.

4.1 Overview

We split the honest bins in FTPC into two groups, G_{max} and G_{min}. We show that both groups behave similar to a selected $(1+\beta)$-choice process, so the gap inside each group is bounded, see Section 4.2. Finally, we bound the gap between the two groups in Section 4.3.

However, in order to show that each group behaves like a $(1+\beta)$-choice process, we prepare auxiliary processes $\alpha_1, \alpha_2, \alpha_3$ and γ_1, γ_2 in Section 4.2.1. Process α_1 is a simple process that only serves as a baseline to build upon. Process α_2 is a $(1+\beta)$-choice process that behaves almost like the considered groups of bins in FTPC. However, the similarities are insufficient to directly compare it against FTPC. That is why we prepare intermediary processes α_3, γ_1 and γ_2 that can be compared both against FTPC and against the $(1+\beta)$-choice process α_2 with a known gap. The comparisons between processes can be done through *majorization*, see Section 4.2.2.

In all the above processes, balls need to fall into bins with the right probabilities. In order to make the probabilities right, we introduce the concept of negative balls. We split 0 balls added into a (positive) ball added plus a negative ball added. Then, all the positive balls behave in a certain way that is easy to analyze and all the negative balls can be analyzed separately.

This allows us to conclude that the gap in FTPC is smaller than the gap in process α_3 or γ_2, which is smaller than the gap in process α_2, which is $O(\log n)$ with high probability.

4.2 Gap inside a group

At each time t, let us split all honest bins into group $G_{max}(t)$ which contains $h_1 = h - k$ most loaded honest bins at time t and group $G_{min}(t)$ which contains $h_2 = k$ least loaded honest bins at time t. We will omit the argument (t) whenever it is clear from the context. In this section we analyze the gap inside each group of honest bins G_{max} and G_{min}.

▶ **Lemma 7.** *During FTPC, the gap inside G_{max} is at most $O(\log n)$ with probability at least $1 - 1/n$.*
Similarly, the gap inside group G_{min} is also at most $O(\log n)$ with probability at least $1 - 1/n$.

The proof of Lemma 7 will appear in the full version.

4.2.1 Auxiliary processes

First we describe auxiliary processes that FTPC can be compared against.

Let us define an auxiliary process α_1. Consider n bins and m balls. With probability $p_{discard}$ a ball is discarded. With probability $p_1 < 1 - p_{discard}$, a ball lands in a random bin. Finally, with probability $p_2 = 1 - p_{discard} - p_1$, two bins are chosen uniformly at random and a ball lands in the chosen bin with fewer balls, with ties resolved arbitrarily.

Note that where a ball lands in may depend on the loads of the bins, but is independent of the previous discarded balls. That is to say, the discarded balls do not affect the rest of the process, so we can conceptually discard them all at the start of the process. Let us call the remaining process α_2. Note that in process α_2, there is $\frac{p_1}{p_1+p_2}$ probability that a ball lands in a random bin and $\frac{p_2}{p_1+p_2}$ probability that a ball chooses two random bins and lands in the bin with fewer balls in it. This means that process α_2 is a $(1+\beta)$-choice process, where $\beta = \frac{p_2}{p_1+p_2}$.

Finally, consider process α_3, which behaves like process α_1 (with the same probabilities p_1, p_2), but each time a ball lands in a bin, an adversary can decide to discard that ball instead. Balls discarded by the adversary in this way will be called *negative balls* in future considerations, so that we can say that the ball was still added to the bin, but a negative ball was added as well.

Now, consider a process γ_1 where instead of partitioning the set of balls into discarded, 1-choice and 2-choice balls, we partition them into discarded, 1-choice, 2-choice and low-choice, with probabilities $p'_{discard} \le p_{discard}, p'_1 \le p_1, p'_2 \le p_2$ and $p'_{low} > 0$ respectively. Balls from the last category, low-choice, can land in any of the bins tied for the lowest load.

Again, we remove all the discarded balls from our considerations, creating a process γ_2.

▶ **Lemma 8.** *The gap in process α_2 is at most $O\left(\frac{p_1+p_2}{p_2}\log n\right)$ with probability at least $1 - \frac{1}{n}$. The gap in process γ_2 is at most $O\left(\frac{p_1+p_2}{p_2}\log n\right)$ with probability at least $1 - \frac{1}{n}$.*

Since process α_2 is a $(1+\beta)$-choice process, the bound on the gap in process α_2 follows immediately from the result about $(1+\beta)$-choice process in [40], i.e., the gap is $O(\log n/\beta)$, where $\beta = \frac{p_2}{p_1+p_2}$. Process γ_2, however, is not proven to be a $(1+\beta)$-choice process. In order to prove Lemma 8 for process γ_2 we will use majorization, see Section 4.2.2.

▶ **Lemma 9.** *The gap in process α_1 is at most $O\left(\frac{p_1+p_2}{p_2}\log n\right)$ with probability at least $1 - \frac{1}{n}$. The gap in process γ_1 is at most $O\left(\frac{p_1+p_2}{p_2}\log n\right)$ with probability at least $1 - \frac{1}{n}$.*

Proof. Recall that processes α_1 and γ_1 differ from processes α_2 and γ_2 respectively due to the discarded balls. The observation follows from Lemma 8 and the fact that the discarded balls do not affect the decision of where a ball lands. Processes α_1 and γ_1 on m balls behave like processes α_2 and γ_2 on $(1 - p_{discard})m$ balls, which results in the same bounds on the gaps $O\left(\frac{p_1+p_2}{p_2}\log n\right)$ with probability at least $1 - 1/n$. ◀

▶ **Lemma 10.** *The gap in process α_3 is at most $O\left(\frac{p_1+p_2}{p_2}\log n\right)$ with probability at least $1-\frac{1}{n}$.*

In order to prove Lemma 10, we will use majorization, see Section 4.2.2.

4.2.2 Majorization

We will use the tool of stochastic majorization for vectors.

▶ **Definition 11** (vector majorization)**.** *Let p and q be two vectors. We say that p is* majorized *by q, denoted by $p \preceq q$, if for every i,*

$$\sum_{j \le i} p_\pi(j) \le \sum_{j \le i} q_\sigma(j) \ ,$$

where π is a permutation sorting p in decreasing order and σ is a permutation sorting q in decreasing order.

We generalize the notion of majorization to ball-and-bin processes as follows.

▶ **Definition 12** (process majorization)**.** *Consider a process X such that at each time t the i-th bin receives a ball with probability $p_i(t)$. Similarly, consider a process Y such that at each time t the i-th bin receives a ball with probability $q_i(t)$. Additionally, process X may receive a negative ball (at no time cost) each time a regular ball is added. Process X is* majorized *by a process Y, denoted $X \preceq Y$, if for every t we have the following majorization of probability vectors $p(t) \preceq q(t)$.*

Note that process γ_2 is majorized by process α_2. Indeed, at each step t, the bins with the lowest load in process γ_2 have a higher probability of receiving a ball than in process α_2, which results in all the other bins having a lower probability of receiving a ball in process γ_2 than in process α_2. This means that $\gamma_2 \preceq \alpha_2$.

Below we generalize Theorem 3.1 from [40]. Notably, our generalization allows for dynamically changing probabilities during the execution. Furthermore, we include the possibility of removing balls from the bins.

▶ **Lemma 13.** *Let $p(t)$ and $q(t)$ be probability vectors associated with processes X and Y respectively at time t. Similarly, let $x(t)$ and $y(t)$ be state vectors associated with processes X and Y respectively at time t. Additionally, process X may receive a negative ball (at no time cost) each time a regular ball is added. Consider some time t_{end}. If $p(t) \preceq q(t)$ for every $0 \le t \le t_{end}$ and $x(0) \preceq y(0)$, then $x(t_{end}) \preceq y(t_{end})$.*

Proof. The proof is via a standard coupling argument. We use the following coupling:

1. Pick a number α uniformly from $[0, 1]$.
2. Let i be such that $\sum_{k=1}^{i-1} p_k(t) \le \alpha < \sum_{k=1}^{i} p_k(t)$. Similarly, let j be such that $\sum_{k=1}^{j-1} q_k(t) \le \alpha < \sum_{k=1}^{j} q_k(t)$.
3. Obtain $x(t)$ by putting a ball in i-th bin and $y(t)$ by putting a ball in j-th bin. Optionally, the adversary may remove one ball from any bin in process X.

It is easy to see that the probabilities involved in both processes remain the same as before the coupling. Therefore, the coupling is valid.

It remains to show that $x(t_{end}) \preceq y(t_{end})$. We prove it by induction.

Base of induction. For $t = 0$ we have $x(0) \preceq y(0)$ from the assumption in the lemma.

Inductive step. Assume that $x(t) \preceq y(t)$ for some t. We will show that $x(t+1) \preceq y(t+1)$.

At time t a value of α was randomly chosen. Let i be such that $\sum_{k=1}^{i-1} p_k(t) \leq \alpha < \sum_{k=1}^{i} p_k(t)$, i.e., a ball landed in i-th bin in process x. Similarly, let j be such that $\sum_{k=1}^{j-1} q_k(t) \leq \alpha < \sum_{k=1}^{j} q_k(t)$, i.e., a ball landed in bin j in process y. Since $p(t) \preceq q(t)$, we have $j \leq i$.

Therefore, $\sum_{k=1}^{i'} x_k(t+1) = \sum_{k=1}^{i'} x_k(t)$ for all $i' < i$ and $\sum_{k=1}^{i'} x_k(t+1) = 1+\sum_{k=1}^{i'} x_k(t)$ for $i' \geq i$. Similarly, $\sum_{k=1}^{j'} y_k(t+1) = \sum_{k=1}^{j'} y_k(t)$ for all $j' < j$ and $\sum_{k=1}^{j'} y_k(t+1) = 1 + \sum_{k=1}^{j'} y_k(t)$ for $j' \geq j$.

Adding inductive assumption that $x(t) \preceq y(t)$, we conclude that $\sum_{k=1}^{i'} x_k(t+1) \leq \sum_{k=1}^{i'} y_k(t+1)$ for $i' < i$, as well as for $i \leq i' \leq j$ and for $j \leq i'$, i.e., for all i'.

Furthermore, the adversary can remove a ball from any bin in process X. However, that will not change the inequalities implied by $x(t+1) \preceq y(t+1)$. This means that $x(t+1) \preceq y(t+1)$. ◀

We are now ready to prove Lemmas 8 and 10.

Proof of Lemmas 8 and 10. According to Lemma 13, at any time t the bin vector in process γ_2 (in process α_3 respectively) is majorized by the bin vector in process α_2. This means that there are less balls in the most loaded bins in process γ_2 (process α_3) than in process α_2 and there are more balls in the least loaded bins in process γ_2 (process α_3) than in process α_2. In particular, at any time t, the number of balls in bin 1 (the most loaded bin) is no greater in process γ_2 (process α_3) than in process α_2, while the number of balls in bin h (the least loaded bin) is no smaller in process γ_2 (process α_3) than in process α_2. Therefore, the gap in process γ_2 (process α_3) is smaller than the gap in process α_2.

Since process α_2 is a $(1+\beta)$-choice process, the gap in process α_2 is at most $O\left(\frac{\log n}{\beta}\right) = O\left(\frac{p_1+p_2}{p_2} \log n\right)$ with probability at least $1 - 1/n$. It follows that the gap in process γ_2 (process α_3) is also at most $O\left(\frac{p_1+p_2}{p_2} \log n\right)$ with probability at least $1 - 1/n$. ◀

4.3 Gap-in-the-middle (Proof of Theorem 3)

We have bounded the gap within group G_{max} and the gap within group G_{min}. What remains is to bound the gap between the two groups.

Let us call the distance between the most loaded bin from G_{min} and the least loaded bin from G_{max} the *gap-in-the-middle*. Note that the gap-in-the-middle is exactly 0 whenever a bin switches its group. At such times, we should expect the gap between the most loaded bin and the least loaded bin to be equal to the sum of gaps inside G_{min} and inside G_{max}, i.e., $O(\log n)$.

It remains to analyze the time intervals when the gap-in-the-middle is greater than 0. Note that in such time intervals the bins never switch their groups.

We now show that the gap $g(t)$ between the average load of a bin in $G_{min}(t)$ and the average load of a bin in $G_{max}(t)$ is $O(n^2 \log n)$ with high probability.

▶ **Lemma 14.** *Let us consider any time t such that the most loaded bin from $G_{min}(t)$ has smaller load than the least loaded bin from $G_{max}(t)$. Let $t_0 < t$ be the latest time such that the above gap is at most* 1*. Then the gap $g(t)$ between the average load of a bin in $G_{min}(t)$ and the average load of a bin in $G_{max}(t)$ is $O(n^2 \log n)$ with probability at least $1 - 2/n^2$.*

The proof of Lemma 14 will appear in the full version.

To summarize, we have proven that the gap within G_{min} and the gap within G_{max} each are $O(\log n)$ and the gap-in-the-middle is $O(n^2 \log n)$ with probability at least $1 - \frac{1}{n}$. Therefore, the gap between the most loaded honest bin and the least loaded honest bin is also at most $O(n^2 \log n)$ with probability at least $1 - O(1/n)$, which completes the proof of Theorem 3.

4.4 Alternative Upper Bound (Proof of Theorem 4)

Here we give an alternative bound for $f \leq n/8$ as stated in Theorem 4. The proofs of Lemmas 15 and 16 below will appear in the full version. We start with examining the case $m = O(n \log n)$.

▶ **Lemma 15.** *For $n, f \geq 1$, and $m \leq cn \ln n$ exchanges, for any constant $c \geq 1$, with probability at least $1 - 1/n$, every honest node receives at most $6c \log n$ balls.*

Next we show that for $f \leq n/8$ and $m = \Omega(n \log n)$, the gap between the average load in G_{max} and G_{min} is negative. Parameters a_m and b_m are the same as in the proof of Lemma 14.

▶ **Lemma 16.** *For $n, h_1, h_2 \geq 1$, $1 \leq f \leq n/8$ and $m \geq cn \log n$, for sufficiently large constant c, with probability at least $1 - O(1/n)$, we have $\frac{a_m}{h_1} - \frac{b_m}{h_2} < 0$.*

From the proof of Lemma 14, we only need to consider the case $h_1, h_2 \geq 1$. Hence, from Lemmas 15 and 16, we can obtain an alternative version of Lemma 14 such that for $f \leq n/8$, the gap $g(t)$ between the average load of a bin in $G_{min}(t)$ and the average load of a bin in $G_{max}(t)$ is $O(\log n)$, with probability at least $1-O(1/n)$. This completes the proof of Thm. 4.

5 Fault-resilient Majority Population Protocols

This section applies our fault-tolerant phase clock to design an efficient implementation of the majority consensus algorithm in the population protocols model, and thus prove Theorem 5. We follow the classic population protocol model introduced in [5] but as adapted in [19] to incorporate faulty (Byzantine) nodes (see Appendix A for more details on the model). Similar to [5], we assume that in an exchange $\{u, v\}$ between nodes u and v, one node is the *initiator* while the other is the *responder*; thus, symmetry-breaking is not an issue in our model.

We study population protocols for the majority consensus problem with $f \leq n/c_f$ Byzantine (faulty) nodes, for some constant c_f. Each honest node starts with an initial preference value in the set $\{A, B\}$ and all honest nodes will decide the same value in $\{A, B\}$. In [19] they use a phase clock with phase duration $D = \Theta(\log^2 n)$ parallel time steps (where a parallel time step corresponds to n consecutive exchanges) that can tolerate f Byzantine nodes for the majority problem. The whole protocol in [19] runs in $O(\log n)$ phases until all honest nodes reach the same decision. Thus, the total running time is $O(\log^3 n)$ and each node has $O(\log^3 n)$ states to maintain the phase clock. Here, we improve this running time by using shorter phases, that is, $D = \Theta(\log n)$.

The phase clock in [19] is implemented at each node u with a local counter C_u which is incremented whenever node u participates in an exchange. Algorithm 1 is an adaptation of the population protocol structure in [19] so that it uses a power of two choices scheme to

Algorithm 1 Majority Population Protocol for Node u.

```
Input: preference value_u ∈ {A, B}
Output: decision_u ∈ {A, B}
// Initialization for node u
C_u ← −1; // local counter initialization
phase_u ← 0; // local phase initialization
decision_u ← ⊥; // decision initialization
// Actions on exchange involving u
foreach exchange {u, v} do
    // maxPhases = Θ(log n) and is the same for all nodes; C_u will stop incrementing after
       phase_u ≥ maxPhases
    if phase_u < maxPhases then
        if C_u < C_v then
            C_u ← C_u + 1; // increment local counter C_u
        else
            if C_u = C_v and u is the initiator of exchange {u, v} then
                C_u ← C_u + 1; // only C_u is incremented in this case because u is the initiator
                   of the exchange; C_v is not incremented because v is the responder
    phase_u ← ⌊C_u/D⌋; // update local phase; D = Θ(log n) is the phase duration which is the same
       for all nodes
    if phase_u < maxPhases then
        // From C_u determine the current phase kind: Cancellation, Resolution, Duplication
        // Cancellation Phase (repeats Θ(1) times)
        if u is in cancellation phase and phase_u = phase_v then
            if value_u is opposite of value_v then
                value_u ← ⊥;
        // Resolution Phase
        if u is in the resolution phase and decision_u = ⊥ then
            // σ_1, σ_2 ∈ Θ(log n) such that σ_1 < σ_2
            During the phase u collects Θ(log n) samples of other nodes' values;
            if at least σ_2 samples are for value X ∈ {A, B}, and at most σ_1 samples are for the
              opposite value then
                decision_u ← X;
                // Once decision_u ≠ ⊥, u will be passive in subsequent phases
        // Duplication Phase
        if u is in duplication phase and phase_u = phase_v then
            if value_u = ⊥ then
                value_u ← value_v;
```

increment the local counters. In an exchange $\{u, v\}$ the local counter C_u is only updated if $C_u < C_v$. In case $C_u = C_v$, then only the initiator of the exchange will have its counter incremented. This allows the Fault-Tolerant Phase Clock (FTPC) properties proven in Theorem 3 to be applied in the local counters and provide tighter length phases.

The current phase number at node u is determined as $phase_u = \lfloor C_u/D \rfloor$. In [19] the maximum phase number is $O(\log^3 n)$, which also gives the same bound on the number of required states per node. Each phase is further divided into three subphases, each of duration $D/3$. The first and third subphases act as buffer time zones for the middle subphase. The main actions of the protocol occur during the middle subphase, since when a node u is in the middle subphase of a phase ϕ, then every other node must be in the same phase ϕ.

The main idea for the protocols for the majority problem is to use three kinds of phases:

- *Cancellation phase:* where each node with value A (or B) is matched with the "opposite" value B (or A); this results in the creation of *empty nodes* (with empty value $\perp$).
- *Resolution phase:* where each node determines whether to decide on a majority value, which is accomplished by randomly sampling $\Theta(\log n)$ values of other nodes. If most of the samples (at least σ_2) are of value $X \in \{A, B\}$ and fewer samples (at most $\sigma_1 < \sigma_2$) are of the opposite value, then all nodes decide X in that phase.
- *Duplication phase:* where each node duplicates its value to an empty node.

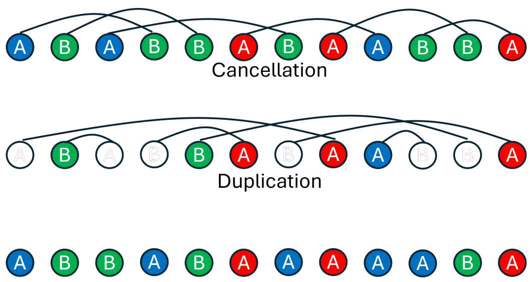

Figure 1 Illustration of inversion of the majority by faulty nodes. The red nodes are faulty and collude to make A the majority. At the top part, we have the initial configuration, where B is the majority for the honest nodes with tally difference $d = 6 - 3 = 3$. After $n/2$ cancellation exchanges (half of a parallel time step) we obtain the middle configuration, where B is still the majority. After $n/2$ duplication exchanges, we obtain the configuration at the bottom where A is now the majority.

Each of the three phases actually requires only $\Theta(\log n)$ time (despite $D = \Theta(\log^2 n)$). Let d denote the absolute difference of the initial preference tallies of the honest nodes. Each cycle of cancellation-resolution-duplication phases increases the difference d by a constant factor. Thus, after $\Theta(\log n)$ cycles (i.e. $\Theta(\log n)$ phases), the difference of tallies becomes $\Omega(n)$ and the majority value prevails. At that point, all honest nodes decide the majority value, since most samples in the resolution phase are of the majority value.

The faulty nodes can adversely affect the outcome. If f is sufficiently large with respect to d, then the faulty nodes may invert the majority value to the opposite value, and hence the minority value could prevail. See Figure 1 for an example of majority inversion. In [19] is actually shown that it must be $d \geq 2f$ (for $f > 0$) to prevent this adverse outcome. Note that in [19] they assume a more general Byzantine adversary than the k-adversary that we defined in Section 1; hence, for the phase clock part of the algorithm, we assume a k-adversary.

Three algorithms are presented in [19], which give tradeoffs between d and f. The first two algorithms (Asymmetric-C-Partial-D and Symmetric-C-Full-D) have deterministic transition functions for the local node states, while the third algorithm (Combined-C-D) combines the first two algorithms into a single algorithm with a unified assumption for d; this algorithm uses randomization. In the three algorithms, the following result (Lemma 5 from from [19]) is central in the analysis, which establishes that $D = \Theta(\log^2 n)$.

Lemma 5 from [19]: For $n \geq e^2$, at the rth exchange issued by the scheduler, where $r \leq cn \ln^3 n$ and constant $c \geq 1$, for all pairs of nodes u and v it holds that $|C_u - C_v| < 2\zeta$, where $\zeta = \sqrt{12c} \ln^2 n$, with probability at least $1 - 2/n$.

The result of Lemma 5 from [19] is independent of f, as it only concerns the local counters of honest nodes. The reason why the duration is $D = \Theta(\log^2 n)$ is due to the use of a Chernoff bound for the difference of the local counters for $O(n \log^3 n)$ exchanges. This can improve by using Theorem 4 which gives the following result for $O(n \log^2 n)$ exchanges.[2]

▶ **Lemma 17** (New Phase Clock for Population Protocols). *At every rth exchange issued by the scheduler, where $r \leq cn \ln^2 n$ and constant $c \geq 1$, for all pairs of nodes u and v it holds that $|C_u - C_v| < \zeta$, where $\zeta = O(\log n)$, with probability at least $1 - O(\log^2 n)/n$.*

[2] Note that Theorem 4 applies with probability $1 - 1/n$. We examine the gap of the clocks every n exchanges. At each such time point, Theorem 4 guarantees $O(\log n)$ gap. Between these points, it can be easily shown with Chernoff bound that the gap could have an excess of $O(\log n)$ (w.h.p.). Since, there are $O(\log^2 n)$ such points, the new phase clock result follows.

Using Lemma 17, the duration of the phase becomes $D = \Theta(\log n)$. This replaces the phase duration in [19]. which immediately improves the time and state bounds by a $\Theta(\log n)$ factor. We continue to discuss the new bounds for each of the three protocols in [19].

5.1 Algorithm Asymmetric-C-Partial-D

The reason behind the name of this algorithm is that the cancellations are asymmetric. In the cancellation phase, the algorithm gives only one chance for each node to cancel its value. This gives more resiliency to forced cancellations by faulty nodes. The duplication phase is treated similarly.

Consider an honest node u with value $X \in \{A, B\}$ which has its first exchange in the cancellation phase. If u meets with another node v with the opposite value than X, then u will cancel its value and replace it with $\perp$. Once u tries to cancel once in the phase, then it will not try again in the same phase. If v happened to also be in the first exchange then, v will also cancel its value together with u. But if v already tried earlier to cancel in the phase (successfully or not), then v will not be affected by u even if it has the opposite value of X. In this way, the probability that u is successful to cancel its value depends on the fraction of nodes that have the opposite value versus the faulty nodes. If d is sufficiently large compared to f then the cancellation will be successful. The whole cancellation phase repeats a constant number of times before proceeding to the next kind of phase (resolution phase).

A similar situation also occurs in the duplication phase, where an empty node u (with value $\perp$) will only attempt once to adopt a value in $\{A, B\}$ in an exchange when it meets with a non-empty node. This way, there is just a single chance for a faulty node to affect u during the phase.

This algorithm works for $d = \Omega(f + \sqrt{n \log n})$. When $f = \Omega(\sqrt{n \log n})$, d is asymptotically optimal due to the lower bound $d = \Omega(f + 1)$. The following is a restatement of Theorem 1.1 from [19], when using the phase clock with $D = \Theta(\log n)$ from Lemma 17.

> *Restated Theorem 1.1 from [19] with $D = \Theta(\log n)$*: For sufficiently large n, and $f \leq n/c_f$, for some constant c_f, if $d = \Omega(f + \sqrt{n \log n})$ then with Algorithm Asymmetric-C-Partial-D all honest nodes decide the majority value in $O(\log^2 n)$ parallel time steps using $O(\log^2 n)$ states per node with probability at least $1 - O(\log^2 n)/n$.

5.2 Algorithm Symmetric-C-Full-D

The reason behind the name of this algorithm is that the cancellations are symmetric, and the nodes have multiple chances to cancel their values during a phase. Similarly, the duplications may be repeated multiple times during a phase. Because of multiple chances in a phase, honest nodes are more vulnerable to faulty nodes. For this reason, the minimum d is a larger function of f.

This algorithm assumes $d = \Omega(\min\{f \log^2 n + 1, n\})$. The algorithm can also work for smaller $d = o(\sqrt{n \log n})$ (compared to Asymmetric-C-Partial-D). However, f is multiplied by a poly-log factor of n. The analysis of the algorithm depends on the following restated Lemma 14 from [19] which depends on the duration of a phase.

> *Restated Lemma 14 from [19]*: The number of exchanges between the first exchange of any node entering phase ϕ and the last exchange of any node in phase ϕ is $O(n \log^2 n)$ with high probability (w.h.p.).

We replace Lemma 14 from [19] with the following result which we prove using Theorem 4.

▶ **Lemma 18.** *Let the $O(\log n)$ gap in Theorem 3 be $c_1 \log n$ for some $c_1 > 0$. Let the length of every phase (i.e., the number of exchanges that each node participates in during a phase) be $D = c_2 \log n$. Then, the number of exchanges between the first exchange of any node entering phase ϕ and the last exchange of any node in phase ϕ, is at most $(c_1 + c_2)n \log n$ w.h.p.*

Proof. Consider a phase $\phi \geq 0$. Let e be the first exchange of any node, say node u, to enter phase ϕ and e' be the last exchange of any node, say node v, in phase ϕ. Note that it may be $u \neq v$. Let τ be the period between the exchanges from e to e'. Suppose to the contrary that the number of exchanges in τ is $c_3 n \log n$, where $c_3 > c_1 + c_2$. In each exchange, the local counter of one participating node increases by one. Hence, by the pigeonhole principle, some node x must have increased its local counter C_x by at least $c_3 \log n$ during τ. Let C_u and C_x be the respective local counters at u and x just after e occurs. It must be that $\phi = \lfloor C_u/D \rfloor$. From Theorem 4 we get $C_u - C_x \leq c_1 \log n$ with high probability. Thus, at the end of τ, the local counter at x becomes at least $C_x + c_3 \log n \geq C_u - c_1 \log n + c_3 \log n > C_u + c_2 \log n = C_u + D$ with high probability. Hence, the local phase of x at the end of τ is at least $\phi' \geq \lfloor (C_u + D)/D \rfloor > \phi$, a contradiction. ◀

With Lemma 18, we obtain the following restatement of Theorem 1.2 from [19], when using the phase clock with $D = \Theta(\log n)$ from Lemma 17.

> *Restated Theorem 1.2 from [19] with $D = \Theta(\log n)$*: For sufficiently large n, and $f \leq n/c_f$, for some constant c_f, if $d = \Omega(\min\{f \log^2 n + 1, n\})$ then with Algorithm Symmetric-C-Full-D all honest nodes decide the majority value in $O(\log^2 n)$ parallel time steps using $O(\log^2 n)$ states per node with probability at least $1 - O(\log^2 n)/n$.

5.3 Algorithm Combined-C-D

This algorithm combines the above two mentioned Algorithms Asymmetric-C-Partial-D and Symmetric-C-Full-D into a single algorithm that has the benefits of both. The algorithm uses a random bias to adapt to the best algorithm for the values of d and f. The transition function of the nodes is *randomized*. Combining the restated Theorem 1.1 from [19] about algorithm Asymmetric-C-Partial-D, as in Section 5.1, with the restated Theorem 1.2 from [19] about algorithm Symmetric-C-Full-D, as in Section 5.2, we obtain the proof of Theorem 5.

6 Conclusions

We showed that it is possible to design a phase clock that tolerates $f < n/3$ faulty nodes under a Byzantine adversary so that the difference between the counters of any two nodes is at most $O(\log n)$. We showed that such a phase clock is tight by giving a respective lower bound analysis for the gap. Then, we applied the phase clock for the analysis of a population protocol for the majority consensus problem. As an open problem, it would be interesting to determine if there are improved majority population protocols with Byzantine faulty nodes that require less than $O(\log^2 n)$ time.

Another open problem is to examine ways to break the symmetry between two nodes involved in an exchange $\{u, v\}$ without assuming that one of the nodes is an initiator and the other the responder. This could be achieved by using random bits in each node and it would be interesting to examine how many such random bits are necessary. In fact, symmetry breaking is only needed when both nodes have the same local counter value ($C_u = C_v$). Hence, another way to avoid symmetry breaking is to allow both local counters to increment; however, this needs to adapt accordingly the analysis of Section 4 which is an interesting problem to study in the future.

References

1 Muhammed Abdulazeez, Pawel Garncarek, Dariusz R. Kowalski, and Prudence W. H. Wong. Lightweight robust framework for workload scheduling in clouds. In *IEEE International Conference on Edge Computing, EDGE 2017, Honolulu, HI, USA, June 25-30, 2017*, pages 206–209. IEEE Computer Society, 2017. doi:10.1109/IEEE.EDGE.2017.36.

2 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. SIAM, 2017. doi:10.1137/1.9781611974782.169.

3 Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *Proceedings of the 2018 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2221–2239, 2018. doi:10.1137/1.9781611975031.144.

4 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. ACM, 2015. doi:10.1145/2767386.2767429.

5 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Comput.*, 18(4):235–253, 2006. doi:10.1007/s00446-005-0138-3.

6 Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, September 2008. doi:10.1007/S00446-008-0067-Z.

7 Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Comput.*, 21(2):87–102, 2008. doi:10.1007/s00446-008-0059-z.

8 Anish Arora, Shlomi Dolev, and Mohamed Gouda. Maintaining digital clocks in step. In *Distributed Algorithms: 5th International Workshop, WDAG'91 Delphi, Greece, October 7–9, 1991 Proceedings 5*, pages 71–79. Springer, 1992.

9 Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics.* Wiley, Second edition, 2004.

10 Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM Journal on Computing*, 29(1):180–200, 1999. doi:10.1137/S0097539795288490.

11 Nikhil Bansal and Ohad N. Feldheim. The power of two choices in graphical allocation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 52–63, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3519995.

12 Nikhil Bansal and William Kuszmaul. Balanced allocations: The heavily loaded case with deletions. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 801–812, 2022. doi:10.1109/FOCS54457.2022.00081.

13 Stav Ben-Nun, Tsvi Kopelowitz, Matan Kraus, and Ely Porat. An o(log3/2 n) parallel time population protocol for majority with o(log n) states. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC '20, pages 191–199, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3382734.3405747.

14 Michael Ben-Or, Elan Pavlov, and Vinod Vaikuntanathan. Byzantine agreement in the full-information model in o(log n) rounds. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 179–186, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1132516.1132543.

15 R. Beraldi and G. Mattia. Power of random choices made efficient for fog computing. *IEEE Transactions on Cloud Computing*, 10(02):1130–1141, April 2022. doi:10.1109/TCC.2020.2968443.

16 Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: The heavily loaded case. *SIAM Journal on Computing*, 35(6):1350–1385, 2006. `doi:10.1137/S009753970444435X`.

17 Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. A population protocol for exact majority with $O(\log^{5/3} n)$ stabilization time and $\Theta(\log n)$ states. In *32nd International Symposium on Distributed Computing (DISC 2018)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. `doi:10.4230/LIPIcs.DISC.2018.10`.

18 Maury Bramson, Yi Lu, and Balaji Prabhakar. Asymptotic independence of queues under randomized load balancing. *Queueing Syst. Theory Appl.*, 71(3):247–292, July 2012. `doi:10.1007/s11134-012-9311-0`.

19 Costas Busch and Dariusz R. Kowalski. Byzantine-resilient population protocols. *CoRR*, abs/2105.07123, 2021. `arXiv:2105.07123`.

20 Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 42–51. ACM, 1993. `doi:10.1145/167088.167105`.

21 Richard Cole, Bruce M. Maggs, Friedhelm Meyer auf der Heide, Michael Mitzenmacher, Andréa W. Richa, Klaus Schröder, Ramesh K. Sitaraman, and Berthold Vöcking. Randomized protocols for low-congestion circuit routing in multistage interconnection networks. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 378–388, New York, NY, USA, 1998. Association for Computing Machinery. `doi:10.1145/276698.276790`.

22 Colin Cooper, Robert Elsässer, and Tomasz Radzik. The power of two choices in distributed voting. In *Automata, Languages, and Programming*, pages 435–446, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. `doi:10.1007/978-3-662-43951-7_37`.

23 Ariel Daliot, Danny Dolev, and Hanna Parnas. Self-stabilizing pulse synchronization inspired by biological pacemaker networks. In *Symposium on Self-Stabilizing Systems*, pages 32–48. Springer, 2003. `doi:10.1007/3-540-45032-7_3`.

24 Shlomi Dolev and Jennifer L. Welch. Self-stabilizing clock synchronization in the presence of byzantine faults. *J. ACM*, 51(5):780–799, September 2004. `doi:10.1145/1017460.1017463`.

25 David Doty, Mahsa Eftekhari, Leszek Gasieniec, Eric E. Severson, Przemyslaw Uznanski, and Grzegorz Stachowiak. A time and space optimal stable population protocol solving exact majority. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1044–1055. IEEE, 2021. `doi:10.1109/FOCS52979.2021.00104`.

26 Paweł Garncarek, Costas Busch, and Dariusz Kowalski. Locally balanced allocations under strong byzantine influence. In *31st International Colloquium On Structural Information and Communication Complexity (SIROCCO 2024)*, 2024.

27 Leszek Gąsieniec and Grzegorz Stachowiak. Enhanced phase clocks, population protocols, and fast space optimal leader election. *J. ACM*, 68(1), November 2020. `doi:10.1145/3424659`.

28 Gaston H. Gonnet. Expected length of the longest probe sequence in hash code searching. *J. ACM*, 28(2):289–304, April 1981. `doi:10.1145/322248.322254`.

29 Rachid Guerraoui and Eric Ruppert. Names trump malice: Tiny mobile agents can tolerate byzantine failures. In *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, volume 5556 of *Lecture Notes in Computer Science*, pages 484–495. Springer, 2009. `doi:10.1007/978-3-642-02930-1_40`.

30 T. Herman. Phase clocks for transient fault repair. *IEEE Transactions on Parallel and Distributed Systems*, 11(10):1048–1057, 2000. `doi:10.1109/71.888644`.

31 Norman L. Johnson and Samuel Kotz. Urn models and their application : an approach to modern discrete probability theory. *International Statistical Review*, 46:319, 1978.

32 Krishnaram Kenthapadi and Rina Panigrahy. Balanced allocation on graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 434–443, 2006. URL: http://dl.acm.org/citation.cfm?id=1109557.1109606.

33 Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. doi:10.1145/357172.357176.

34 Dimitrios Los and Thomas Sauerwald. Balanced allocations with the choice of noise. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, pages 164–175, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538428.

35 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1st edition, 1996. URL: http://www.amazon.com/Distributed-Algorithms-Kaufmann-Management-Systems/dp/1558603484.

36 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. *Distributed Comput.*, 30(1):1–16, 2017. doi:10.1007/s00446-016-0277-8.

37 C. J. Park. Random allocations (valentin f. kolchin, boris a. sevast'yanov and vladimir p. chistyakov). *Siam Review*, 22:104–104, 1980.

38 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980. doi:10.1145/322186.322188.

39 Yuval Peres, Kunal Talwar, and Udi Wieder. The (1+beta)-choice process and weighted balls into bins. In *SODA'2010*, January 2010. URL: https://www.microsoft.com/en-us/research/publication/the-1beta-choice-process-and-weighted-balls-into-bins/.

40 Yuval Peres, Kunal Talwar, and Udi Wieder. Graphical balanced allocations and the $1+\beta$-choice process. *Random Struct. Algorithms*, 47(4):760–775, December 2015. doi:10.1002/rsa.20558.

41 Andréa Richa, Michael Mitzenmacher, and Ramesh Sitaraman. The power of two random choices: A survey of techniques and results. *Handbook of Randomized Computing*, 2001. doi:10.1007/978-1-4615-0013-1_9.

42 A.S. Tanenbaum and M. van Steen. *Distributed Systems*. CreateSpace Independent Publishing Platform, 2017. URL: https://books.google.com/books?id=c77GAQAACAAJ.

43 Berthold Vöcking. How asymmetry helps load balancing. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, page 131, USA, 1999. IEEE Computer Society.

A Detailed Population Protocol Model and Other Preliminaries

A.1 Basic Model

We adopt the population protocol model originally presented in [5] and later revised in [19] for faulty nodes. There is a set of n *nodes* V connected with each other in an unweighted clique. The nodes do not have any identifiers. There is a scheduler that randomly picks exchanges between pairs of nodes. In an exchange, a pair of nodes $\{u, v\}$, where $u, v \in V$, is chosen uniformly at random from the $\binom{n}{2}$ node pairs. In exchange $\{u, v\}$ one of the two nodes is the *initiator* while the other is the *responder*; this is used to break symmetry. (We note that the assumption of initiator-responder symmetry breaking was not used in [19].)

Each node has a *state* taken from a set of states S. An exchange affects the states of the involved nodes according to a joint *transition function* $\Delta : S \times S \to S \times S$. Namely, when an exchange pair $\{u, v\}$ is chosen then their current states s_u, s_v become s'_u, s'_v, where $\Delta(s_u, s_v) = (s'_u, s'_v)$. The transition function takes into account which node is the initiator and which is the responder but in any other way it is symmetric with respect to u and v. Function Δ is deterministic in case of deterministic algorithms, and random in the case of randomized algorithms.

There are also input and output functions related to the task that the population protocol attempts to solve. There is a finite set of input symbols $X \subseteq S$, a finite set of output symbols $Y \subseteq S$, and an output function $\Gamma : S \to Y$. An input function Λ maps every node to an initial input symbol, that is, $\Lambda : V \to X$. For the *majority problem* the input and output symbols are $X = Y = \{A, B\}$. A configuration (collection of node states) at exchange t is *stable* if for each exchange after t the output of every node remains the same.

For the majority problem, a stable configuration is *correct* if the output of all the nodes is the majority value in the initial configuration. A protocol is *always correct* if the correct stable configuration is reached with probability 1, and is *w.h.p. correct* if the correct stable configuration is reached with probability $1 - n^{-c}$ for some constant $c > 0$. Note that for protocols which are w.h.p. correct it is possible that a stable state is not reached, or an incorrect stable state is reached. A *parallel time step* corresponds to n consecutive exchanges. The *stabilization time* is the number of parallel time steps until a correct stable configuration is reached. The *time complexity* is the stabilization time measured in parallel steps.

A.2 Byzantine Adversary

The *Byzantine adversary* corrupts a set of f nodes $F \subseteq V$, while the remaining nodes $V \setminus F$ remain *honest*. A *static* adversary determines before the execution starts which nodes to corrupt (after seeing the nodes' inputs), while a *dynamic* adversary determines which nodes to corrupt by monitoring the execution and seeing the current states of the nodes. We assume here that the adversary is dynamic. Moreover, we assume the adversary is *full dynamic* so that it can observe all current states of nodes. (This is in contrast to the weak dynamic adversary that only observes the state of a node after an exchange occurs with that node.)

Faulty nodes affect the states of honest nodes via exchanges. Changes in the honest nodes' states may only occur through the transition function Δ. When a faulty node u participates in an exchange $\{u, v\}$ with an honest node v, the faulty node u may pretend to have an honest node's state and through the Δ function it may affect the resulting state of the honest node v. A configuration is *stable* if the output of each *honest* node remains the same thereafter. In a correct stable configuration of a majority problem, all honest nodes output the majority value of the initial configuration. The output of faulty nodes in stable configurations is not taken into account. The notions described previously of always correct, w.h.p. correct, and stabilization time, remain the same as in the basic model.

A.3 Other Preliminaries

▶ **Lemma 19** (Chernoff bound). *Let $X_1, X_2, \cdots, X_m$ be independent Poisson trials such that, for $1 \le i \le m$, $X_i \in \{0, 1\}$. Then, for $X = \sum_{i=1}^{m} X_i$, $\mu = E[X]$, and any $\delta > 1$, $\Pr[X \ge (1+\delta)\mu] \le e^{-\frac{\delta\mu}{2}}$, and any $0 \le \delta \le 1$, $\Pr[X \ge (1+\delta)\mu] \le e^{-\frac{\delta^2\mu}{3}}$ and $\Pr[X \le (1-\delta)\mu] \le e^{-\frac{\delta^2\mu}{2}}$.*