Distributed Computing
https://doi.org/10.1007/s00446-020-00385-0

Time-space trade-offs in population protocols for the majority problem

Petra Berenbrink[1] · Robert Elsässer[2] · Tom Friedetzky[3] · Dominik Kaaser[1] · Peter Kling[1] · Tomasz Radzik[4]

Received: 19 July 2019 / Accepted: 22 July 2020

Abstract
Population protocols are a model for distributed computing that is focused on simplicity and robustness. A system of n identical agents (finite state machines) performs a global task like electing a unique leader or determining the majority opinion when each agent has one of two opinions. Agents communicate in pairwise interactions with randomly assigned communication partners. Quality is measured in two ways: the number of interactions to complete the task and the number of states per agent. We present protocols for the majority problem that allow for a trade-off between these two measures. Compared to the only other trade-off result (Alistarh et al. in Proceedings of the 2015 ACM symposium on principles of distributed computing, Donostia-San Sebastián, 2015), we improve the number of interactions by almost a linear factor. Furthermore, our protocols can be made uniform (working correctly without any information on the population size n), yielding the first uniform majority protocols that stabilize in a subquadratic number of interactions.

Keywords Distributed computing · Majority · Population protocols · Stochastic processes

Contents

✉ Peter Kling
peter.kling@uni-hamburg.de

Petra Berenbrink
petra.berenbrink@uni-hamburg.de

Robert Elsässer
elsa@cs.sbg.ac.at

Tom Friedetzky
tom.friedetzky@durham.ac.uk

Dominik Kaaser
dominik.kaaser@uni-hamburg.de

Tomasz Radzik
tomasz.radzik@kcl.ac.uk

[1] Universität Hamburg, Vogt-Kölln-Str. 3, 22527 Hamburg, Germany

[2] University of Salzburg, Jakob-Haringer-Str. 2, 5020 Salzburg, Austria

[3] Durham University, Lower Mountjoy South Road, Durham DH1 3LE, United Kingdom

[4] King's College London, Strand Campus Bush House, London WC2B 4BG, United Kingdom

1 Introduction

In this article we consider the *majority* problem in the probabilistic *population model*. Majority is a fundamental problem in distributed computing. There are n different agents, each with one of two opinions, say A and B and the goal is to agree on the opinion with the larger support. This problem occurs when all elements of a distributed system have to reach consensus on the value of some parameter which reflects the prevailing opinion what this value should be. Because of its importance, the majority problem is frequently used as a case

Published online: 05 August 2020

study in analysis and comparison of strengths and limitations of various models of distributed computing.

The population model was introduced by Angluin et al. [4,5] as a model to explore the computational power of resource-limited, mobile agents. Agents are modeled as finite-state machines. In every step, a pair of agents is chosen uniformly at random, observe each other's state, and perform a deterministic state transition. This is called an *interaction*. States are mapped to outputs by a problem-specific output function. In the case of the majority problem, one can think of an agent's output as being A or B, indicating which opinion the agent believes to be the majority.

The quality of a population protocol is measured in terms of the number of interactions (the runtime) and the number of states per agent required to "successfully compute" the desired output. The number of interactions is sometimes expressed in *parallel time*, which divides the number of interactions by n to account for the inherent parallelism of the system. In order to avoid confusion, we stick to the actual number of interactions throughout the article.

There are several definitions for what is conceived as a "successful computation". A typical requirement is that the system must, eventually, reach a state with correct output and which is *stable*—i.e., no possible future transition can change the agents' output. However, runtime notions differ in *when* this strict guarantee must be achieved. A natural definition is to measure the number t of interactions after which the system is in such a stable state with correct output. This notion is used in most recent publications, especially for lower bounds (cf. Sect. 1.2). Another definition considers the number of interactions t after which the *current execution* always gives the correct output. The former runtime notion is typically referred to as *stabilization* and the latter as *convergence* (see Sect. 2).

One may wonder what the advantage in measuring the convergence time instead of the stabilization time may be. In [9] the authors introduce a hybrid protocol that combines a "fast" protocol that might never converge to the correct answer with a "slow" one that stabilizes at the correct answer. The hybrid protocol switches its output from the fast protocol, which might be incorrect, to that of the slow but always correct protocol when it is likely that the slow protocol has finished. And therein lies the crux: without further safeguards, it is possible, although with only negligible probability, that a correct output reached by the fast protocol at time t is later temporarily overwritten by a currently still wrong output of the slow protocol. Hence, while the system has converged at time t, it is not yet stable. It will stabilize only when the slow protocol does so. The convergence (to the correct output) always happens by the time when the computation stabilizes (on the correct output). The stabilization may, however, be reached later, sometimes much later, than convergence.

A desirable feature of population protocols is *uniformity*, in the sense that a single algorithm should be designed to work for populations of any size. Due to the simplicity of transition-based algorithms and the uniformity, uniform population protocols are well suited to model real-world systems that consist of many but comparatively simple agents, like a flock of birds or large sensor networks aggregating information (count, sum, average, extrema, median, or histogram). In both scenarios the agents' computational power is bounded and the algorithms should not depend on the number of agents.

The underlying theme of this article is to exhibit trade-offs in population protocols between the running time and the required number of states, highlighting methods which help achieving fast stability (in addition to convergence) and uniformity of protocols.

1.1 Our contribution

Our protocols for the majority problem in the population model provide an integer parameter $s \geq 2$ that enables a trade-off between the number of states and the runtime. Our results also depend on the *absolute bias* α, which is the initial absolute difference between the number of agents supporting opinion A and B, respectively. In the following we state the results for the tightest case when $\alpha = 1$; see the corresponding theorems for the full statements.

Our first result is a comparatively simple protocol that, with high probability[1], determines the exact majority in $\mathrm{O}\big(n \cdot (\log n)^2/\log s\big)$ interactions and uses $\Theta\,(s + \log\log n)$ states (Theorem 3). While this high-probability guarantee is comparatively weak with respect to the typical requirement of stabilization or even just convergence (since high-probability correctness allows for some low but positive probability of a permanent error), this protocol is an important building block for the following main results of this article.

1. We present two hybrid exact majority protocols, both having a runtime of $T = \mathrm{O}\big(n \cdot (\log n)^2/\log s\big)$. One *converges* with high probability in T interactions and uses $\Theta\,(s + \log\log n)$ states (Theorem 5). The other *stabilizes* with high probability in T interactions but uses $\Theta\,\big(s \cdot \log_s n\big)$ states (Theorem 4).
2. For a constant s, we provide a uniform version of the second of the above two majority protocols. This protocol has essentially the same guarantee for the stabilization time. However, with high probability it uses $\mathrm{O}\,\big(s \cdot \log_s n \cdot \log\log n\big)$ states (Theorem 6).

[1] We say a property of a protocol holds *with high probability* if for each constant a, the constant parameters of the protocol can be set such that the property holds for each sufficiently large population size n with probability at least $1 - n^{-a}$.

All protocols above except for the uniform one need knowledge of $\lfloor \log \log n \rfloor$. Note that the state space of the uniform protocol is bounded only with high probability; with negligible probability, an agent might need arbitrarily many states. Since this is not covered in the original population model (where agents are *finite*-state machines), for this protocol we adopt a generalized model [14] in which agents are modeled as Turing machines (see Sect. 7).

We highlight a few implications of the above results. For a constant s, our majority results underline an important difference between stabilization and convergence. While the $\Theta(\log n)$ number of states in our stable protocol (Theorem 4) is asymptotically tight for any protocol that stabilizes with high probability in a subquadratic number of interactions[2] [3], our protocol with $\Theta(\log \log n)$ states (Theorem 5) shows that the $\Omega(\log n)$ lower bound can be bypassed if one considers convergence instead of stabilization.

When choosing $s = \log \log n$, our majority protocols converge and stabilize with high probability in $O\big(n \cdot (\log n)^2 / \log \log \log n\big)$ interactions. These and the protocols presented in [11] are the first majority protocols with $O(\text{polylog}\, n)$ states that work in $o\big(n \cdot (\log n)^2\big)$ interactions.

When choosing $s = n^{\epsilon}$, where $\epsilon > 0$ is an arbitrary positive constant, we obtain a majority protocol that stabilizes within asymptotically optimal $O(n \log n)$ interactions using $\Theta(n^{\epsilon})$ states. Before our work, achieving this optimal time required $\Theta\big(n^{3/2}\big)$ states [21].

For a constant parameter s, our uniform protocol that stabilizes in $O\big(n \cdot (\log n)^2\big)$ interactions and uses $O(\log n \cdot \log \log n)$ states (Theorem 6) is the first uniform majority protocol that stabilizes in a subquadratic number of interactions, regardless of the required number of states.

An import ingredient for our results is an improvement to the *phase clock* from [18]—a distributed synchronization mechanism for population protocols. Although this phase clock itself requires just a constant number of states, it is driven by a *junta* of n^{ϵ} agents (for a constant $\epsilon \in [0, 1)$), and selecting such a junta requires $\Theta(\log \log n)$ states. By careful changes to the internals of the junta selection protocol and the interplay between the junta and the phase clocks, we not only simplify the phase clock protocol but also allow agents to "forget" some of the values required to select the junta. This enables us to reduce the number of states required by our majority protocols from a *factor* of $\Theta(\log \log n)$ to an *additive* term of the same order. See Sect. 3.2 for detailed explanations.

1.2 Related literature

The original population model was introduced by Angluin et al. [4,5], assuming that the number of states per agent is constant. Together with Angluin et al. [6,7], their results show that semilinear predicates (which include, e.g., parity and majority) are stably computable in this model. Subsequent results focused on quantifying the runtime and state requirements for specific problems, in particular for the majority and the leader election problems, and on generalizing the model. In the following overview we concentrate on results in the population model for the majority problem. Bear in mind that, as mentioned above, we state any runtime results in terms of the required number of interactions, even when original sources state bounds in parallel time only. For a broader overview of the extent of research and results on protocols for the population model the reader is referred to the survey papers [10] and [17].

Angluin et al. [8] present a protocol with three states and show that, with high probability, the agents agree on the majority after $O(n \log n)$ interactions if the initial difference between both opinions (the *absolute bias* α) is $\omega\big(\sqrt{n} \log n\big)$. Mertzios et al. [20] show that, if agents are required to succeed with probability 1, at least four states are necessary. They also provide a four state protocol that stabilizes with high probability in $O\big(n^2 \log n\big)$ interactions. The same four state protocol was independently (and earlier) studied by Draief and Vojnovic [16], who proved similar results. Alistarh et al. [1] show a lower bound of $\Omega\big(n^2/\alpha\big)$ on the expected interactions for any four state protocol. For any number of states, they show a lower bound of $\Omega(n \log n)$ expected interactions.

To achieve fast runtime, Mocquard et al. [21] consider the population model allowing a super-constant number of states per agent. They present a protocol that calculates the signed difference between the two opinions' support with high probability in asymptotically optimal $O(n \log n)$ interactions but uses polynomial $\Theta\big(n^{3/2}\big)$ number of states. The constant-state but slow quadratic-time protocols [16,20] on the one hand and the fast but polynomial-state protocol [21] on the other, posed the quite natural question of designing fast $O(n\, \text{polylog}\, n)$-time majority protocols which use a relatively small $O(\text{polylog}\, n)$ number of states.

Alistarh et al. [2] show a lower bound on the required number of interactions for population protocols with a small number of states. For majority, their bound states that protocols with less than $(\log \log n)/2$ states require $\Omega\big(n^2/\,\text{polylog}(n)\big)$ interactions in expectation in order to stabilize. Alistarh et al. [3] further improve this lower bound, by showing that any protocol that solves majority and stabilizes in $n^{2-\Omega(1)}$ expected interactions requires $\Omega(\log n)$ states. Both these lower bounds require certain natural monotonic-

[2] Conditioned on some natural properties satisfied by any known protocol, see Sect. 1.2.

ity assumptions which are satisfied by all known majority protocols.

A recent series of papers [1–3,11,13] showed upper bounds. Alistarh et al. [3] present a protocol that stabilizes with high probability in $O\big(n \cdot (\log n)^2\big)$ interactions and requires $O(\log n)$ states. In a recently published result [11], we present a population protocol for majority that reduces the number of interactions to $O\big(n \cdot (\log n)^{5/3}\big)$, both in expectation and with high probability.

The subquadratic-time protocols for majority presented in [1–3,11,13,21] are not uniform. To work correctly, they need an estimate of the size of the population; more precisely, they need a value which is $\Theta(\log n)$. They also, with exception of protocols proposed in [1], provide no means to trade runtime for the number of states required per agent, as our protocols do. Alistarh et al. [1] is the only paper we know of which presents majority protocols with a trade-off of similar nature. For a parameter $m \le n$, their algorithm uses $O(m + \log n \cdot \log m)$ states and stabilizes with high probability in $O\big(n^2 \cdot (\log n)/(\alpha \cdot m) + n \cdot (\log n)^2\big)$ interactions.

In parallel to our work, Kosowski and Uznanski [19] recently designed population protocols, including two majority protocols that converge in $O\big(n(\log n)^3\big)$ and $O\left(n^{1+\epsilon}\right)$ interactions and use $O(\log\log n)$ and constant $f(\epsilon)$ number of states, respectively. Here, ϵ is an arbitrarily chosen positive constant.

With the only exception of [8], all majority protocols mentioned above solve exact majority. That is, they eventually output the correct majority opinion with probability 1. This holds even if the initial bias towards one opinion is as small as only 1.

2 Model & notation

Population protocols are a computational model for a distributed system consisting of n agents, in the following also referred to as *nodes*. Nodes are assumed to be identical finite-state machines.[3] In each time step, an ordered pair of nodes (u, v) is chosen independently and uniformly at random. Node u is called the *initiator* and node v is called the *responder*. Let s_u be the state of u and s_v be the state of v at the beginning of such an *interaction*. Both nodes observe each other's state and update themselves according to a fixed, deterministic transition function of the form $(s_u, s_v) \mapsto (s'_u, s'_v)$. At any time, the global state of the system can be fully described by a function c that maps each node to its current state. This function c is called the *configuration* of the system at that time.

[3] For our uniform protocol, Sect. 7 introduces a generalized model where agents are Turing machines.

Nodes try to reach and stay in a set of *target configurations*, whose definition depends on the considered problem. It is *not* required, indeed not possible in this model, that nodes realize when a target configuration has been reached. Target configurations are specified via an *output function* of the form $s \mapsto o$ that maps a state s to a (problem specific) output value o.

We are interested in population protocols for the majority problem, where nodes start in one of two different states (also called *opinions*). We seek a configuration in which all nodes agree on the opinion with the initially larger support. The *absolute bias* α is the absolute difference between the initial number of supporters for each opinion. We assume $\alpha \ge 1$. The output function maps each state s to an output $o \in \{+1, -1\}$, representing one of the two opinions. The target configurations are all configurations in which node states map all to $+1$, if $+1$ represents the initial majority opinion, or map all to -1, if -1 represents the initial majority.

The quality of a protocol is measured in terms of the number of interactions and the number of states per node required to reach and stay in target configurations. There are two common ways to formalize what exactly is meant by "reach and stay": *stabilization time* and *convergence time*.[4]

Convergence Time: The *convergence time* T_C of a protocol is the random variable that measures the number of interactions until the protocol has reached and remains in the set of target configurations.

Stabilization Time: We say a configuration c is *stable*, if in any configuration c' that is reachable from c by a sequence of interactions, each node has the same output as in c. The *stabilization time* T_{ST} of a protocol is the random variable that measures the number of interactions until the protocol has reached a stable target configuration.

Clearly, $T_C \le T_{ST}$, since reaching a stable target configuration implies that, whatever future interactions may be, the system will always remain in a target configuration. The stabilization time T_{ST} can, however, be strictly larger than the convergence time T_C.

As bounds on the convergence and stabilization time are given in probabilistic terms, one often additionally emphasizes whether a protocol is guaranteed to, eventually, reach a stable target configuration (i.e., whether $T_{ST} < \infty$ holds with probability 1). Such protocols are called *exact* or *always correct*.

The newer results on population protocols, for example [3,18], tend to consider the stabilization time for exact protocols. However, from a practical point of view, con-

[4] The notions as defined here are the ones used predominantly in population protocols in recent literature. However, note that some previous publications (e.g., [2,13]) refer to stabilization time as convergence time.

vergence may provide similarly strong runtime guarantees while enabling more efficient protocols. Indeed, our Theorem 5 shows that the lower bound on the number of states required by any majority protocol that *stabilizes* in $n^{2-\Omega(1)}$ expected interactions does not apply if one considers *convergence* instead.

In the remainder of this article, we define $\mathbb{N}$ as the set of natural numbers without zero and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$.

3 Auxiliary population protocols

In this section we introduce a few auxiliary population protocols that we use as subroutines. These protocols, or variants of them, are well known and have been used in other work on population protocols, as indicated below.

We start with two comparatively simple primitives: *One-way Epidemic* and *Load Balancing*. Afterward we proceed to describe two more involved protocols, one for the creation of a *junta* (Sect. 3.1) and one for the creation of a *phase clock* (Sect. 3.2), both of which require slight adaptions and rephrasing to fit into our setting.

One-way Epidemic A *one-way epidemic* for n nodes is a population protocol with state space $\{0, 1\}$ and transition function $(x, y) \mapsto (x, \max\{x, y\})$. Nodes with value 0 are referred to as *susceptible* and nodes with value 1 as *infected*. We define the *infection time* T_{INF} as the number of interactions required by a one-way epidemic starting with a single infected node to infect the whole population. The following upper and lower high-probability bounds on T_{INF} have been shown in [9].

Lemma 1 *([9, Lemma 2]) For any constant $a > 0$ there exist constants $c_1, c_2 > 0$ such that we have the inequality $Pr[c_1 \cdot n \log n \leq T_{\text{INF}} \leq c_2 \cdot n \log n] \geq 1 - n^{-a}$.*

Load balancing We define a simple population protocol for *load balancing* over n nodes. The state space is $\{-\Lambda, -(\Lambda - 1), \dots, \Lambda - 1, \Lambda\}$, where $\Lambda \in \mathbb{N}$ is a positive integer (which may depend on n). We say a node in state x has *load* x. The transition function is $(x, y) \mapsto \left(\lceil \frac{x+y}{2} \rceil, \lfloor \frac{x+y}{2} \rfloor\right)$. Let $\Delta(t)$ denote the *discrepancy* after t interactions, which is the difference between the maximum and minimum load among all nodes, and set $\Delta := \Delta(0)$. We define the *load balancing time* T_{LB} as the number of interactions required to reduce the initial discrepancy to at most 2. The following upper high-probability bound on T_{LB} has been shown in [12].

Lemma 2 *([12, from Theorem 1]) For any constant $a > 0$, there exists a constant $c > 0$ such that we have the inequality $Pr[T_{\text{LB}} \leq c \cdot n \log(n \cdot \Delta)] \geq 1 - n^{-a}$.*

3.1 Junta

The next protocol rapidly elects a non-empty junta of size at most $n^{1-\Omega(1)}$. It is a variant of a protocol from [18], where each node calculates a *level* from a range of $\Theta(\log \log n)$ values and all nodes with the highest level form the junta. The original protocol is modified such that we can not only provide an upper bound on the highest level reached by any node (as in [18]) but also a lower bound. This change also simplifies the protocol and allows the nodes to realize when the junta selection has finished. Thus, in contrast to [18], nodes are not required to store their level ad infinitum. This is important when using the junta selection as a subprotocol, as storing the level would then increase the number of states per node by a *factor* of $\Theta(\log \log n)$.

We first describe in Sect. 3.1.1 how the levels are calculated. Then we continue to describe how this level calculation can be used to calculate a junta with the desired properties and state the main result for the junta election process in Sect. 3.1.2. The protocol's analysis is given in Sects. 3.1.3 and 3.1.4.

3.1.1 Level calculation

For the level calculation, the state of a node is a tuple of the form (l, a), where the *level* $l \in \mathbb{N}_0$ is a counter and the *activity* bit $a \in \{0, 1\}$ indicates whether a node is active or not. Initially, all nodes have state $(l, a) = (0, 1)$. That is, they are at level 0 and active.

To describe the transition function, we distinguish between a node's first interaction and any of its subsequent interactions. During its first interaction, a node u adopts state $(1, 1)$ if it is the initiator and state $(0, 0)$ if it is the responder. Since the interacting nodes are chosen randomly, this simulates a fair coin toss to decide whether the node should remain active and move up to level 1, or whether it should become inactive.

During any following interaction, u changes its state only if it is still active ($a = 1$) and if it is the initiator of the interaction. In this case, when interacting with a responder in state (l', a'), node u updates its state as follows:

$$\left[(l, 1), (l', a')\right] \mapsto \begin{cases} (l+1, 1) & \text{if } l' \geq l \text{ and} \\ (l, 0) & \text{otherwise.} \end{cases} \tag{1}$$

In words, a node remains active and increases its level as long as it does not encounter a node with a lower level. The only difference to the protocol from [18] is how nodes behave in their first interaction, which allows us to provide a lower bound on the maximum level reached by any node (Lemma 4). We use the random variable L^* to denote this maximum level. Moreover, for $l \in \mathbb{N}_0$ we define B_l as the number of nodes that reach level at least l before becoming inactive.

3.1.2 Junta calculation

We now describe how the above level calculation can be used to calculate a suitable junta. In addition to the level l and activity bit a, each node stores a *marker bit* $b \in \{0, 1\}$ that indicates whether the node is a member of the junta ($b = 1$) or not ($b = 0$) and a *finished bit* $f \in \{0, 1\}$ that indicates whether a node knows that there is at least one marked node ($f = 1$) or not ($f = 0$). Initially, all nodes have $b = 0$ and $f = 0$. If two nodes with finished bit 0 interact, they update their levels as described in Sect. 3.1.1. Any node that reaches level $l_{\max} := \lfloor \log \log n \rfloor - 3$ sets its marker bit $b = 1$ and its finished bit $f = 1$. If two nodes interact and at least one of them has its finished bit set to 1, both nodes set their finished bit to 1; no further state updates happen in this case.

We refer to this protocol as FORMJUNTA. An important difference to the junta protocol from [18] is under which circumstances a node is assumed to be part of the junta. While our protocol starts with an empty junta and marks a node as part of the junta when it reaches level $l_{\max}$, the protocol from [18] assumes that a node is in the junta as long as it has not encountered a node with a higher level. In particular, initially the junta from [18] has linear size and decreases gradually over time. Protocols using a junta typically rely on a junta of size at most $n^{1-\Omega(1)}$. Dealing with the initially oversize junta requires some care, a difficulty avoided by our protocol. Another benefit of our protocol is that once a node sets its finished bit, its level value (and activity bit) are no longer of any relevance and need not be stored any longer. These benefits come with the caveat that our protocol may not finish with a non-zero (but, as we will show, negligible) probability. That is, it is possible that no node is ever marked/finished.

The remainder of this section proves the following theorem.

Theorem 1 *Fix any constant $a > 0$ and let n be sufficiently large with respect to a. With probability at least $1 - n^{-a}$, protocol* FORMJUNTA *calculates a non-empty junta (with all nodes being finished) of size at most $n^{0.98}$ within $O(n \log n)$ interactions. It uses $2 \cdot (l_{\max} + 1) = \Theta(\log \log n)$ states per node. Finished nodes are in one of exactly two states, indicating whether the node is in the junta or not.*

Note that our analysis of Theorem 1 is not designed to keep the involved constants small but instead to make the asymptotic analysis as clear as possible. For example, the current, simple asymptotic analysis wold require an exorbitant large value for n ($\geq e^{800}$). These numbers arise simply out of convenient choices and it is not difficult (if tedious) to improve them to more realistic values. In fact, simple experimental simulations show that these protocols work already well in practice for values of $n \geq 10^6$.

3.1.3 Auxiliary claims about the level calculation

In this section we state and prove some auxiliary claims about the level calculation described in Sect. 3.1.1. We start with upper and lower bounds on the number B_1 of nodes that proceed from level 0 to level 1 (Claim 1). Afterward, we provide both upper and lower bounds on the number of nodes that proceed from level l to level $l + 1$ for $l \in \mathbb{N}$ (Claim 2). Finally, we bound how many levels nodes can proceed beyond any level that is reached by at most $O\left(n^{1/3}\right)$ nodes (Claim 3).

Claim 1 Fix any two constants $a, \epsilon > 0$ and let n be sufficiently large with respect to a and ϵ. Then, $\Pr[|B_1 - n/2| < \epsilon \cdot n/2] \geq 1 - n^{-a}$.

Proof For a node u let the *first interaction* t_u of u denote the earliest interaction during which u was either initiator or responder. We say u is a *singleton* if $t_u \neq t_v$ for all nodes $v \neq u$. Two nodes $u \neq v$ with $t_u = t_v$ are called twins. Let $\mathcal{S}$ denote the set of all singletons and $\mathcal{T}$ the set of all nodes that have a twin.

For each node u we define the binary random variable X_u to be 1 if and only if u is the initiator of t_u. Note that $\Pr[X_u = 1] = 1/2$ and that $B_1 = \sum_u X_u$. We would like to treat B_1 as a binomial distribution $\text{Bin}(n, 1/2)$. Unfortunately, the variables X_u are not independent: for twins u and v, exactly one of X_u and X_v is 1. To fix this, define $K \in \{1, 2, \ldots, \lfloor n/2 \rfloor\}$ as the number of pairs u and v that are twins and let us condition on a fixed $K = k$. The $n - 2k$ variables X_u with $u \in \mathcal{S}$ are completely independent of the remaining process (a node becomes initiator or responder independently with probability 1/2). For the $2k$ variables corresponding to twins, note that their sum is exactly k. We pick an arbitrary subset $\mathcal{T}_1 \subseteq \mathcal{T}$ of k twins and define $X'_u := 1$ for all $u \in \mathcal{T}_1$ as well as $X'_u := 0$ for all $u \in \mathcal{T} \setminus \mathcal{T}_1$. For $u \in \mathcal{S}$, we define $X'_u := X_u$. Obviously, we have $B_1 = \sum_u X_u = \sum_u X'_u$ and the set of all X'_u is mutually independent. Moreover, $\mathbb{E}[B_1 | K = k] = k \cdot 1 + k \cdot 0 + (n - 2k)/2 = n/2$. For any constant $b > 0$, Chernoff (Eq. (21)) gives

$$\Pr[|B_1 - n/2| \geq \delta \cdot n/2 | K = k] \leq 2n^{-b},$$

where $\delta := \sqrt{6b \cdot \log(n)/n} = o(1)$. Using the law of total probability to get rid of the conditioning yields the inequality $\Pr[|B_1 - n/2| \leq \epsilon \cdot n/2] \geq 1 - 2n^{-b}$, which implies the claim's statement by choosing the constant $b = a + 1$. □

Claim 2 Fix any two constants $a > 0$ and $\epsilon \in (0, 1]$ and let n be sufficiently large with respect to a and ϵ. For all $l \in \mathbb{N}$, $\xi_U \in [n^{-1/3}, 1)$, and $\xi_L \in [n^{-1/2} \ln n, 1)$ we have

1. $\Pr[B_{l+1} < (1 + \epsilon)\xi_U^2 \cdot n | B_l \leq \xi_U \cdot n] \geq 1 - n^{-a}$ and
2. $\Pr[B_{l+1} > (1 - \epsilon)\xi_L^2 \cdot n/4 | B_l \geq \xi_L \cdot n] \geq 1 - n^{-a}$.

Proof Fix an $l \in \mathbb{N}$ and consider a node u that just reached level l. Node u is still active and will either become inactive or proceed to level $l+1$ during its next interaction. Let t be u's next interaction.

1. The probability for u to proceed to level $l+1$ during interaction t is at most B_l/n. This holds for all B_l nodes that reach level at least l. By a straightforward coupling argument[5], we get that B_{l+1} is stochastically dominated by a binomially distributed random variable $\mathrm{Bin}(B_l, B_l/n)$. Conditioned on $B_l \le \xi \cdot n$ we can apply Chernoff (Equation (17)) to get
$$\begin{aligned}&\Pr[B_{l+1} \ge (1+\epsilon)\cdot \xi^2 \cdot n | B_l \le \xi \cdot n] \\ &\le e^{-\frac{\epsilon^2 \cdot \xi^2 \cdot n}{3}} \le e^{-\frac{\epsilon^2 \cdot n^{1/3}}{3}},\end{aligned} \tag{2}$$
implying the desired statement.
2. If u is among the last $B_l/2$ nodes that try to proceed from level l to level $l+1$, its probability to proceed to level $l+1$ is at least $B_l/(2n)$. By a straightforward coupling argument[6] shows that B_{l+1} stochastically dominates a binomially distributed random variable $\mathrm{Bin}(B_l/2, B_l/(2n))$. Conditioned on $B_l \ge \xi \cdot n$ we can apply Chernoff (Equation (16)) to get
$$\begin{aligned}&\Pr[B_{l+1} \le (1-\epsilon)\cdot \xi^2 \cdot n/4 | B_l \ge \xi \cdot n] \\ &\le e^{-\frac{\epsilon^2 \cdot \xi^2 \cdot n/4}{2}} \le e^{-\frac{\epsilon^2 \cdot (\ln n)^2}{8}},\end{aligned} \tag{3}$$
implying the desired statement. □

Claim 3 Fix any integer constant $a \ge 1$ and let n be sufficiently large. For all $l \in \mathbb{N}$, we have
$$\Pr[B_{l+4a} = 0 | B_l < 2n^{1/3}] \ge 1 - n^{-a}. \tag{4}$$

Proof Note that $B_l < 2n^{1/3}$ implies $B_{l'} \le B_l < 2n^{1/3}$ for all $l' \ge l$. By Markov's inequality, we have
$$\begin{aligned}&\Pr[B_{l'+1} \ge 1 | B_{l'} < 2n^{1/3}] \\ &\le \mathbb{E}[B_{l'+1} | B_{l'} < 2n^{1/3}] \le 4n^{-1/3}.\end{aligned} \tag{5}$$

We apply Markov's inequality to the next $4a$ levels and get $\Pr[B_{l+4a} \ge 1 | B_l < 2n^{1/3}] \le (4n^{-1/3})^{4a} \le n^{-a}$. □

3.1.4 Analysis of the junta calculation

Equipped with the auxiliary claims from Sect. 3.1.3, we continue with the analysis of the junta calculation. First, we bound the time it takes until all nodes become inactive (Lemma 3). Next, we give upper and lower bounds on the maximum level L^* reached by the nodes (Lemma 4) as well as an upper bound on the number $B_{l_{\max}}$ of nodes that reach level $l_{\max}$ (Lemma 5). Finally, the proof of Theorem 1 is given at the end of this section.

Lemma 3 *Fix any constant $a > 0$ and let n be sufficiently large with respect to a. With probability at least $1 - n^{-a}$ all nodes become inactive during the first $(6a+12)\cdot n \ln n$ interactions.*

Proof The probability that a given node does not interact in a given interaction is $1 - 1/n$. Thus, the probability that a given node does not interact at all during the first $c \cdot n \ln n$ interactions is at most $(1-1/n)^{c \cdot n \ln n} \le n^{-c}$ for any $c > 0$. By a union bound, we get that all nodes interacted at least once after the first $c \cdot n \ln n$ interactions with probability at least $1 - n^{-c+1}$. Together with Claim 1 and a union bound, we know that, with probability $1 - 2n^{-c+1}$, there are at least $n/3$ nodes in state $(0,0)$ after $c \cdot n \ln n$ interactions.

From that point on, the probability for any fixed node to become inactive during a given interaction is at least $\frac{1}{3n}$ (the node is chosen as the initiator of the interaction and its communication partner is one of the $n/3$ nodes in state $(0,0)$). Thus, the probability that any fixed node remains active during the next $c \cdot n \ln n$ interactions is at most $(1 - 1/(3n))^{c \cdot n \ln n} \le n^{-c/3}$. By a union bound, all nodes become inactive during the next $c \cdot n \ln n$ interactions with probability at least $1 - n^{-c/3+1}$. Combining, we get that all nodes become inactive within $2c \cdot n \ln n$ interaction with probability at least $1 - 2n^{-c+1} - n^{-c/3+1} \ge 1 - 3n^{-c/3+1}$. We can make this probability to be at least $1 - n^{-a}$ by choosing $c = 3a + 6$. □

Lemma 4 *Fix any constant $a > 0$ and let n be sufficiently large with respect to a. With probability at least $1 - n^{-a}$ we have*
$$\lfloor \log\log n \rfloor - 3 \le L^* \le \log\log n + 4\cdot(a+1). \tag{6}$$

Proof Let $\delta := 1/10$, $\hat{\xi}_0 = \check{\xi}_0 := 1$, and define for $l \in \mathbb{N}$
$$\begin{aligned}&\hat{\xi}_l := (1+\delta)^{2^l - 1} \cdot 2^{-2^{l-1}} \\ \text{and}\quad &\check{\xi}_l := (1-\delta)^{2^l - 1} \cdot 2^{-3 \cdot 2^{l-1} + 2}.\end{aligned} \tag{7}$$

[5] Run the original process and mark all nodes that reach level l. Then run the coupled process and use the same random choices. Proceeding from level l' to $l'+1$ for $l' \in \mathbb{N}_0 \setminus \{l\}$ works as in the original process. However, for a node to proceed from level l to $l+1$ its interaction partner must have been marked in the original process.

[6] Run the original process and let b denote the number of nodes that reach level l. Mark the first $b/2$ nodes that try to proceed from level l to level $l+1$. Then run the coupled process and use the same random choices. Proceeding to the next level works as in the original process, except for the last $b/2$ nodes that try to proceed from level l to level $l+1$: such nodes proceed only if their interaction partner has been marked in the original process.

Note that $\hat{\xi}_l$ and $\check{\xi}_l$ are monotonically decreasing in l and that for $l \in \mathbb{N}_0$ we have $\hat{\xi}_{l+1} = (1+\delta) \cdot \hat{\xi}_l^2$ and $\check{\xi}_{l+1} = (1-\delta) \cdot \check{\xi}_l^2/4$.

For the upper bound on L^*, apply Claim 1 and Claim 2.1, to get that, for any $l \in \mathbb{N}$ with $\hat{\xi}_{l-1} \geq n^{-1/3}$ and for any constant $a > 0$,

$$\Pr[B_l < \hat{\xi}_l \cdot n | B_{l-1} \leq \hat{\xi}_{l-1} \cdot n] \geq 1 - n^{-a-1}. \tag{8}$$

(Note that, since $\hat{\xi}_0 = 1$ and $B_0 = n$, the conditioning is void for $l = 1$.) Since $\hat{\xi}_l < n^{-1/3}$ for $l \geq \log\log n$, we can apply Eq. (8) iteratively to see that there is an $l \leq \log\log n$ such that $\Pr[B_l < n^{2/3}] \geq 1 - l \cdot n^{-a-1}$. Together with another application of Claim 2.1, we get an $l \leq \log\log n + 1$ such that $\Pr[B_l < (1+\delta) \cdot n^{1/3}] \geq 1 - l \cdot n^{-a-1}$. Combined with Claim 3 we get an $l \leq \log\log n + 1 + 4 \cdot (a+1)$ such that $\Pr[B_l = 0] \geq 1 - l \cdot n^{-a-1}$.

For the lower bound on L^*, similarly apply Claim 1 and Claim 2.2 to get that, for any $l \in \mathbb{N}$ with $\check{\xi}_{l-1} \geq n^{-1/3}$ and for any constant $a > 0$,

$$\Pr[B_l > \check{\xi}_l \cdot n | B_{l-1} \geq \check{\xi}_{l-1} \cdot n] \geq 1 - n^{-a-1} \tag{9}$$

(As above, since $\check{\xi}_0 = 1$ and $B_0 = n$, the conditioning is void for $l = 1$.) Since $\check{\xi}_l \geq n^{-1/3}$ for all $l \leq \log\log n - 3$, we can apply Eq. (9) iteratively to see that, for $l = \lfloor \log\log n \rfloor - 3$, $\Pr[B_l > n^{2/3}] \geq 1 - l \cdot n^{-a-1}$.

The lemma's statement follows via a union bound. □

Lemma 5 *Fix any constant $a > 0$ and let n be sufficiently large with respect to a. Then we have the bound $\Pr[B_{l_{\max}} < n^{0.98}] \geq 1 - n^{-a}$.*

Proof Define δ and $\hat{\xi}_l$ as in the proof of Lemma 4. By their definition and since $l_{\max} = \lfloor \log\log n \rfloor - 3$, we have for any $n \in \mathbb{N} \setminus \{1\}$

$$\begin{aligned}
\hat{\xi}_{l_{\max}} &= (1+\delta)^{2^{l_{\max}}-1} \cdot 2^{-2^{l_{\max}-1}} \\
&\leq (1+\delta)^{2^{\log\log n - 4}-1} \cdot 2^{-2^{\log\log n - 4 - 1}} \\
&= \frac{1}{1+\delta} \cdot (1+\delta)^{\log(n)/16} \cdot 2^{-\log(n)/32} \\
&= \frac{1}{1+\delta} \cdot n^{\log(1+\delta)/16} \cdot n^{-1/32} \\
&= \frac{1}{1+\delta} \cdot n^{\frac{2\log(1+\delta)-1}{32}} < n^{-0.02}.
\end{aligned} \tag{10}$$

Let $\epsilon := 1 - 0.02 = 0.98$. Analogously to the proof of Lemma 4, we have for any $l \in \mathbb{N}$ with $\hat{\xi}_{l-1} \geq n^{\epsilon-1}$ and for any constant $a > 0$

$$\Pr[B_l < \hat{\xi}_l \cdot n | B_{l-1} \leq \hat{\xi}_{l-1} \cdot n] \geq 1 - n^{-a-1}. \tag{11}$$

Since $\hat{\xi}_l < n^{\epsilon-1}$ for $l \geq l_{\max}$ (by Eq. (10) and by the monotonicity of $\hat{\xi}_l$), we can apply Eq. (11) iteratively to see that there is an $l \leq l_{\max}$ such that $\Pr[B_l < n^{\epsilon}] \geq 1 - l \cdot n^{-a-1} \geq 1 - l \cdot n^{-a}$. This implies the lemma's statement. □

Proof of Theorem 1 We first prove the bound on the runtime. Lemma 3 states that, with high probability, all nodes become inactive within $O(n \log n)$ interactions. Lemma 4 states that, with high probability, at least one node reaches level $l_{\max}$ and, thus, sets its marked and finished bits. Lemma 5 states that, with high probability, at most $n^{0.98}$ nodes reach level $l_{\max}$. Finally, by Lemma 1 the finished bit (which spreads via a one-way epidemic) is, with high probability, set in all nodes after $O(n \log n)$ additional interactions. A union bound over all these results yields the desired runtime bound.

The number of states per node required for FORMJUNTA is

$$\underset{\text{activity bit}}{\underset{|}{2}} \times \underset{\text{level}}{\underset{|}{l_{\max}}} + \underset{\text{marker bit}}{\underset{|}{2}}. \tag{12}$$

Note that a node's activity bit and level counter become irrelevant once its finished bit is set (which happens at latest when reaching level $l_{\max}$). Thus, when a node's finished bit is set, it leaves the $2l_{\max}$ states that store the activity bit and the levels $0, 1, \ldots, l_{\max} - 1$ and enters one of two states: one indicating that it has finished and has the marker bit not set, and one indicating that it has finished and has the marker bit set. □

3.2 Phase clock

Distributed protocols often benefit from some form of synchronization. Phase clocks [9] are one way to synchronize nodes in a population protocol. The idea is to equip each node with a clock that measures time in (let's say) *hours* consisting of $m \in \mathbb{N}$ *minutes*. These clocks do not run at a consistent speed and are not fully synchronized (a node's clock might run faster during a period in which the node is activated uncharacteristically often). However, the clocks can be set up such that, with high probability, each of the first $\operatorname{poly}(n)$ hours

1. lasts $\Theta(n \log n)$ interactions for each node and
2. all nodes *simultaneously* spend $\Theta(n \log n)$ interactions in each such hour.

We adapt the phase clock implementation from [18] to our needs. Each node has a *phase counter* $p \in \mathbb{N}_0$ that keeps track of the current time in minutes. The value $m \in \mathbb{N}$ represents the number of minutes per hour. Its exact value must be chosen carefully as specified by Lemma 6 and its proof. The time for a node with phase counter p can be expressed as $\lfloor p/m \rfloor$ hours and $p \mod m$ minutes. To limit the number of states per node, we do arithmetic on the phase counter modulo $h \cdot m$ for a parameter $h \in \mathbb{N}$. We use $\textsc{PhaseClock}_h$ to refer to the protocol that uses the parameter h[7] (which may be a constant or grow with n, depending on the protocol using the phase clock). We also allow $h = \infty$, which means that $\textsc{PhaseClock}_h$ uses exact phase counters that may become arbitrarily large.

We continue with a formal description of the phase clock implementation in Sect. 3.2.1. That section also states the key result (Lemma 6) regarding the synchronization properties of $\textsc{PhaseClock}_h$. The protocol description is based on two simplifying assumptions. Section 3.2.2 explains how to get rid of these. To simplify the usage of the phase clock protocol in the description of other population protocols, Sect. 3.2.3 describes an interface and its guarantees (extracted from Lemma 6) to access the phase clock, resulting in this section's main result (Theorem 2).

3.2.1 Phase clock protocol & synchronization

The state of a node is a tuple of the form (p, b). The *phase counter* $p \in \mathbb{N}_0$ indicates the total number of minutes passed. Initially, all nodes have $p = 0$. The *marker bit* $b \in \{0, 1\}$ indicates whether the node is marked ($b = 1$) or not ($b = 0$). We make two simplifying assumptions for the following description:

1. We assume $h = \infty$ (so we describe $\textsc{PhaseClock}_\infty$). In particular, the phase counters are unbounded.
2. We assume that the number of marked nodes lies in the interval $[1, n^{1-\xi}]$ at the start of any interaction. Here, $\xi \in (0, 1]$ is an arbitrary constant. Note that the identity as well as the number of marked nodes is allowed to change arbitrarily from interaction to interaction, as long as the number of marked nodes stays within the mentioned interval.

Section 3.2.2 explains how to get rid of these assumptions.

Consider an interaction between an initiator u with state (p, b) and a responder in state (p', b'). Protocol $\textsc{PhaseClock}_\infty$ causes node u to update its state according to the following transition function (from [18]):

$$\big[(p, b), (p', b')\big] \mapsto \begin{cases} (\max\{p, p' + 1\}, b) & \text{if } b = 1 \text{ and} \\ (\max\{p, p'\}, b) & \text{otherwise.} \end{cases} \tag{13}$$

The responder's state remains unchanged.

[7] Technically, m could also appear as a parameter in the index. However, for all our applications m will be a constant with respect to n. Thus, we omit m in the index and always assume it is chosen suitably according to Lemma 6.

Synchronization Properties. Remember that the m denotes the number of minutes in an hour. We define the *hour* $\mathrm{H}_u(t) \in \mathbb{N}_0$ of node u with phase counter $p(t)$ after t interactions as $\mathrm{H}_u(t) := \lfloor p(t)/m \rfloor$. We say a node reached hour $i \in \mathbb{N}_0$ if its phase counter is at least $i \cdot m$.

We now define the notion of *rounds*, which represents a period of interactions during which all nodes have the same hour. Let $\mathrm{R}_{\mathrm{Start}}(i)$ (*start of round* i) denote the interaction during which the last node reaches hour i. Similarly, let $\mathrm{R}_{\mathrm{End}}(i)$ (*end of round* i) denote the interaction during which the first node reaches hour $i + 1$. If $\mathrm{R}_{\mathrm{Start}}(i) \le \mathrm{R}_{\mathrm{End}}(i)$ (which is not necessarily true), then $\mathrm{R}_{\mathrm{End}}(i) - \mathrm{R}_{\mathrm{Start}}(i)$ equals the number of interactions t for which *all* nodes u have, simultaneously, the same hour $\mathrm{H}_u(t) = i$. Thus, for any $i \in \mathbb{N}_0$ we define the *length of round* i as $\mathrm{R}_{\mathrm{Length}}(i) := \max\{0, \mathrm{R}_{\mathrm{End}}(i) - \mathrm{R}_{\mathrm{Start}}(i)\}$. We also define the *stretch of round* i as $\mathrm{R}_{\mathrm{Stretch}}(i) := \mathrm{R}_{\mathrm{End}}(i) - \mathrm{R}_{\mathrm{End}}(i-1)$. In other words, the stretch of round i denotes the time it takes for the first node to reach hour $i+1$ after the first node reached hour i. In particular, we always have $\mathrm{R}_{\mathrm{Length}}(i) \le \mathrm{R}_{\mathrm{Stretch}}(i)$.

A key property of the above phase clock construction is captured by the following lemma. It states that, by carefully choosing the phase clock parameter m, we can ensure that both the round length and stretch of the first poly(n) many rounds are $\Theta(n \log n)$. It is a reformulation of [18, Theorem 3.1] to fit our setting and proofs. A brief proof based on a technical lemma from [18] is given in "Appendix B".

Lemma 6 *Let $a, c, d_1 > 0$ be constants and assume n to be sufficiently large with respect to them. There is a constant parameter $m \in \mathbb{N}$ (from the definition of $\textsc{PhaseClock}_\infty$) and a constant $d_2 > 0$ such that, with probability at least $1 - n^{-a}$, for all $i \in \{0, 1, \ldots, n^c\}$*

1. $\mathrm{R}_{\mathrm{Length}}(i) \ge d_1 \cdot n \log n$.
2. $\mathrm{R}_{\mathrm{Stretch}}(i) \le d_2 \cdot n \log n$.

Note that in the above lemma, the constant parameter m depends on the involved constants a, c, and d_1. In particular, it increases with the exponent a of the desired error probability.

3.2.2 Fixing the odds and ends

We briefly explain how the simplifying assumptions we made for the protocol description can be avoided.

Computing a junta on the fly Our protocol description in Sect. 3.2.1 assumes that the number of marked nodes lies in the interval $[1, n^{1-\xi}]$ at the start of any interaction, where $\xi \in (0, 1]$ is an arbitrary constant. Instead of assuming a priori the existence of such a junta in each round, we can use protocol $\textsc{FormJunta}$ from Sect. 3.1 to generate such a junta (with $\xi = 0.02$) with high probability within $\mathrm{O}(n \log n)$

interactions using $2 \cdot (\lfloor \log \log n \rfloor - 2)$ states per node (see Theorem 1).

The state of a node is a tuple (l, a, b, f, p). The (sub-) tuple (l, a, b, f) is used as the state for the junta protocol and consists of the level $l \in \{0, 1, \dots, \lfloor \log \log n \rfloor - 3\}$, the activity bit $a \in \{0, 1\}$, the marker bit $b \in \{0, 1\}$, and the finished bit $f \in \{0, 1\}$. The (sub-) tuple (p, b) is used for the phase clock protocol and consists of the phase counter $p \in \mathbb{N}_0$ and the marker bit $b \in \{0, 1\}$. Note that the marker bit b is used by both protocols. All nodes start in state $(0, 1, 0, 0, 0)$ (with only the activity bit set) and execute protocol FORMJUNTA on the (sub-) tuple (l, a, b, f). Once the finished bit f of a node is set it starts to execute the phase clock protocol from Sect. 3.2.1 on the (sub-) tuple (p, b). We make two simple observations:

1. PHASECLOCK$_\infty$ starts only when (if) the first node in FORMJUNTA becomes marked (and, thus, finished). By Theorem 1, this happens with high probability within $O(n \log n)$ interactions. Additionally, since the finished bit spreads via a one-way epidemic, with high probability *all* nodes start to execute PHASECLOCK$_\infty$ in $O(n \log n)$ interactions (by Lemma 1).
2. When PHASECLOCK$_h$ starts, it does so with a junta of size exactly 1. During its execution, the junta might grow (due to more nodes becoming marked in FORMJUNTA). However, by Theorem 1, with high probability the junta does not grow beyond size $n^{0.98}$.

It follows that Lemma 6 holds also if the junta is computed on the fly, with the only difference being that it takes $O(n \log n)$ interactions before the first node starts to increase its phase counter. This yields the following observation.

Observation 1 We can change PHASECLOCK$_\infty$ such that, with high probability, it computes a non-empty junta (marked nodes) of size at most $n^{0.98}$ on the fly and Lemma 6 still holds.

Unbounded state space The population protocol as described in Sect. 3.2.1 requires an unbounded number of states, since a node's phase counter p is unbounded. We can avoid this by performing any arithmetic on the phase counter modulo $h \cdot m$. Here, $h \in \mathbb{N}$ is a parameter that controls how many hours nodes can count exactly (a node reaching hour h cannot tell whether it has hour h or hour 0).

Note that Lemma 6 implies that during the first poly(n) many rounds all nodes are, with high probability, in neighboring hours (otherwise, if there was a time where one node is in hour i and another node is in hour $i + 2$, those nodes could never be simultaneously in hour $i + 1$). Thus, $h = 3$ is already enough to allow a node, with high probability, to distinguish whether its interaction partner is in an earlier, in the same, or in a later hour. We get the following observation.

Observation 2 Assume $h \geq 3$. Define PHASECLOCK$_h$ analogously to PHASECLOCK$_\infty$ (see Eq. (13) but with all arithmetic on the phase counter p done modulo $h \cdot m$. With high probability, all nodes can correctly determine the maximum in the transition function of PHASECLOCK$_h$ (Eq. (13)) during the first n^c rounds, where c is the constant from Lemma 6. In particular, Lemma 6 holds also for PHASECLOCK$_h$.

3.2.3 Phase clock interface

To simplify the usage of the phase clock in our Majority protocols, we provide an interface to PHASECLOCK$_h$, together with the guarantees implied by Lemma 6. The parameter $h \in \mathbb{N} \cup \{\infty\}$ is assumed to be at least 3. We group the guarantees of the different interface functions in three categories:

(A) The following function calls to PHASECLOCK$_h$ are guaranteed to work as described with probability 1:
   - PHASECLOCK$_h(u, v)$: Update the state of u according to Equation (13).
   - `PCmarked`(u): Return TRUE iff u's marker bit b is set (meaning u is a junta node).
   - `PCfinishedJunta`(u): Return TRUE iff u's finished bit f is set.
   - `PCoverflowed`(u): Return TRUE iff, in the past, the phase counter of u decreased at least once in absolute value (due to the modulo $h \cdot m$ arithmetic).
   - `PCnewHour`(u): Return TRUE iff u reached a new hour the last time it updated the phase counter.
   - `PCskippedHour`(u): Return TRUE iff there was an interaction during which the hour of node u increased by at least 2 (this happens if the clocks are not properly synchronized).

(B) The following function calls to PHASECLOCK$_h$ are guaranteed to work as described for n^c many rounds with probability $1 - n^{-a}$ for any constants $a, c > 0$ (assuming m was chosen suitably and n is sufficiently large):
   - `PCdifferentHour`(u, v): Return TRUE iff u is currently in a different hour as v.
   - `PCsameHour`(u, v): Return TRUE iff u is currently in the same hour as v.
   - `PCsmallerHour`(u, v): Return TRUE iff u is currently in a smaller hour than v.
   - `PClargerHour`(u, v): Return TRUE iff u is currently in a larger hour than v.

(C) Moreover, until the first node reaches hour h (i.e., for the first $R_{\text{End}}(h - 1)$ many interactions), all function calls work as described with probability 1.

Protocols using the phase clock should be aware that, with negligible probability, the phase clock might not run at all (no nodes were marked) or might run too fast (if $n^{1-o(1)}$ nodes were marked).

We gather the above guarantees in Theorem 2, the main result of this section. In the following, remember that $l_{\max} = \lfloor \log\log n \rfloor - 3$ is the maximum junta level.

Theorem 2 *Let $a, c > 0$ be constants and assume n to be sufficiently large with respect to them. Consider a parameter $h \in \{3, 4, \dots\} \cup \{\infty\}$. PHASECLOCK_h supports the interface specified above with Guarantees (A) to (C) and uses $\Theta(h + \log\log n)$ states per node. A node whose phase clock is running (finished bit from junta creation is set) is in one of $\Theta(h)$ many states.*

Proof of Theorem 2 $\text{PHASECLOCK}_h(\cdot)$, $\texttt{PCmarked}(\cdot)$, as well as $\texttt{PCfinishedJunta}(\cdot)$ are simple state updates and lookups. As such, they are correct by definition. The function calls $\texttt{PCoverflowed}(\cdot)$, $\texttt{PCnewHour}(\cdot)$, and $\texttt{PCskippedHour}(\cdot)$ can be implemented by providing a bit for each of them, which is set to either TRUE or FALSE according to the respective function description (note that the corresponding conditions can be easily checked locally by a node). This implies Guarantee (A).

The statements from Guarantee (B) (which cover the function calls $\texttt{PCdifferentHour}(\cdot)$, $\texttt{PCsameHour}(\cdot)$, $\texttt{PCsmallerHour}(\cdot)$, and $\texttt{PClargerHour}(\cdot)$) are a consequence of the choice $h \geq 3$ and Lemma 6 and Observation 2. These ensure that, with high probability, for $\text{poly}(n)$ rounds, the hours of any pair of nodes differ by at most one.

Before the first node reaches hour h, nodes store their exact phase counter and, thus, know their exact hour. This implies Guarantee (C).

We now bound the number of states each node requires. By Theorem 1, the on the fly creation of the junta requires $2 \cdot (l_{\max} + 1)$ states. Note that, as described in Sect. 3.2.2, the values of a node's phase clock state (marker bit, phase counter, bit for $\texttt{PCoverflowed}(\cdot)$, bit for $\texttt{PCnewHour}(\cdot)$, bit for $\texttt{PCskippedHour}(\cdot)$) only become relevant once its finished bit from the junta creation is set. At that moment, nodes can forget the level from the junta calculation and only need to store whether they are finished and marked or finished and unmarked. Thus, for each of the two value of the marker bit when the node is finished, the maximum number of states that can occur is bounded by $h \cdot m \times 2^3$. So in total, the number of states per node is

$$\underbrace{2 \cdot l_{\max}}_{\substack{\text{junta}\\ \text{calculation}}} + \underbrace{2}_{\text{marked?}} \times \underbrace{h \cdot m}_{\text{phase counter}} \times \underbrace{2^3}_{\substack{\texttt{PCoverflowed}(\cdot)\\ \texttt{PCnewHour}(\cdot)\\ \texttt{PCskippedHour}(\cdot)}}. \tag{14}$$

Since we have $l_{\max} = \Theta(\log\log n)$ and $m = \Theta(1)$, this is $\Theta(h + \log\log n)$. □

4 Simple majority

In this section we present and analyze our first majority protocol, $\text{SIMPLEMAJORITY}_{s,h}$, which works correctly with high probability. It is parameterized by two integer values s and h (the latter value is used for the phase clocks). As many majority protocols, it is based on a variant of the *cancellation* and *doubling* approach (see, e.g., [9]). Here, the general idea is that nodes first perform cancellation (opposite opinions cancel each other out) for $\Theta(n \log n)$ consecutive interactions and then, for another $\Theta(n \log n)$ consecutive interactions, each node with an opinion finds a node whose opinion was canceled and copies its opinion onto that node. Cancellation boost the ratio between the support of majority and minority opinions, while duplication eliminates non-opinionated nodes created during cancellation.

Our protocol uses *cancellation* as described above. However, nodes do not simply create a single copy of their opinion but $s \geq 2$ copies (*load explosion*). These copies are distributed among the nodes via a load balancing mechanism. This approach allows us to reduce the number of required phases. We will prove the following theorem:

Theorem 3 *Let $s \in \mathbb{N} \setminus \{1\}$ and $h \in \mathbb{N} \setminus \{1, 2\}$. Consider the majority problem for n nodes with initial absolute bias $\alpha \in \mathbb{N}$. With high probability, protocol $\text{SIMPLEMAJORITY}_{s,h}$ correctly identifies the majority for all interactions $t = \Omega\left(n \log n \cdot \log_s(n/\alpha)\right)$. It uses $\Theta(hs + \log\log n)$ states per node.*

According to Theorem 3 there is no benefit by choosing $h > 3$. However, our stable protocol presented in Sect. 5 does rely on a larger value of h.

We now describe the protocol's state space and its transition function (see also Algorithm 1). Afterward, we give the proof of Theorem 3.

```
SIMPLEMAJORITY_{s,h}(u, v)
1 PHASECLOCK_h(u, v)                        /* synchronization */
2 if PCnewHour(u) then                      /* load explosion  */
3     load_u ← load_u · s
4 if PCsameHour(u, v) then                  /* load balancing  */
5     (load_u, load_v) ← (⌈(load_u+load_v)/2⌉, ⌊(load_u+load_v)/2⌋)
```

Algorithm 1: Pseudocode for transition function of $\text{SIMPLEMAJORITY}_{s,h}$ (initiator u and responder v)

State space The state of a node u consists of the states required for the PHASECLOCK_h protocol (which subsumes the states of FORMJUNTA, cf. Section 3.2) and a *load value* load_u. The load value load_u represents u's current opinion (sign) and its "magnitude" (absolute value). It is initialized with either $+1$ or -1, depending on u's initial opinion. The output function maps the state of a node to the sign of its load value. Thus, the majority guess of a node u is equal to $\text{sign}(\text{load}_u)$.[8]

For most of the analysis, we assume unbounded, integral load values. In the proof of Theorem 3, we will see that, with high probability, load values will be integers not exceeding $3s$ unless all nodes have already the same positive or negative sign. This allows us to *cap the absolute load values* at $3s$ (i.e., setting a node u's load via the assignment $\text{load}_u \leftarrow x$ to a value x with $|x| \geq 3s$ instead sets $\text{load}_u \leftarrow \text{sign}(x) \cdot 3s$)

[8] The value $\text{sign}(\text{load}_u) = 0$ (i.e., $\text{load}_u = 0$) can be interpreted as an "undecided" opinion. In the proof of Theorem 3 we show that, with high probability, all nodes eventually agree on a non-zero sign value.

while still maintaining the high probability guarantee from Theorem 3.

Transition function Consider an interaction between two nodes u (initiator) and v (responder). The nodes' actions can be divided into three parts: *synchronization*, *load explosion*, and *load balancing*. During the synchronization, the PHASECLOCK$_h$ protocol is triggered with initiator u and responder v to update the states of u's phase clock. During the load explosion, u uses the `PCnewHour`$(\cdot)$ method to check whether this is its first interaction in its current hour. If yes, it multiplies its load by a factor of s. During the load balancing, the nodes use the phase clock's `PCsameHour`$(\cdot)$ method to check whether they are in the same hour and, if so, perform a simple load balancing step by balancing their respective loads as evenly as possible.

The following observation follows from the fact that all phase clock function calls work correctly with probability 1 until the first node reaches hour h (Guarantee (C) in Sect. 3.2.3). In particular, since nodes u and v balance their loads only if `PCsameHour`(u, v) returns TRUE (Line 4 in Algorithm 1), we get:

Observation 3 Whenever two nodes u and v balance their loads in SIMPLEMAJORITY$_{s,h}$ before some node reached hour h, both u and v are guaranteed to be in the same hour.

This observation will be of importance for our stable majority protocol presented in Sect. 5 (which is based on SIMPLEMAJORITY$_{s,h}$).

Total & scaled total load Let $\mathrm{load}_u(t)$ denote the load of node u after t interactions and $\mathrm{expl}_u(t)$ the number of load explosions node u experienced after t interactions. Define the *total load* $\Phi(t)$ and the *scaled total load* $\Psi(t)$ after t interactions as

$$\Phi(t) := \sum_{u\in[n]} \mathrm{load}_u(t) \quad \text{and} \quad \Psi(t) := \sum_{u\in[n]} \frac{\mathrm{load}_u(t)}{s^{\mathrm{expl}_u(t)}}.$$

Observe that $\Psi(0) = \Phi(0)$ is the total initial load. Thus, $\mathrm{sign}(\Psi(0)) = \mathrm{sign}(\Phi(0))$ reflects the initial majority and $|\Psi(0)| = |\Phi(0)|$ equals the initial absolute bias α.

The following lemma will be useful to show that, if SIMPLEMAJORITY$_{s,h}$ works for $\mathrm{O}(\log n)$ rounds as expected (the phase clock runs, is properly synchronized, and the loads balance out), all nodes forever agree on the correct initial majority.

Lemma 7 *Let $t \in \mathbb{N}_0$ and assume that whenever two nodes u and v balance their loads in an interaction $t' \le t$, $\mathrm{expl}_u(t') = \mathrm{expl}_v(t')$. Then $\Psi(t) = \Psi(0)$. If, additionally, for all nodes u and v we have $\mathrm{sign}(load_u(t)) = \mathrm{sign}(load_v(t))$, then all nodes forever agree on the correct initial majority opinion after interaction t.*

Proof The invariant for $\Psi(t)$ follows via a simple induction over t. For the second part, assume all nodes' load values have the same sign after t interactions. Note that no load balancing action can change this, afterward. Moreover, the total scaled load $\Psi(t)$ also has the same sign as each single node. So every node's sign forever equals $\mathrm{sign}(\Psi(t))$. Since the lemma's first part states $\mathrm{sign}(\Psi(t)) = \mathrm{sign}(\Psi(0))$ (the initial majority opinion), this implies that each node's sign forever equals the correct initial majority opinion after interaction t. $\square$

We are now ready to prove this section's main result.

Proof of Theorem 3 For $i \in \mathbb{N}_0$ let T_i denote the last interaction of round i (with $T_i = \infty$ if $\mathrm{R}_{\mathrm{Length}}(i) = 0$). Define $i^* := \lceil \log_s(2n/\alpha) \rceil$. Applying Lemmas 2 and 6, with d_1 from Lemma 6 equal to the constant c from Lemma 2, and using a union bound over the first $i^* + 1 = \mathrm{O}(\log n)$ rounds yields that, with high probability, the following properties hold:

(1) For all $i \in \{0, 1, \ldots, i^*\}$, we have $\mathrm{R}_{\mathrm{Length}}(i) = \Omega(n \log n)$ and $\mathrm{R}_{\mathrm{Stretch}}(i) = \mathrm{O}(n \log n)$ (Lemma 6).
(2) For all $i \in \{0, 1, \ldots, i^*\}$, the loads have discrepancy at most 2 after interaction T_i (Lemma 2).

Note that Property (1) implies that no node skips any hour $i \in \{0, 1, \ldots, i^*\}$: If there were such a node, it had hour $< i$ and met a node in hour $> i$ when it skipped hour i. But then, by definition of a round's length, we have $\mathrm{R}_{\mathrm{Length}}(i) = 0$. This contradicts Property (1).

We now condition on the high probability event that the above properties hold and consider the first T_{i^*} interactions. Because nodes are properly synchronized, the calls to `PCsameHour`$(\cdot)$ (Line 4) correctly indicate whether two nodes are in the same hour or not. Also, since no node skipped an hour, any node in hour i experienced exactly i load explosions. This implies that, whenever two nodes balance their loads during the first T_{i^*} interactions, they experienced the same number of load explosions. Lemma 7 gives $\Psi(T_{i^*}) = \Psi(0)$, and the scaled total load definition gives $|\Psi(T_{i^*})| = |\Phi(T_{i^*})|/s^{i^*} \le \alpha \cdot |\Phi(T_{i^*})|/(2n)$. By using $|\Psi(T_{i^*})| = |\Psi(0)| = \alpha$ this yields $|\Phi(T_{i^*})| \ge 2n$.

Note that if $|\Phi(T_{i^*})| \ge 2n$, the absolute value of the average load is at least 2. Hence, either all nodes have load exactly 2 (or -2), or there is at least one node with load ≥ 3 (≤ -3). In the later case it follows from Property (2) that all other nodes have load at least 1 (at most -1). In both cases, all loads have the same sign after interaction T_{i^*}. Thus, using again Lemma 7, all nodes forever agree on the correct initial majority opinion after interaction T_{i^*}. The runtime bound follows since, by Property (1), the first $i^* + 1 = \mathrm{O}\left(\log_s(n/\alpha)\right)$ rounds have stretch $\mathrm{O}(n \log n)$.

To bound the number of states, observe that - conditioned on the high probability event that the above properties hold -no absolute load value exceeds $2s$ unless all nodes' loads have the same sign. Indeed, if not all loads have the same sign at the end of a round, the discrepancy bound (Property (2)) implies that all loads have absolute load at most 2. After the load explosion in the following round the load will be at most $2s$. This allows us to cap the absolute load values at $3s$ as described at the beginning of this section and, with high probability, the protocol outcome will not change.[9] These load values are combined with the states from PHASECLOCK_h. By Theorem 2, that protocol requires in total $\Theta(h + \log\log n)$ states per node, but only $\Theta(h)$ states per node once the node has finished the junta election process. From that time on, each node needs to store the current phase of the PHASECLOCK_h protocol and the current load value. Thus, $\text{SIMPLEMAJORITY}_{s,h}$ requires $\Theta(hs + \log\log n)$ states per node. □

5 Stable majority

In this section, we present and analyze the protocol STABLEMAJORITY_s, a hybrid majority protocol which stabilizes efficiently. We prove the following theorem:

Theorem 4 *Let $s \in \{2, 3, \ldots, n\}$. Consider the majority problem for n nodes with initial absolute bias $\alpha \in \mathbb{N}$. Protocol STABLEMAJORITY_s is exact and stabilizes with high probability and in expectation in $O(n \log n \cdot \log_s(n/\alpha))$ interactions. It uses $\Theta\left(s \cdot \log_s n\right)$ states per node.*

We now describe the protocol's state space and its transition function (see also Algorithm 2). Afterward, we give the proof of Theorem 4.

```
$\text{STABLEMAJORITY}_s(u, v)$
1 $\text{BACKUPMAJORITY}/(u, v)$

2 if $\mathit{finished}_v$ then $\text{finished}_u \leftarrow \text{TRUE}$
3 if $\mathit{error}_v$   then $\text{error}_u \leftarrow \text{TRUE}$

4 if $\neg\mathit{error}_u \wedge \neg\mathit{finished}_u$ then
5   $\text{SIMPLEMAJORITY}_{s,h}(u, v)$

6 if $\mathtt{PCoverflowed}(u) \vee |\mathit{load}_u| \geq 3s$ then
7   $\text{finished}_u \leftarrow \text{TRUE}$

8 if $\left(\mathit{finished}_u \wedge \mathit{finished}_v \wedge \text{sign}(\mathit{load}_u) \neq \text{sign}(\mathit{load}_v)\right)$
    $\vee\ \mathtt{PCskippedHour}(u)$ then
9   $\text{error}_u \leftarrow \text{TRUE}$
```

Algorithm 2: Pseudocode for transition function of STABLEMAJORITY_s (initiator u and responder v). It calls $\text{SIMPLEMAJORITY}_{s,h}$ with $h := \lceil \log_s(4n) \rceil + 2$

[9] For $\text{SIMPLEMAJORITY}_{s,h}$, we could also cap at $2s$. The cap at $3s$ is used in our stable majority protocol in Sect. 4. Note that, if the load balancing works as expected (discrepancy ≤ 2), any node with load $3s$ can be sure that *all* loads have the same, non-zero sign.

Each node u executes a slow but exact protocol BACKUP-MAJORITY [10] (Line 1) as well as up to h rounds of our fast but possibly incorrect $\text{SIMPLEMAJORITY}_{s,h}$ (Lines 4 to 5), with $h := \lceil \log_s(4n) \rceil + 2$. As output, we use the output of the backup protocol if the phase clock is not yet running (u's phase counter is zero and $\mathtt{PCoverflowed}(u) =$ FALSE) or if u thinks that protocol $\text{SIMPLEMAJORITY}_{s,h}$ failed (an error bit is set). Otherwise, we use the output of $\text{SIMPLEMAJORITY}_{s,h}$.

Node u stops $\text{SIMPLEMAJORITY}_{s,h}$ via a *finished bit* finished_u and checks whether $\text{SIMPLEMAJORITY}_{s,h}$ failed via an *error bit* error_u. Both bits are initially FALSE and are spread via a one-way epidemic (Lines 2 to 3). $\text{SIMPLEMAJORITY}_{s,h}$ is executed only while both bits are FALSE (Lines 4 to 5).

The (first) finished bit is set for one of two reasons (Lines 6 to 7): (i) u reached hour h (i.e., its phase counter overflowed). This marks the end of the first h rounds. Stopping at this point ensures that any load balancing operation happens between two nodes in the same hour (Observation 3). (ii) u has absolute load at least $3s$ after its first[11] interaction in an hour i. Then it had absolute load at least 3 at the end of round $i - 1$. If $\text{SIMPLEMAJORITY}_{s,h}$ managed to balance the loads during round $i - 1$, the load of any other node differs by at most 2. Thus, all nodes have the same sign, which we will show to be correct if no node sets its error bit.

The (first) error bit is also set for one of two reasons (Lines 8 to 9): (i) Two finished nodes whose loads have different signs interact with each other, in which case $\text{SIMPLEMAJORITY}_{s,h}$ obviously failed. (ii) A node skipped an hour. Then it is no longer true that a node in hour i experienced exactly i load explosions. This might cause $\text{SIMPLEMAJORITY}_{s,h}$ to fail, since two nodes that experienced a different number of load explosions might balance their loads.

Since the backup protocol is exact, our protocol is exact if the error bit is set. A major part of the analysis is to show that it is also exact if none of the error bits is set. Moreover, we have to show that, with high probability, no error bit is set and the protocol stabilizes fast.

Proof of Theorem 4 Let us first bound the number of states per node. By Theorem 3, $\text{SIMPLEMAJORITY}_{s,h}$ requires $\Theta(hs + \log\log n) = \Theta\left(s \cdot \log_s n\right)$ states. This is increased by a constant factor from the 4 states for BACKUPMAJORITY and the 4 combinations of the bits finished_u and error_u, yielding the desired bound.

[10] We use the 4-state protocol from [20] for this, which stabilizes in $O\left(n^2 \log n\right)$ interactions in expectation, implying a finite stabilization time and, thus, exactness.

[11] Note that the absolute load of a node u can only increase to $\geq 3s$ because of a load explosion. So when the condition $|\text{load}_u| \geq 3s$ holds for the first time, that node just went through a load explosion and, thus, just entered a new hour.

Next, we prove that StableMajority$_s$ is exact. That is, if T_{ST} denotes the stabilization time of protocol StableMajority$_s$, we show that $T_{\text{ST}} < \infty$ with probability 1. We distinguish three cases:

(i) *The phase clock does not start:* That is, in FormJunta all nodes set their activity bit to 0 before reaching level $l_{\max}$. No node is marked, such that the phase counters cannot increase and `PCoverflowed`(u) always returns false. Then all nodes forever use the output of the backup protocol, which has finite stabilization time. Thus, $T_{\text{ST}} < \infty$ in this case.

(ii) *The phase clock starts and some node sets its error bit:* The error bit is spread via a one-way epidemic (Line 3). Thus, with probability 1 eventually all nodes set their error bit. From then on, they use the output of the backup protocol, yielding again $T_{\text{ST}} < \infty$.

(iii) *The phase clock starts and no node ever sets its error bit:* If the phase clock runs, FormJunta marks at least one node and, eventually, all nodes u set their finished bit finished$_u$: Indeed, nodes with an unset finished bit execute the phase clock (via SimpleMajority$_{s,h}$), such that they have a non-zero probability to increase their phase counter (since there is a marked node, see Sect. 3.2.1). Thus, eventually the phase counter overflows and the finished bit is set (Lines 6 to 7).
Let $T < \infty$ denote the interaction after which all finished bits are set. Since no error bit is ever set, no node ever skips an hour (Lines 8 to 9). Thus, any two nodes that balance their loads are not only in the same hour $i < h$ (as checked by SimpleMajority$_{s,h}$) but also experienced both exactly i load explosions. Moreover, after interaction T the loads no longer change and all nodes have the same load signs (otherwise, eventually two finished nodes of different sign meet and an error bit is set, contradicting the case assumption). Thus, by Lemma 7, all nodes forever agree on the correct initial majority after interaction T, such that $T_{\text{ST}} \leq T < \infty$.

It remains to prove the theorem's runtime bounds. We first show that $T_{\text{ST}} = \mathrm{O}\left(n \log n \cdot \log_s(n/\alpha)\right)$ with high probability.

To this end, let $H^* \in \mathbb{N}$ denote the maximal hour ever reached by any node and for $i \in \{0, 1, \ldots, H^* - 1\}$ let T_i be the last interaction of round i. Define T^* as the first interaction during which some node sets its finished or error bit. By Lines 6 to 9, the *first* finish or error bit is set because of three possible reasons:

(i) a node had load at least $3s$ after its first interaction in an hour (finished bit),
(ii) a node's phase counter overflowed (finished bit), or
(iii) a node skipped an hour (error bit).

In a similar way to the proof of Theorem 3, we first show that, with high probability, Reason (i) applies and that all nodes agree on the correct initial majority after T^* interactions without setting the error bit. At that moment, we might not yet have stabilized, since there's still a non-zero probability for a node to set the error bit because of Reason (iii). But with high probability that won't happen before all nodes set their finished bit by the one-way epidemic (Line 2), after which the error bit cannot be set anymore. We formalize this idea below.

Note that the finished bit is set when a node reaches hour h (its phase counter overflows), so $H^* \leq h$. As in Theorem 3's proof, we apply Lemmas 6 and 2 via a union bound to the first $H^* \leq h = \mathrm{O}(\log n)$ rounds to get, with high probability, the following properties:

(1) $T^* < \infty$ (the phase clock runs and some node sets its error bit or, eventually, its phase counter overflows).
(2) For all $i \in \{0, 1, \ldots, H^* - 1\}$, we have $\mathrm{R}_{\text{Length}}(i) = \Omega(n \log n)$ and $\mathrm{R}_{\text{Stretch}}(i) = \mathrm{O}(n \log n)$ (Lemma 6).
(3) For all $i \in \{0, 1, \ldots, H^*-1\}$, the loads have discrepancy at most 2 after interaction T_i (Lemma 2).

As in the proof of Theorem 3, Property (2) implies that no node ever skips an hour.

In the remainder we condition on the high probability event that the above properties hold. Since no node ever skips an hour, whenever two nodes in hour $i \in \{0, 1, \ldots, H^* - 1\}$ balance their loads, both of them experienced exactly i load explosions. Thus, Lemma 7 gives $\Psi(T_i) = \Psi(0)$ for all $i \in \{0, 1, \ldots, H^*-1\}$. With this, we can show that $H^*-1 \leq \lceil \log_s(4n/\alpha) \rceil =: i^*$: Indeed, otherwise all nodes go through round i^* and a similar calculation as in Theorem 3's proof yields $|\Phi(T_{i^*})| = |\Psi(T_{i^*})| \cdot s^{i^*} = |\Psi(0)| \cdot s^{i^*} = \alpha \cdot s^{i^*} \geq 4n$. By an average argument as in Theorem 3's proof, all nodes have absolute load ≥ 3 after interaction T_{i^*}. This implies that any node reaching hour $i^* + 1 < H^*$ has absolute load $\geq 3s$ after the load explosion and sets its finished bit, contradicting H^*'s choice ($i^* + 1$ would be the maximal hour).

Thus, we have $H^* - 1 \leq i^* \leq h - 2$. Let u^* denote the initiator of interaction T^* and remember the three possible reasons why u^* could have set its finished or error bit (Reasons (i) to (iii)). Reason (iii) does not apply since no node skipped any hour. Reason (ii) does not apply since the maximal hour is $H^* \leq i^* + 1 \leq h - 1$, so no node's phase counter overflows. Thus, u^* set its finished bit because of Reason (i): it had absolute load at least $3s$ after its first interaction in hour H^*. Then u^* had absolute load at least 3 after T_{H^*-1} interactions (the end of round $H^* - 1$). Together with Property (3),, either all nodes had load at least 1 or all nodes had load at most -1 after T_{H^*-1} interactions. In particular, all nodes have the same sign, which cannot change subsequently. Since we already saw that load balancing happens only between nodes that experienced the same number of load explosions, Lemma 7 implies that the nodes' sign is also the initial majority.

In summary, with high probability, after T_{H^*-1} interactions, no error bit is set and all nodes *forever* have the correct load sign. This is still the case after interaction T^*. Note that this does not imply $T_{\text{ST}} \le T^*$, since so far only one node finished and there is still a non-zero probability that some node skips an hour and, thus, sets the error bit after interaction T^*. However, with high probability, the finished bit spreads to all nodes within $T_{\text{INF}} = \mathrm{O}(n \log n)$ interactions (the infection time, see Lemma 1). Thus, by using Lemma 6 with a large enough constant d_1, we can ensure that, with high probability, $\mathrm{R}_{\text{Length}}(H^*) \ge T_{\text{INF}}$, such that no node skips an hour before all finished bits are set. Once all nodes are finished, the error bit cannot be set anymore, since all nodes have the same sign.

Combining everything above via a union bound, this yields that, with high probability, $T_{\text{ST}} \le T^* + T_{\text{INF}} = H^* \cdot \mathrm{O}(n \log n) + T_{\text{INF}} = \mathrm{O}\left(n \log n \cdot \log_s(n/\alpha)\right)$, yielding the desired high-probability bound on the stabilization time.

Finally, we show that the stabilization time T_{ST} of StableMajority$_s$ is $\mathrm{O}\left(n \log n \cdot \log_s(n/\alpha)\right)$ in expectation. To this end, observe that we know that, with high probability, the stabilization time T_{ST} is $\mathrm{O}(n \log n \cdot \log_s(n/\alpha))$. That is, for any constant $a > 0$, there is a constant $C > 0$ and appropriate values of the constant protocol parameters such that

$$\Pr[T_{\text{ST}} \le C \cdot n \log n \cdot \log_s(n/\alpha)] \ge 1 - n^{-a}. \tag{15}$$

To show that T_{ST} is $\mathrm{O}\left(n \log n \cdot \log_s(n/\alpha)\right)$ in expectation, we show the following statement: For some fixed η (independent of the constant parameters of protocol StableMajority$_s$), for each sufficiently large n, and for each configuration $\mathcal{C}$ reachable from the initial configuration, the protocol stabilizes from $\mathcal{C}$ within n^η interactions in expectation. Once this is shown, we can calculate

$$\begin{aligned}\mathbb{E}[T_{\text{ST}}] \le\ & \mathbb{E}[T_{\text{ST}} | T_{\text{ST}} \le C \cdot n \log n \cdot \log_s(n/\alpha)] \\ & + n^{-a} \cdot \mathbb{E}[T_{\text{ST}} | T_{\text{ST}} > C \cdot n \log n \cdot \log_s(n/\alpha)] \\ \le\ & C \cdot n \log n \cdot \log_s(n/\alpha) \\ & + n^{-a} \cdot \left(C \cdot n \log n \cdot \log_s(n/\alpha) + n^\eta\right) \\ \le\ & 2C \cdot n \log n \cdot \log_s(n/\alpha).\end{aligned}$$

The first inequality above follows from Equation (15). The second inequality follows from the bound n^η on the expected stabilization time from the configuration $\mathcal{C}$ reached after the first $C \cdot n \log n \cdot \log_s(n/\alpha)$ interactions. Finally, the last inequality holds by taking $a = \eta$.

We use the following facts about the expected running time of some basic protocols:

1. Protocol BackupMajority stabilizes within $\mathrm{O}(n^2 \cdot \log n) \le n^3$ (for sufficiently large n) interactions in expectation [20].
2. The one-way epidemic completes within $\mathrm{O}(n \log n)$ interactions in expectation.
3. For each $K \ge 1$, the number of interactions required so that each node is the initiator of at least K interactions is $\mathrm{O}(K n \log n)$ in expectation (a simple consequence from the expected completion time $\mathrm{O}(n \log n)$ of the coupon collector problem).

Let $\mathcal{C}$ be any non-stable configuration of the protocol StableMajority$_s$ that is reachable from the initial configuration. We distinguish several configuration types with respect to $\mathcal{C}$:

1. *The phase clocks of all nodes have reached their limit of $h \cdot m$, viewing the clocks (for the purpose of this analysis but without actually altering anything in the protocol) as if there were kept running until reaching the limit:* Configuration $\mathcal{C}$ is not stable, so either there is a node with the error flag raised, or all nodes have their finished flag raised but not all nodes have the same signs of their load. Otherwise the configuration is stable.
   In the former case (when there is a node with an error), one instance of one-way epidemic raises the error flag in all nodes in expected $\mathrm{O}(n \log n)$ interactions, and then protocol StableMajority$_s$ stabilizes within additional expected n^3 interactions by completing BackupMajority.
   Similarly, in the latter case, two consecutive one-way epidemics (the first one to make two nodes with different load signs meet and the second one to spread out the information about the error) and then the completion of the BackupMajority protocol are sufficient to stabilize StableMajority$_s$.
   In both cases, the protocol StableMajority$_s$ stabilizes within additional $\mathrm{O}\left(n^3\right)$ interactions in expectation.
2. *There is at least one marked node (by protocol* FormJunta*), but there is still at least one node whose clock has not yet reached the limit of $h \cdot m$ (as above, we view the clocks as if running until the limit):* The marked node with the largest clock value increases its clock within one instance of one-way epidemics. Thus within at most $h \cdot m$ consecutive instances of one-way epidemics, one marked node reaches the clock limit. One additional one-way epidemic makes all clocks reach the limit. This takes $(h \cdot m + 1) \cdot \mathrm{O}(n \log n) = \mathrm{O}(n \log n \cdot \log_s(n))$ interactions in expectation and takes us to a configuration of Type 1.
3. *No node is marked (by protocol* FormJunta*):* We consider two sub-cases:

(a) *All nodes in* FormJunta *are inactive:* No node is ever marked and the phase clock never starts. Thus, StableMajority$_s$ stabilizes in n^3 additional interactions in expectation (via BackupMajority).

(b) *There is at least one active node in* FormJunta*:* If an active node is the initiator of an interaction, then it either increases its junta level or becomes inactive. Thus, when this node initiated $l_{\max}$ interactions, either it reached level $l_{\max}$ and got marked, or it has become inactive. So within $l_{\max} \cdot \mathrm{O}(n \log n) = \mathrm{O}(n \log n \cdot \log \log n)$ interactions in expectation (after each node initiated at least $l_{\max}$ interactions) we either reach a configuration with the first marked node (Type 2) or a configuration with no marked node but only inactive nodes (Type 3a).

In summary, we see that StableMajority$_s$ stabilizes from any configuration $\mathcal{C}$ reachable from the initial configuration within $\mathrm{O}\left(n^3\right) \leq n^4$ (for sufficiently large n) interactions in expectation. □

6 Convergent majority

In this section, we present and analyze the protocol ConvergentMajority$_s$, a hybrid majority protocol which converges efficiently. The main idea of the protocol is that all nodes execute SimpleMajority$_{s,3}$, which converges quickly. However, there is a positive probability that it returns the wrong answer without detecting the error. Therefore, every node switches its output to the backup protocol after a (polynomially) long time. To determine that this time has passed, we use a simple approach based on counting the number of consecutive interactions with junta nodes. Formally, we prove the following theorem:

Theorem 5 *Let* $s \in \{2, 3, \ldots, n\}$. *Consider the majority problem for n nodes with initial absolute bias* $\alpha \in \mathbb{N}$. *Protocol* ConvergentMajority$_s$ *is exact and converges with high probability and in expectation in* $\mathrm{O}(n \log n \cdot \log_s(n/\alpha))$ *interactions. The protocol uses* $\Theta(s + \log \log n)$ *states per node.*

We now describe the protocol's state space and its transition function (see also Algorithm 3). Afterward, we give the proof of Theorem 5.

Nodes first execute a backup protocol BackupMajority .[12] Additionally, each node u executes protocol SimpleMajority$_{s,3}$ as long as it did not encounter 600 marked nodes in a row. The number of such encounters is

```
ConvergentMajority_s(u, v)
1 BackupMajority/(u, v)
2 if $count_u < 600$ then
3   if PCmarked(v) then
4     $count_u \leftarrow count_u + 1$
5   else
6     $count_u \leftarrow 0$
7   SimpleMajority$_{s,3}(u, v)$
```

Algorithm 3: Pseudocode for transition function of ConvergentMajority$_s$ (initiator u and responder v)

stored in a counter value $\text{count}_u \in \{0, 1, \ldots, 600\}$. The value 600 is chosen merely for convenience and has no special meaning. It simply ensures that it takes a long time before a node permanently changes its output to that of the backup protocol (see next paragraph).

The output function maps the state of a node u to a majority guess as follows: Use the output of the backup protocol if the phase counter of the phase clock is zero or if the counter count_u has reached 600. Otherwise, use the output of protocol SimpleMajority$_{s,3}$. Switching eventually to the backup protocol's solution ensures that - even if SimpleMajority$_{s,3}$ fails - the protocol is exact. Using the output of SimpleMajority$_{s,3}$ in between (and switching to the backup protocol's solution only after a long time, when it is correct with high probability) implies that, with high probability, convergence (but not stability) is achieved fast.

Proof of Theorem 5 We first show that our protocol ConvergentMajority$_s$ is exact. This follows easily by considering the following two cases:

1. The phase clock never starts (no node is selected into the underlying junta). Since in this case all counters remain zero forever, the output of the protocol ConvergentMajority$_s$ equals the output of BackupMajority, which is exact.
2. The phase clock starts (meaning the junta is not empty). The probability for a node u to increase its count_u in the next interaction is at least $1/n^2$ (u initiates the interaction with a marked node as responder). This happens 600 times in a row with probability at least $1/n^{1200} > 0$ (a crude but sufficient bound). Thus, eventually all nodes u reach $\text{count}_u = 600$. From that point on the output of ConvergentMajority$_s$ equals that of BackupMajority, which is exact.

This shows that protocol ConvergentMajority$_s$ is exact. The bound on the number of states per nodes follows also easily: Since BackupMajority requires only four states and the counters are bounded by the constant 600, the number of states per node is a constant factor times the number of states required by SimpleMajority$_{s,3}$, which is $\Theta\left(s + \log \log n\right)$.

[12] As before, in Sect. 5, we use the 4-state protocol from [20] for this, which stabilizes in $\mathrm{O}\left(n^2 \log n\right)$ interactions in expectation, implying a finite stabilization time and, thus, exactness.

It remains to prove the convergence time bound for the ConvergentMajority$_s$ protocol. Assuming a non-empty junta (which, by Theorem 1, holds with high probability after $\mathrm{O}(n \log n)$ interactions), we can derive the desired bound from the following observations.

(i) Once the junta is established, the output of protocol ConvergentMajority$_s$ during the next $\mathrm{poly}_1(n)$ interactions equals, with high probability, that of SimpleMajority$_{s,3}$. That protocol converges with high probability in at most $\mathrm{O}\left(n \log n \cdot \log_s(n/\alpha)\right)$ interactions.

(ii) After $\mathrm{poly}_1(n)$ interactions, the nodes' outputs start switching gradually to the output of BackupMajority, which stabilizes in $\mathrm{O}\left(n^2 \log n\right)$ interactions. After $\mathrm{poly}_2(n) > \mathrm{poly}_1(n)$ interactions, with high probability all nodes have switched their output to BackupMajority.

By choosing parameters such that $\mathrm{poly}_1(n) = n^3$, the switch to the backup protocol happens only when it has, with high probability, stabilized. Thus, with high probability, the subprotocol SimpleMajority$_{s,3}$ converges to the correct outcome within $\mathrm{O}\left(n \log n \cdot \log_s(n/\alpha)\right)$ interactions and, when the nodes start switching their output to BackupMajority, that subprotocol also has the correct output. Together, this implies the desired bound on the convergence time.

To see Observation (i), note that, once there is a non-empty junta of size at most $n^{0.98}$ (which happens with high probability in $\mathrm{O}(n \log n)$ interactions by Theorem 1), the probability that a node samples a junta node 600 times in a row is at most $(n^{0.98}/n)^{600} = n^{-12}$. Using a union bound, with high probability no node reaches counter value 600 (and switches to the backup protocol) before $\Theta\left(n^3\right)$ interactions. Observation (ii) follows by a simple Markov bound applied to the expected number of interactions a node requires to switch its output back to the backup protocol (which is upper bounded by $\mathrm{O}\left(n^{600}\right)$) together with a union bound over all nodes.

It remains to prove the bound on the expected convergence time T_{C} of ConvergentMajority$_s$. For this, using the same argument as in the proof of Theorem 4, it is sufficient to show the following statement: For some fixed η, for each sufficiently large n, and for each configuration $\mathcal{C}$ reachable from the initial configuration, the protocol stabilizes from $\mathcal{C}$ within n^η interactions in expectation. Once this is shown, the same calculation as for Theorem 4 yields $\mathbb{E}[T_{\mathrm{C}}] \leq 2C \cdot n \log n \cdot \log_s(n/\alpha)$. To this end, we proceed as in the proof of Theorem 4 and distinguish the following configuration types:

1. *All nodes have switched to the backup protocol:* Then ConvergentMajority$_s$ stabilizes in n^3 additional interactions in expectation (via BackupMajority).
2. *Some node has not yet switched to the backup protocol:* The time until all nodes switch to the backup protocol is dominated by the sum of n geometrically distributed random variables with parameter $\geq 1/n^{1200}$ (see the exactness proof above). Thus, there is a constant η such that all nodes switch their output to BackupMajority after at most $n^{\eta-1}$ many interactions in expectation. This takes us to a configurations of Type 1.

In summary, wee see that ConvergentMajority$_s$ converges from any configuration $\mathcal{C}$ reachable from the initial configuration within $n^{\eta-1} + n^3 \leq n^\eta$ (for sufficiently large n) interactions in expectation. □

7 A note on uniformity

Uniformity in population protocols means that a single algorithm is designed to work for populations of any size. In particular, nodes have no information on the population size n. Protocols where nodes are restricted to a constant number of states are always uniform. But most of the newer protocols allow for a super-constant number of states and use some upper bounds on n, so they are not uniform. In particular, protocols that stop their computation once a counter reaches a value of $\mathrm{polylog}(n)$ fall into this category of non-uniform protocols. This section presents a uniform population protocol for majority.

Uniform population model For studying *uniform* population protocols whose state requirements increase with the population size n, the original model—which considers nodes as finite-state machines (FSM), see Sect. 2 - turns out to be inadequate. Indeed, if each node is an FSM with a state space of size $f(n)$ for a non-trivial function f, then the nodes and, thus, the protocol inherently depend on n and cannot be simply "deployed" in a population of different size.

Doty and Eftekhari [14] introduce a generalized population model that is better suited for this scenario and which we adopt in the remainder of this section . In their model, each node is represented by a 2-tape deterministic Turing machine (TM). We assume that both tapes are infinite to the left and right and that the origin is marked by a special *origin symbol*. Tape 1 (read-only) is called the *input tape* and tape 2 (read-write) the *working tape*. One two-way infinite working tape is sufficient for us since it allows maintaining two unbounded variables, as required in our protocol. For protocols with more unbounded variables, similarly to [15] one can use a TM with as many (one-sided infinite) input/working tapes as there are unbounded variables (whose number must not depend on n).

At the beginning of any interaction, a node's working tape is identical to its working tape at the end of the previous interaction. Whenever two nodes interact, they copy each other's working tape onto their own input tape and restart their TM by entering a start TM-state (which then computes a new state, updates the node's working tape, and halts). We define the number of states used during a protocol execution as $|\Sigma|^s$, where Σ is the (binary) tape alphabet and s is the maximum number of tape cells written by any node during the execution.

Having the above formal model in mind, our description sticks with the standard population protocol terminology. In particular, we assume a suitable encoding of the nodes' states using the alphabet Σ and simply identify the content of a node's working tape with its state. An important implication of the model is that nodes might now use an unbounded number of states (write an unbounded number of cells on the working tape). However, in our uniform majority protocol, the number of used states is finite with probability 1 and $O(\log n \cdot \log\log n)$ in the population size n with high probability.

Uniform majority One of the rare examples of a uniform protocol whose state requirements increase with n is the junta protocol from [18]; we refer to it as FORMJUNTAUNIFORM. Observe that our protocol FORMJUNTA is not uniform, as nodes need to know $l_{\max} = \lfloor \log\log n \rfloor - 3$ in order to mark themselves (see Sect. 3.1). See below for a brief description of FORMJUNTAUNIFORM.

Since our majority protocols from the previous sections use the non-uniform junta FORMJUNTA, none of them is uniform. In fact, to the best of our knowledge, until now there was no exact, uniform majority protocol that would stabilize with high probability in $n^{2-\Omega(1)}$ interactions. The following theorem shows that we get such a uniform majority protocol by applying slight modifications to protocol STABLEMAJORITY$_s$.

Theorem 6 *Let $s \in \mathbb{N} \setminus \{1\}$ be a constant. Consider the majority problem for n nodes with initial absolute bias $\alpha \in \mathbb{N}$. Protocol* UNIFORMMAJORITY$_s$ *is an exact and uniform variant of* STABLEMAJORITY$_s$. *With high probability and in expectation, it stabilizes in $O(n \log n \cdot \log_s(n/\alpha))$ interactions. While the number of used states can be arbitrarily high with non-zero probability, with high probability it uses only $s \cdot O(\log_s(n/\alpha) \cdot \log\log n)$ states per node.*

Note that, in order for UNIFORMMAJORITY$_s$ to be uniform, the parameter s must be constant (i.e., s may not depend on n).

Protocol UNIFORMMAJORITY$_s$ is identical to protocol STABLEMAJORITY$_s$ with the following changes:

1. It uses subprotocol SIMPLEMAJORITY$_{s,\infty}$ instead of SIMPLEMAJORITY$_{s,\lceil \log_s(4n) \rceil + 2}$. In particular, the phase clock PHASECLOCK$_\infty$ is used, which cannot overflow. Thus, nodes always know their exact hour.
2. The phase clock uses FORMJUNTAUNIFORM instead of FORMJUNTA.

Using the original junta instead of ours has the drawback that nodes must remember their level from the junta calculation indefinitely. However, since protocol FORMJUNTA is inherently non-uniform, this seems unavoidable when aiming for a uniform protocol.

For the sake of completeness, we give a brief description of FORMJUNTAUNIFORM, using slightly different wording and notation than in [18], in order to fit it into our framework.[13] We also describe how the phase clock is adapted to the changed junta protocol. Afterward, we give the proof of Theorem 6.

Description of UNIFORMMAJORITY$_s$. As our junta protocol FORMJUNTA, protocol FORMJUNTAUNIFORM is based on the level calculation described in Sect. 3.1.1. Recall that the level calculation uses a level l, an activity bit a, and the transition function described by Equation (1). In addition to the level l and activity bit a, each node stores a *marker bit* $b \in \{0, 1\}$ (indicating whether the node assumes to be in the junta or not) and a *defeated bit* $d \in \{0, 1\}$. Initially, all nodes have $b = 0$ and $d = 0$. A node that just became inactive at a level $l \geq 1$ sets b to 1. If an inactive node at level $l \geq 1$ encounters a node on a higher level, it becomes *defeated*: it sets d to 1, b to 0, and will from now on simply adopt the larger level during any interaction (not changing any of its other state values related to the junta). If encountered by another node in the level calculation, a defeated node is treated as if it were in state $(0, 0)$, independent of its actual level counter l.

Phase Clocks on different Levels & Reset. Compared to FORMJUNTA, any (inactive) node starts with the belief of being in the junta until it becomes defeated by a node from a higher level. This ensures that the junta is never empty. However, when used in the phase clock protocol, there will be a large number of nodes in the junta for the first few interactions (until lower-level nodes become defeated), causing the phase clock to run too fast.

To avoid problems in the protocol relying on the synchronization of the phase clock, nodes now also use the level (from the junta protocol) in the phase clock protocol. This basically results in multiple phase clocks running on different levels. When a node running a phase clock on level l encounters a node running a phase clock on a higher level l', it resets its phase counter to zero (and - by the junta protocol - updates its level to l'). This *reset* also triggers a reset of the

[13] Technically, the described protocol differs slightly from the one in [18]: The first interaction of a node is slightly changed - as described in Sect. 3.1.2 - in order to enable us to prove a lower bound on the maximum level reached by any node. This property is not required for the uniform protocol, so one could use the original junta protocol from [18].

protocol using the phase clock. In our case this is the majority protocol $\textsc{StableMajority}_s$. For a node u this entails a reset of the bits finished_u and error_u to 0, and a reset of the load value from $\textsc{SimpleMajority}_{s,\infty}$ to ± 1, depending on the original opinion of u. This idea of phase clocks running on different levels and a corresponding reset was first proposed and used in [18] for the case of leader election.

Proof of Theorem 6 First note that the protocol requires no knowledge of n, meaning that it is uniform. The remaining proof is similar to that of Theorem 4. In fact, having unbounded phase counters (which avoid overflows in the phase clock) and the guarantee from $\textsc{FormJuntaUniform}$ that the junta is never empty simplify the argumentation considerably. Also note that the exact phase counters guarantee that any load balancing action is *always* guaranteed to be done only between nodes in the same hour.

Let T_{ST} denote the stabilization time of protocol $\textsc{UniformMajority}_s$. To proof exactness, remember the three cases from the exactness proof of Theorem 4. The first two cases are trivial: Case (i) (phase clock does not start) cannot occur, since we use $\textsc{FormJuntaUniform}$. Case (ii) (phase clock starts and some error bit is set) is identical, since with probability 1 eventually all nodes set their error bit and use the output of the backup protocol. For Case (iii) (phase clock starts and no error bits are ever set) we again first show that all nodes finish with probability 1. However, in Theorem 4 this was proven via the overflowing phase counters, which cannot happen for $h = \infty$. Thus, we use a different argument: For the sake of a contradiction assume no node finishes (if one node finishes, all nodes finish eventually). Since the phase clock runs and no error ever occurs, all nodes reach any hour $i \in \mathbb{N}$. Consider an interaction t when all nodes are in hour at least $\iota := \lceil \log_s(s \cdot 3n/\alpha) \rceil$. Since load balancing is only performed between nodes in the same hour and no node ever skips an hour (or an error would occur), Lemma 7 gives $\Psi(t) = \Psi(0)$. But then, similar to previous arguments, our choice of ι ensures that $|\Phi(t)| \geq s \cdot 3n$. So there would be a node u with absolute load at least $3s$. This yields the desired contradiction, since u would have set its finished bit at the beginning of its current hour. Once we know that all node finish with probability 1 in this case, the exactness follows again as in Case (iii) in the exactness proof of Theorem 4.

To prove that, with high probability, we have $T_{\text{ST}} = \mathrm{O}(n \log n \cdot \log_s(n/\alpha))$, we use the same argumentation as for the corresponding part in the proof of Theorem 4, just slightly simplified since nodes now store their exact phase counters. Basically, we again take a union bound over the first $i^* := \lceil \log_s(4n/\alpha) \rceil$ rounds and get the same three properties as in the proof of Theorem 4: (1) $T^* < \infty$ (as we have shown above for the exactness) for the interaction T^* when the first finish or error bit is set. (2) The first i^* rounds have length $\Omega(n \log n)$ and stretch $\mathrm{O}(n \log n)$. (3) The load discrepancy is at most 2 at the end of each of the first i^* rounds.

With these properties, the remaining argumentation from Theorem 4's proof goes through.

The proof for the bound on the expected stabilization time is also identical, by noting that all nodes complete protocol $\textsc{FormJuntaUniform}$ in expected $\mathrm{O}\big(n \log n\big)$ interactions [18].

For the bound on the number of states, note that replacing the phase clock's junta algorithm $\textsc{FormJunta}$ by $\textsc{FormJuntaUniform}$ increases the required number of states by a *factor* of $\Theta(\log\log n)$ (instead of an *additive* term), since we now need to store the level indefinitely. Now, above we saw that, with high probability, all nodes finish after at most $\mathrm{O}(\log_s(n/\alpha))$ rounds. Thus, with high probability, no phase counter is larger than $\mathrm{O}(\log_s(n/\alpha))$ in absolute value. Finally, there is a factor of $(3s+1) \cdot \Theta(1) = s \cdot \Theta(1)$ for the load values and the bits finished_u and error_u, yielding the desired bound. □

8 Conclusions & future work

We analyzed three similar variants of a population protocol for the majority problem: $\textsc{SimpleMajority}_{s,3}$, $\textsc{ConvergentMajority}_s$, and $\textsc{UniformMajority}_s$. All of them based on the so-called *doubling and cancellation approach*. They feature a parameter s that allows for a trade-off between runtime and memory per node.

A natural open question is to improve the bounds we provide. In particular, for $s = \log\log n$ our protocol $\textsc{StableMajority}_s$ has stabilization time $o(n \cdot (\log n)^2)$ while using $\mathrm{O}(\operatorname{polylog} n)$ states. There is (to the best of our knowledge) one other result that also achieves this guarantee [11]. While it does not feature a trade-off capability, it comes with a better stabilization time. It seems non-trivial but also not impossible to combine our trade-off result with the improved stabilization time. Also, it would be interesting whether it is possible to derive parameterized *lower bounds* in which one can similarly see the effect on the running time of increasing or decreasing the number of states per node.

Another open research question for population protocols deals with the phase clock introduced in [18]. It is unclear whether one can derive a similar phase clock that requires only a constant number of states and still synchronizes the population for a polynomial number of interactions with high probability. If it exists, such a phase clock could be used to devise constant-state (majority) protocols that converge in polylogarithmic time with high probability.

Our results formally show that lower bounds for the stabilization time can be bypassed by considering the convergence time. Unfortunately, there are currently no strong lower bounds regarding the convergence time. As convergence time might be considered the more practical runtime notion, finding such lower bounds and tightening the corresponding upper bounds should be deemed a worthy but challenging task.

Acknowledgements Open Access funding provided by Projekt DEAL. Robert Elsässer and Dominik Kaaser were partially supported by the Austrian Science Fund (FWF) under grant no. P 27613 ("Distributed Voting in Large Networks"). Tomasz Radzik's work was supported by EPSRC grant EP/M005038/1 ("Randomized Algorithms for Computer Networks").

A Probabilistic tools

Lemma 8 *(Chernoff Bounds). Let $n \in \mathbb{N}$ and consider a sequence $(X_i)_{i\in[n]}$ of mutually independent binary random variables. Define $X := \sum_{i\in[n]} X_i$ and let $\mu_U, \mu_L \geq 0$ be such that $\mu_L \leq \mathbb{E}[X] \leq \mu_U$. The following inequalities hold for any $\delta \geq 0$ and $\phi \geq 6\mu_U$:*

$$Pr[X \leq (1-\delta)\cdot\mu_L] \leq e^{-\frac{\delta^2\cdot\mu_L}{2}}, \tag{16}$$

$$Pr[X \geq (1+\delta)\cdot\mu_U] \leq e^{-\frac{\delta^2\cdot\mu_U}{2+\delta}}, \quad \text{and} \tag{17}$$

$$Pr[X \geq \phi] \leq 2^{-\phi}. \tag{18}$$

Let $\mu := \mathbb{E}[X]$. We often use the following simplified Chernoff bounds:

$$\Pr[X \leq (1-\delta)\cdot\mu] \leq n^{-a} \tag{19}$$

$$\Pr[X \geq \max\{13a\cdot\log n, (1+\delta)\cdot\mu\}] \leq n^{-a} \quad \text{and,} \tag{20}$$

where $a \geq 0$ is an arbitrary constant and $\delta := \sqrt{3a\cdot\log(n)/\mu}$. For convenience, we sometimes combine both bounds into

$$\Pr[|X-\mu| \geq \max\{13a\cdot\log n, \delta\cdot\mu\}] \leq 2n^{-a}. \tag{21}$$

B Auxiliary protocols: phase clock

This section shows how Lemma 6 follows from the following technical lemma from [18]. We paraphrase the lemma slightly in order to make the dependencies on the involved constants more explicit.

Lemma 9 *([18, Lemma 3.7]) Let $a, d > 0$ be constants and assume n to be sufficiently large with respect to them. There is a constant $K > 0$ such that the following holds: Let $p_{\max}$ denote the maximum and $p_{\min}$ the minimum phase counter after an interaction $t \in \mathbb{N}$. Assume $p_{\max} - p_{\min} \leq 2K$. With probability at least $1 - n^{-a}$, there is a $t' > t + d\cdot n\log n$ such that:*

1. *t' is the first interaction after which the maximum phase counter is $p_{\max} + K$.*
2. *After interaction t', all nodes have a phase counter value of at least $p_{\max}$.*

With this, we are ready to restate and prove Lemma 6.

Lemma 6 *Let $a, c, d_1 > 0$ be constants and assume n to be sufficiently large with respect to them. There is a constant parameter $m \in \mathbb{N}$ (from the definition of* PhaseClock$_\infty$*) and a constant $d_2 > 0$ such that, with probability at least $1 - n^{-a}$, for all $i \in \{0, 1, \ldots, n^c\}$*

1. $\mathrm{R}_{\mathrm{Length}}(i) \geq d_1 \cdot n\log n$.
2. $\mathrm{R}_{\mathrm{Stretch}}(i) \leq d_2 \cdot n\log n$.

Proof The lower bound on $\mathrm{R}_{\mathrm{Length}}(i)$ follows via an induction over i by applying Lemma 9 with $d = d_1$ and by setting the phase clock parameter m to $3K$. For the upper bound on the stretch, note that the one-way epidemic (cf. Lemma 1) implies that, with high probability, the maximum phase counter increases within $\mathrm{O}(n\log n)$ rounds (when a marked node finally sees the maximum phase counter). Thus, with high probability, it takes at most $m\cdot\mathrm{O}(n\log n) = \mathrm{O}(n\log n)$ interactions for a node to leave a given round. □

References

1. Alistarh, D., Gelashvili, R., Vojnovic, M.: Fast and exact majority in population protocols. In: Georgiou, C., Spirakis, P.G. (eds.) Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 –23, 2015, pp. 47–56. ACM (2015). https://doi.org/10.1145/2767386.2767429
2. Alistarh, D., Aspnes, J., Eisenstat, D., Gelashvili, R., Rivest, R.L.: Time-space trade-offs in population protocols. In: Klein, P.N. (ed.) Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19, pp. 2560–2579. SIAM (2017). https://doi.org/10.1137/1.9781611974782.169
3. Alistarh, D., Aspnes, J., Gelashvili, R.: Space-optimal majority in population protocols. In: Czumaj, A. (ed.) *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7–10,*

2018, pp. 2221–2239. SIAM (2018). https://doi.org/10.1137/1.9781611975031.144
4. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. In: Chaudhuri, S., Kutten, S. (eds.) Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25–28, 2004, pp. 290–299. ACM (2004). https://doi.org/10.1145/1011767.1011810
5. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distrib. Comput. **18**(4), 235–253 (2006). https://doi.org/10.1007/s00446-005-0138-3
6. Angluin, D., Aspnes, J., Eisenstat, D.: Stably computable predicates are semilinear. In: Proc. PODC, pp. 292–299, New York (2006)
7. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distrib. Comput. **20**(4), 279–304 (2007)
8. Angluin, D., Aspnes, J., Eisenstat, D.: A simple population protocol for fast robust approximate majority. Distrib. Comput. **21**(2), 87–102 (2008). https://doi.org/10.1007/s00446-008-0059-z
9. Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols with a leader. Distrib. Comput. **21**(3), 183–199 (2008). https://doi.org/10.1007/s00446-008-0067-z
10. Aspnes, J., Ruppert, E.: An introduction to population protocols. Bull. Euro. Assoc. Theor. Comput. Sci. **93**, 98–117 (2007)
11. Berenbrink, P., Elsässer, R., Friedetzky, T., Kaaser, D., Kling, P., Radzik, T.: A population protocol for exact majority with o(log5/3 n) stabilization time and theta(log n) states. In: Schmid, U., Widder, J. (eds.) 32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018, volume 121 of *LIPIcs*, pp. 10:1–10:18. Schloss Dagstuhl -Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.DISC.2018.10
12. Berenbrink, P., Friedetzky, T., Kaaser, D., Kling,P.: Tight & simple load balancing (2019). accepted at IPDPS 2019. An earlier version can be found under the arXiv ID arXiv:1808.05389 [cs.DC]
13. Bilke, A., Cooper, C., Elsässer, R., Radzik, T.: Brief announcement: Population protocols for leader election and exact majority with $O(\log(^2)\ n)$ states and $O(\log(^2)\ n)$ convergence time. In: Schiller, E.M., Schwarzmann, A.A. (eds.) Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017, pp. 451–453. ACM (2017). https://doi.org/10.1145/3087801.3087858
14. Doty, D., Eftekhari, M.: Efficient size estimation and impossibility of termination in uniform dense population protocols. In: Robinson, P., Ellen, F. (eds.) In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019, pp. 34–42. ACM (2019). https://doi.org/10.1145/3293611.3331627
15. Doty, D., Eftekhari, M., Michail, O., Spirakis, P.G., Theofilatos, M.: Exact size counting in uniform population protocols in nearly logarithmic time. CoRR, abs/1805.04832 (2018). URL http://arxiv.org/abs/1805.04832
16. Draief, M., Vojnovic, M.: Convergence speed of binary interval consensus. SIAM J. Control Optim. **50**(3), 1087–1109 (2012). https://doi.org/10.1137/110823018
17. Elsässer, R., Radzik, T.: Recent results in population protocols for exact majority and leader election. *Bull. EATCS* 126 (2018). URL http://bulletin.eatcs.org/index.php/beatcs/article/view/549/546
18. Gasieniec, L., Stachowiak, G.: Fast space optimal leader election in population protocols. In: Czumaj, A. (ed.) Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7–10, 2018, pp. 2653–2667. SIAM (2018). https://doi.org/10.1137/1.9781611975031.169
19. Kosowski, A., Uznanski, P.: Brief announcement: Population protocols are fast. In: Newport, C., Keidar, I. (eds.) Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018, pp. 475–477. ACM (2018). URL https://dl.acm.org/citation.cfm?id=3212788
20. Mertzios, G.B., Nikoletseas, S.E., Raptopoulos, C.L., Spirakis, P.G.: Determining majority in networks with local interactions and very small local memory. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) Automata, Languages, and Programming—41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I, volume 8572 of *Lecture Notes in Computer Science*, pp. 871–882. Springer (2014). https://doi.org/10.1007/978-3-662-43948-7_72
21. Mocquard, Y., Anceaume, E., Aspnes, J., Busnel, Y., Sericola, B.: Counting with population protocols. In: Avresky, D.R., Busnel, Y. (eds.) 14th IEEE International Symposium on Network Computing and Applications, NCA 2015, Cambridge, MA, USA, September 28–30, 2015, pp. 35–42. IEEE Computer Society (2015). https://doi.org/10.1109/NCA.2015.35