Time-Space Trade-offs in Population Protocols

Dan Alistarh
Microsoft Research
daalista@microsoft.com

James Aspnes
Yale
james.aspnes@yale.edu

David Eisenstat
Google
eisenstatdavid@gmail.com

Rati Gelashvili
MIT
gelash@mit.edu

Ronald L. Rivest
MIT
rivest@mit.edu

Abstract

Population protocols [AAD+06] are a popular model of distributed computing, in which randomly-interacting agents with little computational power cooperate to jointly perform computational tasks. Recent work has focused on the complexity of fundamental tasks in the population model, such as *leader election* (which requires convergence to a single agent in a special ''leader" state), and *majority* (in which agents must converge to a decision as to which of two possible initial states had higher initial count). Known upper and lower bounds point towards an inherent trade-off between the *time complexity* of these protocols, and the *space complexity*, i.e. size of the memory available to each agent.

In this paper, we explore this trade-off and provide new upper and lower bounds for these two fundamental tasks. First, we prove a new unified lower bound, which relates the space available per node with the time complexity achievable by the protocol: for instance, our result implies that any protocol solving either of these tasks for n agents using $O(\log\log n)$ states must take $\Omega(n/\mathsf{polylog}n)$ expected time. This is the first result to characterize time complexity for protocols which employ super-constant number of states per node, and proves that fast, poly-logarithmic running times require protocols to have relatively large space costs.

On the positive side, we show that $O(\mathsf{polylog}n)$ convergence time can be achieved using $O(\log^2 n)$ space per node, in the case of both tasks. Overall, our results highlight a time complexity separation between $O(\log\log n)$ and $\Theta(\log^2 n)$ state space size for both majority and leader election in population protocols. At the same time, we introduce several new tools and techniques, which should be applicable to other tasks and settings.

1 Introduction

Population protocols [AAD+06] are a model of distributed computing in which agents with little computational power and no control over the interaction schedule cooperate to collectively perform computational tasks. While initially introduced to model animal populations [AAD+06], they have proved a useful abstraction for wireless sensor networks [PVV09, DV12], chemical reaction networks [CCDS14], and gene regulatory networks [BB04].

Specifically, a population protocol consists of a set of n finite-state agents, interacting in pairs, where each interaction may update the local state of both participants. The protocol starts in a valid initial configuration, and the goal is to have all agents converge to an output value, representing the output of the computation, which is a predicate of the initial state of nodes. The set of interactions occurring in each step is assumed to be decided by a *probabilistic* scheduler, which picks the next pair to interact uniformly at random in each step. The fundamental measure of convergence is *parallel time*, defined as the number of pairwise interactions until convergence, divided by n.

Considerable work has been invested in studying the complexity of certain *fundamental tasks* in the population model. One such fundamental task is *majority (consensus)* [AAE08b, PVV09, DV12], in which agents start in one of two input states A and B, and must converge on a decision as to which state has a higher initial count. A complementary fundamental task is *leader election* [AAE08a, AG15, DS15], which requires the system to converge to states in which a *single* agent is in a special *leader* state. Efficient leader election is key for fast predicate computation [AAD+06, AAE08a, AAER07], as most known constructions require a leader to co-ordinate phases of computation. A parallel line of applied research has shown that these tasks can be implemented at the level of molecules [CDS+13], and that they are connected to computational tasks solved by living cells in order to function correctly [CCN12].

A progression of deep technical results [Dot14, CCDS14] culminated in showing that *leader election in sublinear time is impossible* for protocols which are restricted to a *constant* number of states per node [DS15]. At the same time, it is now known that leader election can be solved in $O(\log^3 n)$ time via a protocol requiring $O(\log^3 n)$ states per node [AG15]. For the majority task, the space-time complexity gap is even wider: sublinear time is impossible for protocols restricted to having at most *four* states per node [AGV15], while there exists a poly-logarithmic time protocol which requires a number of states per node that is *linear* in n [AGV15].

These results strongly suggest a trade-off between the *running time* of a population protocol and the *space*, or number of states, available at each agent. This relation is all the more important since time efficiency is critical in practical implementations, while technical constraints limit the number of states currently implementable in a molecule [CDS+13].[1] However, the characteristics of the time-space trade-off in population protocols are currently an open question.

Contribution: In this paper, we take a step towards answering this question by exhibiting a general trade-off between the number of states available to a population protocol, and its time complexity, as well as providing new and improved upper bounds for both majority and leader election, which are tight within poly-logarithmic factors. Along the way, we introduce several new tools and techniques.

Lower Bound: More precisely, when applied to majority, our lower bound proves that there exist constants $c \in (0,1)$ and $K \geq 1$ such that any protocol using $\lambda_n \leq c \log\log n$ states must take $\Omega(n/(K^{\lambda_n} + \epsilon n)^2))$ time, where ϵn is the difference between the initial counts of the two competing states. Specifically, any protocol using *constant* states and supporting a constant initial difference necessarily takes *linear* time.

For leader election, the bound shows that there exist constants $c \in (0,1)$ and $K \geq 1$ such that any protocol using $\lambda_n \leq c \log\log n$ states and electing at most $\ell(n)$ leaders, requires $\Omega(n/(K^{\lambda_n} \cdot \ell(n)^2))$ expected time. Specifically, any protocol electing one leader using $\leq c \log\log n$ states requires $\Omega(n/\text{polylog } n)$ time.

[1] One such technical constraint is the possibility of *leaks*, i.e. spurious creation of states following an interaction. In practice, the more states a protocol implements, the higher the likelihood of a leak, and the higher the probability of divergence.

Algorithms: On the positive side, we give new poly-logarithmic-time algorithms for majority and leader election which use $O(\log^2 n)$ space. Our majority algorithm, called Split-Join, runs in $O(\log^3 n)$ time with high probability, and uses $O(\log^2 n)$ states per node. The only previosly known algorithm to achieve sublinear time [AGV15] required $\Theta(n)$ states per node. Our new leader election algorithm uses $O(\log^2 n)$ states, and converges in $O(\log^9 n)$ expected parallel time. This reduces the state space size by a logarithmic factor over the best known algorithm [AG15], at the cost of a poly-logarithmic running time increase.

Techniques: The core of the lower bound is a technical argument proving that a hypothetical algorithm which would converge faster than allowed by the lower bound may reach "stable" configurations[2] in which certain low-count states can be "erased." This leads to a contradiction, since these low-count states may be exactly the set of all current leaders (in the case of leader election protocols), or a set of nodes whose state may sway the outcome of the majority computation. In particular, our argument employs the method of bounded differences to obtain a stronger version of the main density theorem of [Dot14], and develops a new technical characterization of the stable states which can be reached by a protocol, which does not require constant bounds on state space size, generalizing upon [DS15]. The argument provides a unified analysis: the bounds for each task in turn are corollaries of the main theorem characterizing stable configurations.

On the algorithmic side, we introduce a new *synthetic coin* technique, which allows nodes to generate almost-uniform local coins within a *constant* number of interactions, by exploiting the randomness in the scheduler, and in particular the properties of random walks on the hypercube. Synthetic coins are useful in both our protocols, for instance by allowing nodes to estimate the total number of agents in the system, and may be of independent interest as a way of generating randomness in a constrained setting. The Split-Join protocol is based on a new quantized averaging technique, by which nodes represent their output opinions and their relative strength by encoding them as powers of two, and opinions are averaged on each interaction.

Summary: In sum, our results can be seen as bad news for algorithm designers, since they show that convergence for both exact majority and leader election will be slow even if the protocol is able to implement a super-linear number of states per node. However, we show that achievable convergence time improves quickly as the size of the state space nears the logarithmic threshold: in particular, fast, poly-logarithmic time can be achieved using poly-logarithmic space.

It is interesting to notice that previous work by Chatzigiannakis et al. [MNRS14] identified the space threshold of $\Theta(\log\log n)$ for the computational power of a family of population protocols called *passively mobile machines*. In particular, their results show that variants of such systems in which nodes are limited to $o(\log\log n)$ space per node are limited to only computing *semilinear predicates*, whereas $O(\log\log n)$ space is sufficient for computing non-semilinear predicates, and $O(\log n)$ space is sufficient to compute general symmetric predicates. By contrast, we show a *complexity* separation between algorithms which use $O(\log\log n)$ space per node, and algorithms employing $\Omega(\log n)$ space per node.

2 Preliminaries

Population Protocols: We have n agents (also called cells, or nodes) each executing as a deterministic state machine with states from a finite set Λ, with a finite set of input symbols $X \subseteq \Lambda$, a finite set of output symbols Y, a transition function $\delta : \Lambda \times \Lambda \to \Lambda \times \Lambda$, and an output function $\gamma : \Lambda \to Y$. Each agent starts with an input from the set X, and keeps updating its state following interactions with other agents, according to the transition function δ. For simplicity of exposition, we assume that the agents have identifiers from the set $V = \{1, 2, \ldots, n\}$, although these identifiers are not known to agents, and not used by the protocols.

The agents' interactions proceed according to a directed *interaction graph* G without self-loops, whose edges indicate possible agent interactions. Usually, the graph G is considered to be the complete graph on n vertices, a convention we also adopt in this work.

[2] Roughly, a configuration is stable if it may not generate any new types of states.

The execution proceeds in *steps*, where in each step a new edge (u, w) is chosen uniformly at random from the set of edges of G. Each of the two chosen agents updates its state according to the function δ.

A population protocol computes a function $g : X^V \to Y$ within ℓ steps with probability $1 - \phi$ if for all $x \in g^{-1}(Y)$, the configuration $c : V \to \Lambda$ reached by the protocol after ℓ steps satisfies the following two properties with probability $1 - \phi$: (i) for all $v \in V$, $g(x) = \gamma(c(v))$. Specifically, all agents have the correct output in c; (ii) for every configuration c' reachable from c, and for all $v \in V$, $g(x) = \gamma(c'(v))$.

Parallel Time: The above setup considers sequential interactions; however, in general, interactions between pairs of distinct agents are independent, and are usually considered as occurring in parallel. In particular, it is customary to define one unit of *parallel time* as n consecutive steps of the protocol.

The Majority Problem: In the *majority problem*, agents start with arbitrary initial states in the input set $X = \{A, B\}$. Let a be the number of agents starting in state A, and b be the number of agents starting in state B, and let $\epsilon = |a - b|/n$ denote the relative advantage of an initial state. The output set is $Y = \{0, 1\}$.

A population protocol solves the majority problem within ℓ steps with probability $1 - \phi$, if, for any configuration $c : V \to \Lambda$ reachable by the protocol after $\geq \ell$ steps, it holds with probability $1 - \phi$ that (1) If $a > b$ for the given input, then for any agent i, $\gamma(c(i)) = 1$, and, conversely, (2) If $b > a$ for the given input, then for any agent i, $\gamma(c(i)) = 0$. We emphasize that in this paper we consider the *exact* majority task, as opposed to *approximate* majority [AAE08b], which may return the wrong answer with some probability.

Leader Election: In the *leader election* problem, all agents start in the same initial state A, i.e. the only state in the input set $X = \{A\}$. The output set is $Y = \{Win, Lose\}$.

A population protocol solves leader election within ℓ steps with probability $1 - \phi$, if it holds with probability $1 - \phi$ that for any configuration $c : V \to \Lambda$ reachable by the protocol after $\geq \ell$ steps, there exists a unique agent i such that, (1) for the agent i, $\gamma(c(i)) = Win$, and, (2) for any agent $j \neq i$, $\gamma(c(j)) = Lose$.

3 Lower Bound

3.1 Preliminaries

We now refine the notation of the previous section, to make it more amenable to proving lower bounds. We start by making the dependence on parameter n explicit. In particular, in this section, a population protocol $\mathcal{P}$ will be a sequence of protocols $\mathcal{P}_1, \mathcal{P}_2, \ldots$ with $\mathcal{P}_n$, one for each value of n, where for each $i \geq 1$ we have that $\mathcal{P}_i = (\Lambda_i, \delta_i)$, where Λ_i and δ_i are the set of protocol states and transitions for i agents, respectively.

We say that a protocol is *monotonic* if, for all $i \geq 1$, (1) $|\Lambda_i| \leq |\Lambda_{i+1}|$, and (2) If $|\Lambda_i| = |\Lambda_{i+1}|$, then $\Lambda_i = \Lambda_{i+1}$, *and* $\delta_i = \delta_{i+1}$. This definition allows for different population protocols for different number of states, and captures all protocols where the number of states is given as a parameter to the protocol, and may be super-constant, e.g. [AG15, AGV15]. We do require that if the number of states used by the population protocol is the same for agent counts i and j, then the whole protocol must necessarily be the same.

A configuration c of the system is formally a function $c : \Lambda_n \to \mathbb{N}$, where $c(s)$ represents the *count of s in c*. We let $|c|$ stand for the sum, over all states $s \in \Lambda$, of $c(s)$. For a given task, let $I = I_1, I_2, \ldots$ be a sequence of sets, where $|I_n| = n$ and I_n is a subset (possibly all) of allowed initial configurations for n agents. In leader election, I_n consists of a single uniform configuration where all agents are in the same state. For majority, I_n contains configurations where each agent is in one of two initial states A and B.

We say that a configuration y of n agents has a *stable leader*, if for all y' such that $y \implies y'$, it holds that $\sum_{\ell \in L_n} y'(\ell) = 1$, where L_n is the set of all leader states in Λ_n. Analogously, a configuration y has a *stable majority decision* for initial value A, if for all y' with $y \implies y'$, every agent in y' is in a state that corresponds to decision A. We say that a protocol *stabilizes*, when it reaches a stable output configuration. The quantity of interest is the expected parallel time for a population protocol to reach a configuration with a stable leader or with a stable majority decision from an initial configuration. We will also sometimes refer to this quantity as the *convergence time*. If the expected time is finite, then we say that population protocol stably elects a leader (or stably computes majority decision).

3.2 Technical Machinery

We now prove a set of technical lemmas, on which the main argument relies. In the following, we assume n to be fixed; Λ_n is the set of all states of the protocol, whose size may depend on n. Let S_0 be the set of states in the initial configuration. For all integers $k \geq 1$, define inductively the set of states

$$S_k = S_{k-1} \cup \delta_n(S_{k-1}, S_{k-1}).$$

Assume without loss of generality that all states in Λ_n actually occur in some configurations reachable by the protocol $\mathcal{P}_n$ from some input configuration. Then, it holds that $S_{|\Lambda_n|-1} = S_{|\Lambda_n|} = \ldots$, and $S_{|\Lambda_n|-1} = \Lambda_n$. We say that a configuration is *dense* if all present states have count $\geq n/M$ for some constant M. We prove the following statement, which generalizes the main result of [Dot14] to super-constant state counts.

Lemma A.1. *Let $\beta = 1/100$ be a constant. For all population protocols A using $|\Lambda_n| \leq 1/2 \log\log n$ states and starting in a dense initial configuration, with probability $\geq 1 - (1/n)^{1-\beta}$, there exists an integer j such that the configuration reached after j steps is $n^{1-\beta}$-rich, all states have counts more than $n^{1-\beta}$.*

Let $f : \mathbb{N} \to \mathbb{R}^+$ be some function. We say that a transition $\alpha : r_1, r_2 \to p_1, p_2$ is an *f-bottleneck* for configuration c, if $c(r_1) \cdot c(r_2) \leq f(|c|)$. If every transition sequence from initial configurations to final configurations contains a bottleneck, then we get a lower bound on the convergence time. Conversely, the next lemma shows that, if a protocol converges fast, then it is possible to converge using a bottleneck-free transition sequence from $n^{1-\beta}$-rich intermediate configurations (which are reachable by Lemma A.1).

Lemma A.3. *Let $\mathcal{P} = (\Lambda, \delta)$ be a population protocol using $|\Lambda_n| \leq 1/2 \log\log n$ states for all sufficiently large n, and let I be a sequence of sets of dense initial configurations. Fix a function f. Assume that for sufficiently large n, $\mathcal{P}$ stabilizes in expected time $o(\frac{n}{f(n)|\Lambda_n|^2})$ from all $i_n \in I_n$. Then, for all sufficiently large $m \in \mathbb{N}$ there exists a configuration x_m with $|x_m| = m$ agents, reachable from some $i \in I_m$ and transition sequence p_m with the following properties: (1) $x_m(s) \geq m^{1-\beta}$ for all $s \in \Lambda_m$, where $\beta = 1/100$ is a constant, (2) $x_m \Longrightarrow_{p_m} y_m$, where y_m is stable, and (3) p_m has no f-bottleneck.*

The above lemma establishes the existence of a bottleneck-free transition sequence to convergence from a sufficiently rich configuration. The next *transition ordering lemma*, due to [CCDS14], proves a property of such a transition sequence. We can order all states whose counts decrease more than some set threshold such that, for each of these states d_i, the sequence contains at least a certain number of a specific transition that consumes d_i, but does not consume or produce any states $d_1, \ldots, d_{i-1}$ that are earlier in the ordering.

Lemma A.4. *Let $b \in \mathbb{N}$ and $x, y \in \mathbb{N}^{\Lambda_n}$ in a system of n agents, such that $\forall s \in \Lambda_n : x(s) \geq b_2$ and $x \Longrightarrow_q y$ via a transition sequence q without a $(b_2)^2$-bottleneck, where $b_2 = |\Lambda_n|^2 \cdot b + |\Lambda_n| \cdot b$. Define*

$$\Delta = \{d \in \Lambda_n \mid y(d) \leq b\}$$

i.e. the set of states whose count in configuration y is at most b. Then, there is an order $\Delta = \{d_1, d_2, \ldots, d_k\}$, such that, for all $i \in \{1, \ldots, k\}$, there is a transition α_i of the form $d_i, s_i \to o_i, o_i'$ with $s_i, o_i, o_i' \notin \{d_1, \ldots, d_i\}$, and α_i occurs at least b times in q.

3.3 The Lower Bound Argument

Our framework is based on the lower bound argument of [DS15], but differs from it in a few key points. Specifically, the crux of the argument in [DS15] is the existence of a set Γ of *unbounded states*, whose counts grow unbounded in stable configurations, as the number of agents n tends to infinity. This property is then used to construct a leaderless reachable configuration where only states in Γ have non-zero counts. The unbounded property of Γ is used multiple times throughout the proof, and together with Dickson's Lemma it establishes the existence of a sequence of stable configurations with growing counts for all states in Γ. In turn, this implies that the leaderless configuration must be stable, as stability is closed downward[3], completing the contradiction argument.

[3] A configuration c must be stable if another configuration c' is stable with counts of all states not less than in c.

This argument breaks in our case: Dickson's Lemma does not apply for non-constant state counts, and even the definition of Γ becomes problematic. A notational issue is that we consider sequence of population protocols $\mathcal{P}_n$ for different values n, without requiring that they use the same state space. Even if they did, the set Γ according to its original definition could just be empty[4].

Instead, given a family of population protocols $\mathcal{P} = \mathcal{P}_1, \mathcal{P}_2, \ldots$ whose number of states grow as a function of n, for any fixed n, configuration $c \in \mathbb{N}^{\Lambda_n}$ and function $g : \mathbb{N} \to \mathbb{N}^+$, we define $\Gamma_g(c) = \{s \in \Lambda_{|c|} \mid c(s) > g(|c|)\}$ and $\Delta_g(c) = \{s \in \Lambda_{|c|} \mid c(s) \leq g(|c|)\}$, where $|c|$ is the number of agents in the system in this configuration. Notice that $\Gamma_g(c) = \Lambda_n - \Delta_g(c)$.

The proof strategy is to first show that if a protocol converges "too fast," then it can also reach configurations where all agents are in states in $\Gamma_g(c)$. These configurations are useful because of the following.

Lemma A.5. *Consider a population protocol in a system with any fixed number of agents n, and an arbitrary fixed function $h : \mathbb{N} \to \mathbb{N}^+$ such that $h(n) \geq 2^{|\Lambda_n|}$. Let $h'(n) = 2^{|\Lambda_n|}$. For all configurations $c, c' \in \mathbb{N}^{\Lambda_n}$, such that all agents in c are in states in $\Gamma_h(c)$ (i.e. $\forall d \in \Delta_h(c) : c(d) = 0$) and $\Gamma_h(c) \subseteq \Gamma_{h'}(c')$, any state producible from c is also producible from c'. Formally, if $c \implies y : y(s) > 0$ for some $s \in \Lambda_n$ and $y \in \mathbb{N}^{\Lambda_n}$, then there also exists $y' \in \mathbb{N}^{\Lambda_n}$ such that $c' \implies y'$ and $y'(s) > 0$.*

Next, we prove the main theorem.

Theorem 3.1. *Let $\mathcal{P}$ be a monotonic population protocol using at most $|\Lambda_n| \leq 1/2 \log\log n$ states for all sufficiently large n. Let $g : \mathbb{N} \to \mathbb{N}^+$ be a function such that $g(n) \geq 2^{|\Lambda_n|}$ for all n and $6^{|\Lambda_n|} \cdot |\Lambda_n|^2 \cdot g(n) = o(n^{1-\beta}) = o(n^{0.99})$. Let I be the sequence of sets of dense initial configurations for $\mathcal{P}$ with the property that $\forall i_1 \in I_{n_1}, i_2 \in I_{n_2}$, if $|\Lambda_{n_1}| = |\Lambda_{n_2}| = |\Lambda_{n_1+n_2}|$, then $i_1 + i_2 \in I_{n_1+n_2}$.*

If $\mathcal{P}$ stabilizes in $o\left(\frac{n}{(6^{|\Lambda_n|} \cdot |\Lambda_n|^3 \cdot g(n))^2}\right)$ expected time for any initial configuration, then we can find infinitely many m, such that for each m there exists an initial configuration $i \in I_{2m}$, and a stable final configuration y of m agents, reachable from an initial configuration in I_m, such that

(1) $i \implies z$, where $z \in \mathbb{N}^{\Gamma_g(y)}$ (i.e. all agents in configuration z are in states from $\Gamma_g(y)$).

(2) $i + c_g \implies z'$, where $z' \in \mathbb{N}^{\Gamma_g(y)}$ and c_g is configuration consisting of $g(m)/2 + 1$ agents in some state $s \in \Delta_g(y) : y(s) \leq g(m)/2 - 1$ (for any such state s, if such a state exists).

Proof. For simplicity, set $b(n) = (6^{|\Lambda_n|} + 2^{|\Lambda_n|}) \cdot g(n)$, $b_2(n) = |\Lambda_n|^2 \cdot b(n) + |\Lambda_n| \cdot b(n)$, and $f(n) = (b_2(n))^2$. The condition in the theorem statement implies that the protocol stabilizes in $o\left(\frac{n}{f(n)|\Lambda_n|^2}\right)$ time. Then, by Lemma A.3, for all sufficiently large m we can find $i_m, x_m, y_m \in \mathbb{N}^{\Lambda_m}$, all configurations with m agents, with $i_m \in I_m$ such that:

- $i_m \implies x_m \implies_{p_m} y_m$, where y_m is a stable final configuration and the transition sequence p_m does not contain an f-bottleneck (i.e. a $(b_2)^2$-bottleneck).
- $\forall s \in \Lambda_m : x_m(s) \geq b_2(m)$, where $\beta = 1/100$. (Here, we use the assumption on the function g.)

Moreover, because $|\Lambda_n| \leq 1/2 \log\log n$ for sufficiently large n, for infinitely many m it also additionally holds that $|\Lambda_m| = |\Lambda_{m+1}| = \ldots = |\Lambda_{3m}|$ which according to our definitions means that population protocols $\mathcal{P}_m, \mathcal{P}_{m+1}, \ldots, \mathcal{P}_{3m}$ are all the same. Otherwise $|\Lambda_n|$ would grow at least logarithmically in n.

Consider any such m. Then, we can invoke Lemma A.4 with x_m, y_m, transition sequence p_m and parameter $b = b(m)$. The definition of Δ in the lemma statement matches $\Delta_b(y_m)$, and b_2 matches $b_2(m)$. Thus, we get an ordering of states $\Delta_b(y_m) = \{d_1, d_2, \ldots, d_k\}$ and a corresponding sequence of transitions $\alpha_1, \alpha_2, \ldots, \alpha_k$, where each α_i is of a form $d_i, s_i \to o_i, o_i'$ with $s_i, o_i, o_i' \notin \{d_1, d_2, \ldots, d_i\}$. Finally, each transition α_i occurs at least $b(m) = (6^{|\Lambda_m|} + 2^{|\Lambda_m|}) \cdot g(m)$ times in p_m.

We will not perform a set of transformations on the transition sequence p_m, called *surgeries*, with the goal of converging to a desired type of configuration. The next two claims specify these transformations, which are similar to the surgeries used in [DS15], with the key difference that counts in Δ were bounded

[4] Consider for instance when the stable configurations for larger n increase the counts for the newest states.

while the counts in Γ grew without limit, allowing the gap to get arbitrarily large. This is no longer the case here, so we need to carefully bound the counts of agents in certain states, and adopt the second transformation and its proof to get the desired counts. The proofs are provided in the appendix.

Let c_g as defined in the theorem statement be a configuration of $g(m)$ agents each in some state $s \in \Lambda_m$ with $y_m(s) = 0$. The configuration y in the theorem statetement will be y_m. For brevity, we use $\Gamma_g = \Gamma_g(y_m)$, $\Delta_g = \Delta_g(y_m)$, $\Gamma_b = \Gamma_b(y_m)$ and $\Delta_b = \Delta_b(y_m)$.

Claim A.6. *There exist configurations $e, e' \in \mathbb{N}^{\Lambda_m}$ and $z_1, z_1' \in \mathbb{N}^{\Gamma_g}$, such that $e + x_m \implies z_1$, $e' + c_g + x_m \implies z_1'$. Moreover, we have an upper bound on the counts of states in e and e': $\forall s \in \Lambda_m : e(s) \leq 2^{|\Lambda_m|} \cdot g(m)$ and $e'(s) \leq 2^{|\Lambda_m|} \cdot g(m)$.*

The configurations $e + x_m$ and $e' + c_g + x_m$ have at most $2^{|\Lambda_m|} \cdot g(m) \cdot |\Lambda_m| + g(m) + m$ agents, which is less than $3m$ for sufficiently large m. In the end, when we construct i, i', z, z' and the transition sequences, it will also be in the system of at most $3m$ agents, corresponding to the exact same protocol as for m agents.

For any configuration $e \in \mathbb{N}^{\Lambda_m}$, let e^{Δ} be its projection onto Δ, i.e. a configuration consisting of only the agents in states Δ. We can define e^{Γ} analogously. By definition, $e^{\Gamma} = e - e^{\Delta}$.

Claim A.7. *Let e be any configuration with the property that $\forall s \in \Lambda_m : e(s) \leq 2^{|\Lambda_m|} \cdot g(m)$. There exist configurations $p \in \mathbb{N}^{\Delta_b}$ and $w \in \mathbb{N}^{\Gamma_g}$, such that $p + x_m \implies p + w + e^{\Delta_g}$. Moreover, for counts in p, we have that $\forall s \in \Lambda_m : p(s) \leq b(m)$ and for counts in w^{Γ_g}, we have $\forall s \in \Gamma_g : w(s) \geq 2^{|\Lambda_m|} \cdot g(m)$.*

Let our initial configuration i be $i_m + i_m$, which according to the Theorem assumptions, must also be a dense initial configuration from I_{2m}. Then, trivially $i \implies x_m + x_m$. Let us apply Claim A.7 with e as defined in Claim A.6, but use one x_m instead of p. This is possible because $\forall s \in \Lambda_m : x(s) \geq b_2(m) \geq b(m) \geq p(s)$. Hence, we get $x_m + x_m \implies x_m + w + e^{\Delta_g} = x_m + e + (w - e^{\Gamma_g})$. The configuration $w - e^{\Gamma_g}$ is well-defined because both w and e^{Γ_g} contain agents in states in Γ_g, with each count in w being larger or equal to the respective count in e^{Γ_g}, by the bounds from the claims.

Finally, by Claim A.6, we have $x_m + e + (w - e^{\Gamma_g}) \implies z_1 + (w - e^{\Gamma_g})$. We denote the resulting configuration (with all agents in states in Γ_g) by z, and have $i \implies z$, as desired. The proof for $i + c_g$ is fully analogous, by simply using the respective clause of Claim A.6 to get from $c_g + x_m + e + (w - e^{\Gamma_g}) \implies z_1' + (w - e^{\Gamma_g})$ in the last step of the argument. We let z' be the resulting configuration. □

This theorem implies the following lower bounds.

Corollary 3.2. *Any population protocol that uses $|\Lambda_n| \leq 1/2 \log\log n$ states for for all sufficiently large number of agents n and stably elects at least one and at most $\ell(n)$ leaders, must take $\Omega\left(\frac{n}{144^{|\Lambda_n|} \cdot |\Lambda_n|^6 \cdot \ell(n)^2}\right)$ expected parallel time to convergence.*

Proof. We set $g(n) = 2^{|\Lambda_n|} \cdot \ell(n)$. Input configurations for leader election consist of all agents in the same dedicated starting state. Hence, the sum of two input configurations of the same population protocol for leader election is also an input configuration, and all these configurations are dense.

Assume, to the contrary, that the protocol converges in parallel time $o\left(\frac{n}{144^{|\Lambda_n|} \cdot |\Lambda_n|^6 \cdot \ell(n)^2}\right)$. For all n, let I_n contain the only initial configuration with n agents in the initial state. Using Theorem 3.1, we can find infinitely many configurations i and z of $2m$ agents, such that (1) $i \implies z$, (2) $i \in I_{2m}$ is an initial configuration with $2m$ agents, (3) the same protocol is used for all number of agents between m and $2m$, implying $|\Lambda_m| = |\Lambda_{2m}|$, and (4) in z all agents are in states in $\Gamma_g(y)$, i.e. the states that each have counts at least $2^{|\Lambda_m|} \cdot \ell(m)$ in some stable final configuration y (of $|y| = m$ elements).

Because y is a stable final configuration of a protocol that elects at most $\ell(m)$ leaders, none of these states in $\Gamma_g(y)$ that are present in strictly larger counts ($2^{|\Lambda_m|} \cdot \ell(m) > \ell(m)$) in y and z can be leader states. Therefore, the configuration z does not contain a leader. This is not sufficient for a contradiction, because a leader election protocol may well pass through a leaderless configuration before converging to a stable configuration with at most $\ell(m)$ leaders. We prove below that any configuration reachable from z

must also have zero leaders. This implies an infinite time of convergence from a valid initial configuration i (as $i \Longrightarrow z$) and completes the proof by contradiction.

If we could reach a configuration from z with an agent in a leader state, then by Lemma A.5, from a configuration c' that consists of $2^{|\Lambda_m|}$ agents in each of the states in $\Gamma_g(y)$, it is also possible to reach a configuration with a leader, let us say through transition sequence q. Recall that the configuration y contains at least $2^{|\Lambda_m|} \cdot \ell(m)$ agents in each of these states in $\Gamma_g(y)$, hence there exist disjoint configurations $c'_1 \subseteq y$, $c'_2 \subseteq y$, etc, $\ldots, c'_{\ell(m)} \subseteq y$ contained in y and corresponding transition sequences $q_1, q_2, \ldots, q_{\ell(m)}$, such that q_i only affects agents in c'_i and leads one of the agents in c'_i to become a leader. Configuration y is a final configuration so it contains at least one leader agent already, which does not belong to any c'_i because as mentioned above, c'_i's only contain agents in non-leader states. Therefore, it is possible to reach a configuration from y with $\ell(m)+1$ leaders via a transition sequence q_1 on the c'_1 component of y, followed by q_2 on c'_2, etc, $q_{\ell(m)}$ on $c'_{\ell(m)}$, contradicting that y is a stable final configuration. □

The proof of the majority lower bound follows similarly, and is deferred to the Appendix.

Corollary A.8. *Any population protocol that uses $|\Lambda_n| \leq 1/2 \log\log n$ states for for all sufficiently large number of agents n and stably computes majority decision among two initial states with majority advantage ϵn, must take $\Omega\left(\frac{n}{36^{|\Lambda_n|} \cdot |\Lambda_n|^6 \cdot \max(2^{|\Lambda_n|}, \epsilon n)^2}\right)$ expected parallel time to convergence.*

4 Synthetic Coin Flips

The state transition rules in population protocols are deterministic, i.e. the interacting nodes do not have access to random coin flips. In this section, we introduce a general technique that extracts randomness from the schedule and after only constant parallel time, allows the interactions to rely on close-to-uniform synthetic coin flips. This turns out to be an useful gadget for designing efficient protocols.

Suppose that every node in the system has a boolean parameter $coin$, initialized with zero. This extra parameter can be maintained independently of the rest of the protocol, and only doubles the state space. When agents x and y interact, they both *flip* the values of their coins. Formally, $x'.coin \leftarrow 1 - x.coin$ and $y'.coin \leftarrow 1 - y.coin$, and the update rule is fully symmetric.

The nodes can use the $coin$ value of the interaction partner as a random bit in a randomized algorithm. However, these bits are not independent and uniform. However, we prove that with high probability, very quickly, the distribution of $coin$ becomes close to uniform and remains that way. We use the concentration properties of random walks on the hypercube, analyzed previously in various other contexts, e.g. [AR15]. We also note that a similar algorithm is used by Laurenti et al. [LCK16] to generate the uniform distribution in chemical reaction networks.

Theorem 4.1. *Suppose $k \geq \alpha n$ for a fixed constant $\alpha \geq 2$. Let X_i be the number of ones in the system after i interactions. For all sufficiently large n, we have that $\Pr[|X_k - n/2| \geq n/2^{4\alpha}] \leq 2\exp(-\alpha n/8)$.*

Proof. Let us number the nodes from 1 to n, and represent their coin values by a binary vector of size n. Suppose we knew a fixed vector v representing the coin values of the nodes after the interaction $k - \alpha n$. For example, if $k = \alpha n$, we know v is a zero vector, because of the way the algorithm is initialized. For $1 \leq t \leq \alpha n$, define by Y_t the pair of nodes that are flipped during interaction $k - \alpha n + t$. Then, X_k is a deterministic function of $Y_1, \ldots, Y_{\alpha n}$, (whereby the function encodes the fixed starting vector v). Moreover, Y_j are independent random variables and changing any one Y_j only changes X_k by at most 4. Hence, we can apply McDiarmid's inequality with $X_k = f_v(Y_1, \ldots, Y_{\alpha n})$.

Claim 4.2 (McDiarmid's inequality)**.** *Let $Y_1, \ldots, Y_m$ be independent random variables and let X be their function $X = f(Y_1, \ldots, Y_m)$, such that changing variable Y_j only changes the function value by at most c_j. Then, we have that $\Pr[|X - \mathbb{E}[X]| \geq \epsilon] \leq 2 \cdot \exp(-\frac{2\epsilon^2}{\sum c_j^2})$.*

Setting $\epsilon = \alpha\sqrt{n}$ we get $\Pr[|X_k - \mathbb{E}[X_k]| \geq \alpha\sqrt{n}] \leq 2 \cdot \exp(-\alpha n/8)$, given that the coin values after interaction $k - \alpha n$ are fixed and represented by vector v. Fixing v also fixes the number of ones among coin values in the system at that moment, which we will denote by x, i.e. $x = \sum_{j=1}^{n} v_j = X_{k-\alpha n}$. We can now prove the following.

Claim C.1. $\mathbb{E}[X_{i+m} \mid X_i = x] = n/2 + (1 - 4/n)^m \cdot (x - n/2)$.

By Claim C.1 we have $\mathbb{E}[X_k \mid X_{k-\alpha n} = x] = n/2 + (1 - 4/n)^{\alpha n} \cdot (x - n/2)$, which as $0 \leq x \leq n$ and $(1 - 4/n)^{\alpha n} \leq \exp(-4\alpha)$, implies $n/2 - n/2^{4\alpha+1} \leq \mathbb{E}[X_k \mid X_{k-\alpha n} = x] \leq n/2 + n/2^{4\alpha+1}$. For each fixed v, we can apply McDiarmid inequality as above, and get an upper bound on the probability that X_k (given the fixed v), diverges from the expectation by at most $\alpha\sqrt{n}$. But as we just established, for any v, the expectation we get in the bound will be at most $n/2^{4\alpha+1}$ away from $n/2$. Combining these and using $n/2^{4\alpha+1} \geq \alpha\sqrt{n}$ for all sufficiently large n gives the desired bound. □

Approximate Counting: Synthetic coins can be used to estimate the number of agents in the system, as follows. Each node counts the number of consecutive 1 synthetic flips it observes, until the first 0. The agents then exchange their values, always recording the maximum. The agents will converge to a number which is a constant-factor approximation of $\log n$. This property is made precise in the proof of Lemma D.2.

5 The Lottery Leader Election Algorithm

We now present a leader election protocol using $O(\log^2 n)$ states, converging in expected $O(\log^9 n)$ parallel time. The protocol starts from the leader-minion mechanism of [AG15], and uses synthetic coin flips to reduce the size of the state space. More precisely, a node's state in the algorithm is determined by six parameters: *coin*, *mode*, *payoff*, *level*, *counter*, and *ones*. All agents start in the same state, in which initial values are zero for all parameters.

Variables: *coin* admits binary values and is initialized to 0. *mode* describes the mode of operation of the agent, and assumes one of the four values: seeding, lottery, tournament, or minion. Its initial value is seeding. *payoff* $\leftarrow 0$, *level* $\leftarrow 0$ and *counter* $\leftarrow 4$ are all positive integers and *ones* $=$ true is a boolean. We will assume that $n \geq 196$, as needed for the synthetic coin flips. We fix a parameter m such that $(18 \log n)^2 \leq m \leq \frac{\exp(n/2)}{n^8}$, and the number of states per node will be $O(m)$ (16 for the four modes and binary parameters *ones* and *coin*, and $7m/3$ for the other three parameters combined).

Seeding Mode: All agents start in seeding mode. While seeding an agent decreases its *counter* on every interaction until it reaches 0. When the *counter* reaches 0, the agent changes to lottery mode. The idea is to make sure that the values of the agents' coins diverge sufficiently from the initial all-zero assignment before the majority of the agents start the rest of the protocol.

Lottery Mode: In lottery mode, an agent starts counting in its own *payoff* the number of consecutive interactions until observing 0 as the *coin* value of an interaction partner. When the agent first meets a 0, or if the agent reaches the maximum possible value that *payoff* can hold (set to $\sqrt{m}$), the agent x keeps its *payoff* in variable p_x, and changes its *mode* to tournament.

Tournament Mode: In tournament mode, the agents start at the *level* 0, and make repeated attempts to increase their level. Each agent x keeps track of *phases*, each consisting of $\Theta(\log p_x)$ consecutive interactions. For each phase, if all coin values of interaction partners are 1, then the agent's level $x.level$ is incremented; otherwise, it stays the same. An agent that reaches the maximum possible *level* (set at $\sqrt{m}/3 \log m$) remains in tournament mode, but stops increasing its level.

Whenever two agents x and y in *tournament* mode meet, they compare states $(x.payoff, x.level, x.coin)$ and $(y.payoff, y.level, y.coin)$. If the former is larger, then agent x eliminates agent y from the tournament. Practically, agent y sets its *mode* to *minion*. Note that agents with higher lottery payoff always have priority; if both *payoff* and *level* are equal, the *coin* value is used as a tie-breaker.

Minion Mode: An agent in minion mode keeps a record of the maximum $.payoff, .level$ pair ever encountered in any interaction in its own $payoff$ and $level$ parameters. If $x.mode =$ minion and $y.mode =$ tournament, and $(x.payoff, x.level) > (y.payoff, y.level)$, then the agent in state y will be eliminated from contention, and turned into a minion. Intuitively, as in [AG15], minions help leaders with high payoffs and levels to eliminate other contenders by spreading information. Importantly, minions do not use the coin value as a tie-breaker (as this could lead to a leader eliminating itself).

The intuition behind this process is that the agent with the highest lottery payoff eventually becomes the leader. This is an agent that manages to reach a high level, and will turn other competitors into its minions, which further propagate the information about the highest payoff and level through the system.

Analysis Overview: Recall that only nodes with $mode =$ minion are non-leaders, and once a node becomes a minion it remains a minion. Therefore, we start by proving that not all nodes can be minions at the same time, and if there are $n-1$ minions in the system, then there is a stable leader. Later, we will establish a finite convergence bound on expected time until only one node is not a minion, which proves correctness.

Lemma D.1. *In any reachable configuration, at least one node is not a minion. Furthermore, a configuration with $n-1$ minions must have a stable leader, meaning that the non-minion node will never become a minion, while all minions will remain minions.*

Lemma D.2. *Then, the expected number of interactions until all nodes have $.mode =$ tournament or $.mode =$ minion is $O(n \log^2 n)$. Moreover, once that happens, with probability at least $1 - 5/n^3$, at most $12 \log n$ nodes will hold the maximum $payoff$ value, which will be at least $\log n/4$ and at most $16 \log n$.*

Once all agents have left the seeding and lottery modes, the *second stage* of the algorithm begins. In this second stage, all nodes are in either tournament or minion modes. For any agent, if its payoff is p, it will now use the $counter$ parameter to observe $6 \log p + 12$ consecutive coin values of interaction partners and increase its $level$ if they all happen to be one. The maximum allowed $counter$ value is thus $6 \log m/2 + 12 \leq 7 \log m$, but it only counts up to $6 \log p + 12$, since p is the actual payoff as opposed to the maximum possible payoff $\sqrt{m}$. We will set the maximum possible value of $level$ to $\frac{\sqrt{m}}{3 \log m}$, which when combined with $\sqrt{m}$ possible values of $payoff$ and $7 \log m$ for $counter$, gives $7m/3$ for these three parameters combined.

Next, we focus on the second stage. Below, let us call *contenders* the nodes that have the highest $payoff$ after the first stage and are in tournament mode.

Lemma D.3. *With probability at least $1 - \frac{O(\log n)}{n^3}$, only one contender reaches level $\ell = \frac{3 \log n}{\log 18 + \log \log n}$, and for each level up to ℓ, it takes at most $O(n \log^9 n)$ interactions before some contender gets to a larger level. Conditioned on this high probability event, the expected number of interactions before having $n-1$ minions is $O(n \log^9 n)$.*

The above technical lemma implies the global convergence bound.

Corollary D.4. *The algorithm converges in expected parallel time $O(\log^9 n)$ and with high probability in parallel time $O(\log^{10} n)$.*

6 The Split-Join Majority Algorithm

Description: We now give an algorithm for exact majority using $O(\log^2 n)$ states per node.

Each algorithm state corresponds to a pair of integers x and y, represented by $\langle x, y \rangle$, whereby both integers come from $x, y \in \{0, 1, 2, 2^2, \ldots, 2^{\lceil \log n \rceil}\}$. Not all possible pairs are valid states, in particular, for all states of the algorithm, $x \neq y$ and $2 \cdot \min(x, y) \neq \max(x, y)$ hold. For technical reasons, we have two additional *weak states* represented as pairs $\langle 0, 0 \rangle^+$ and $\langle 0, 0 \rangle^-$. Therefore, if x and y are equal, the pair $\langle x, y \rangle$ must be either $\langle 0, 0 \rangle^+$ or $\langle 0, 0 \rangle^-$. We will refer to states and their corresponding value pairs

State Space:
$StrongStates = \{\langle x, y\rangle | x, y \in \{0, 1, 2, 2^2, \ldots, 2^{\lceil \log n \rceil}\}, x \neq y, 2 \cdot \min(x, y) \neq max(x, y)\}$,
$WeakStates = \{\langle 0, 0\rangle^+, \langle 0, 0\rangle^-\}$
Input: States of two nodes, $\langle x_1, y_1\rangle$ and $\langle x_2, y_2\rangle$
Output: Updated states $\langle x_1', y_1'\rangle$ and $\langle x_2', y_2'\rangle$

1 $Reduce(u, v) = \begin{cases} [0, 0] & \text{if } u = v \\ [u - v, 0] & \text{if } u = 2v \\ [0, v - u] & \text{if } 2u = v \\ [u, v] & \text{otherwise.} \end{cases}$

2 **procedure** cancel(x_1, y_1, x_2, y_2)
3 $\quad [x_1', y_2'] \leftarrow Reduce(x_1, y_2)$
4 $\quad [x_2', y_1'] \leftarrow Reduce(x_2, y_1)$
5 **procedure** join(x_1, y_1, x_2, y_2)
6 $\quad$ **if** $(x_1 - y_1 > 0$ *and* $x_2 - y_2 > 0$ *and* $y_1 = y_2)$ **then** $y_1' \leftarrow y_1 + y_2$ and $y_2' \leftarrow 0$
7 $\quad$ **else** $y_1' \leftarrow y_1$ and $y_2' \leftarrow y_2$
8 $\quad$ **if** $(x_1 - y_1 < 0$ *and* $x_2 - y_2 < 0$ *and* $x_1 = x_2)$ **then** $x_1' \leftarrow x_1 + x_2$ and $x_2' \leftarrow 0$
9 $\quad$ **else** $x_1' \leftarrow x_1$ and $x_2' \leftarrow x_2$
10 **procedure** split$(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle)$
11 $\quad$ **if** $(x_1 - y_1 > 0$ *or* $x_2 - y_2 > 0)$ ***and*** $\max(x_1, x_2) > 1$ ***and*** $\min(x_1, x_2) = 0$ **then**
12 $\quad\quad x_1' \leftarrow \max(x_1, x_2)/2$ and $x_2' \leftarrow \max(x_1, x_2)/2$
13 $\quad$ **else** $x_1' \leftarrow x_1$ and $x_2' \leftarrow x_2$
14 $\quad$ **if** $(x_1 - y_1 < 0$ *or* $x_2 - y_2 < 0)$ ***and*** $\max(y_1, y_2) > 1$ ***and*** $\min(y_1, y_2) = 0$ **then**
15 $\quad\quad y_1' \leftarrow \max(y_1, y_2)/2$ and $y_2' \leftarrow \max(y_1, y_2)/2$
16 $\quad$ **else** $y_1' \leftarrow y_1$ and $y_2' \leftarrow y_2$
17 **procedure** normalize(x, y, v)
18 $\quad [\hat{x}, \hat{y}] \leftarrow Reduce(x, y)$
19 $\quad$ **if** $x = 0$ *and* $y = 0$ **then**
20 $\quad\quad$ **if** $v \geq 0$ **then** $\langle x', y'\rangle \leftarrow \langle 0, 0\rangle^+$
21 $\quad\quad$ **else** $\langle x', y'\rangle \leftarrow \langle 0, 0\rangle^-$
22 $\quad$ **else** $\langle x', y'\rangle \leftarrow \langle x, y\rangle$
23 **procedure** interact$(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle)$
24 $\quad$ **if** $x_1 = y_1 = x_2 = y_2 = 0$ **then** $[\langle x_1', y_1'\rangle, \langle x_2', y_2'\rangle] \leftarrow [\langle x_1, y_1\rangle, \langle x_2, y_2\rangle]$
25 $\quad$ **else**
26 $\quad\quad [\hat{x_1}, \hat{y_1}, \hat{x_2}, \hat{y_2}] \leftarrow$ split(join(cancel(x_1, y_1, x_2, y_2)))
27 $\quad\quad \langle x_1', y_1'\rangle \leftarrow$ normalize$(\hat{x_1}, \hat{y_1}, \hat{x_2} - \hat{y_2})$
28 $\quad\quad \langle x_2', y_2'\rangle \leftarrow$ normalize$(\hat{x_2}, \hat{y_2}, \hat{x_1} - \hat{y_1})$

Figure 1: The state update rules for the Split-Join majority algorithm.

interchangeably.

Nodes start in one of two special states. By convention, nodes starting in state A have the initial pair $\langle 2^{\lceil \log n \rceil}, 0\rangle$, and the nodes starting in state B have the initial pair $\langle 0, 2^{\lceil \log n \rceil}\rangle$. We define a *value* of a state corresponding to a pair $\langle x, y\rangle$ as $value(\langle x, y\rangle) = x - y$. Then, the output function γ maps each state to the output based the sign of its value (treating $\langle 0, 0\rangle^+$ as positive and $\langle 0, 0\rangle^-$ as negative). To show that the algorithm solves exact majority, we prove that all nodes converge to the values of the same sign as the initial majority (positive for A, negative for B), and that the convergence is fast with high probability.

The algorithm, specified in Figure 1, consists of a set of simple deterministic update rules for the node state. In the pseudocode, a pair $\langle x, y\rangle$ always stands for a state, while $[x, y]$ just means a tuple of integer values (without refering to a state). Hence, the *Reduce* helper function takes two values and returns two values. The main interaction rule between the states $\langle x_1, y_1\rangle$ and $\langle x_2, y_2\rangle$ of two interacting nodes is described by the function *interact*. The states after the interaction are $\langle x_1', y_1'\rangle$ and $\langle x_2', y_2'\rangle$. All nodes start in the designated initial states and continue to interact according to the interact rule.

If both of the interacting states are weak, nothing changes (line 24). Otherwise, three elementary reactions, *cancel*, *join*, and *split* are applied, in this order. The reaction *cancel* takes four values x_1, y_1, x_2, y_2 and returns (possibly updated) values x_1', y_1', x_2', y_2'. These values are then fed as inputs to the *join* reaction, that also returns four values which are then fed to *split*. The four outputs of *split* are $\hat{x_1}, \hat{y_1}, \hat{x_2}, \hat{y_2}$, which are then normalized to form the output states (because $\langle \hat{x_1}, \hat{y_1}\rangle$ may not be a valid state).

The elementary reactions are each described as a function of four inputs x_1, y_1, x_2, y_2 and four outputs

x'_1, y'_1, x'_2, y'_2 We say that a *cancel*, *split* or *join* reaction is *successful* if $x_i \neq x'_i$ or $y_i \neq y'_i$ for $i \in \{0, 1\}$.

Correctness and Convergence: The first observation is that the sum of values in the system is constant throughout the execution. Since the initial sum is of the majority sign, the algorithm is guaranteed to be correct. The proof of convergence follows by carefully tracking the maximum value in the system, and showing that minority values get cancelled out and switch sign quickly. Due to space limitations, we only state the main claim here, and defer the complete proof to the Appendix.

Theorem B.1. *The Split-Join algorithm will never converge to the minority decision, and is guaranteed to converge to the majority decision within $O(\log^3 n)$ parallel time, w.h.p.*

7 Conclusion

We studied the trade-off between time and space complexity in population protocols, and showed that a super-constant state space is necessary to obtain fast, poly-logarithmic convergence time for both leader election and exact majority. On the positive side, we gave algorithms which achieve poly-logarithmic expected convergence time using $O(\log^2 n)$ states per node for both tasks.

Our findings are not necessarily good news from the practical standpoint, as even small constant state counts are currently difficult to implement [CDS+13]. It is interesting to note how nature appears to have overcome this impossibility: algorithms solving majority at the cell level do so *approximately*, allowing for a positive probability of error, using small constant states per node and converging in poly-logarithmic time [AAE08b].

We open several avenues for future work. The first is to tightly characterize the time-space trade-off, between $\log \log n$ and $\mathsf{polylog} n$ states. This question appears challenging, and will likely require the development of analytic techniques parametrized by the number of states that an algorithm employs. A second direction is to explore the space-time trade-off in the case of approximately correct algorithms.

References

[AAD+06] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed computing*, 18(4):235–253, 2006.

[AAE08a] Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, September 2008.

[AAE08b] Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, July 2008.

[AAER07] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, November 2007.

[AG15] Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *Automata, Languages, and Programming*, pages 479–491. Springer, 2015.

[AGV15] Dan Alistarh, Rati Gelashvili, and Milan Vojnovic. Fast and exact majority in population protocols. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, 2015.

[AR15] Alexandr Andoni and Ilya Razenshteyn. Tight lower bounds for data-dependent locality-sensitive hashing. *arXiv preprint arXiv:1507.04299*, 2015.

[BB04] James M Bower and Hamid Bolouri. *Computational modeling of genetic and biochemical networks*. MIT press, 2004.

[CCDS14] Ho-Lin Chen, Rachel Cummings, David Doty, and David Soloveichik. Speed faults in computation by chemical reaction networks. In *Distributed Computing*, pages 16–30. Springer, 2014.

[CCN12] Luca Cardelli and Attila Csiksz-Nagy. The cell cycle switch computes approximate majority. *Nature Scientific Reports*, 2:656, 2012.

[CDS+13] Yuan-Jyue Chen, Neil Dalchau, Niranjan Srnivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from dna. *Nature Nanotechnology*, 8(10):755–762, 2013.

[Dot14] David Doty. Timing in chemical reaction networks. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 772–784. SIAM, 2014.

[DS15] David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. In *Proceedings of the 2015 International Symposium on Distributed Computing*, DISC '15, 2015.

[DV12] Moez Draief and Milan Vojnovic. Convergence speed of binary interval consensus. *SIAM Journal on Control and Optimization*, 50(3):1087–1109, 2012.

[LCK16] Luca Laurenti, Luca Cardelli, and Marta Kwiatkowska. Programming discrete distributions with chemical reaction networks. 2016. http://arxiv.org/abs/1601.02578.

[MNRS14] George B. Mertzios, Sotiris E. Nikoletseas, Christoforos Raptopoulos, and Paul G. Spirakis. Determining majority in networks with local interactions and very small local memory. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, ICALP '14, pages 871–882, 2014.

[PVV09] Etienne Perron, Dinkar Vasudevan, and Milan Vojnovic. Using three states for binary consensus on complete graphs. In *INFOCOM 2009, IEEE*, pages 2527–2535. IEEE, 2009.

A Lower Bound

Lemma A.1 (Density Lemma)**.** *Let $\beta = 1/100$ be a constant. For all population protocols A using $|\Lambda_n| \leq 1/2 \log\log n$ states and starting in a dense initial configuration, with probability $\geq 1 - (1/n)^{1-\beta}$, there exists an integer j such that the configuration reached after j steps is $n^{1-\beta}$-rich.*

Proof. We begin by defining, for integers $k \geq 0$, the function

$$f(k) = n51^{-2^k+1}.$$

Alternatively, we have that $f(k)^2 = f(k+1)n/51$.
Let $c = 1/2$, and $\beta = 1/100$. Given the above, we notice that, for our choice of c and β, it holds that, for sufficiently large $n \geq 2$,

- $3(c \log\log n)^2/n \leq (1/n)^{1-\beta}$, and
- for $0 \leq k \leq c \log\log n$, we have that $f(k) \geq \max(n^{1-\beta}, 50\sqrt{n \log n})$.

We divide the execution into phases of index $k \geq 0$, each containing $n/2$ consecutive interactions. For each $0 \leq k \leq |\Lambda_n| - 1$, we denote by C_k the system configuration at the beginning of phase k.

Inductive Claim.: We use probabilistic induction to prove the following claim: assuming that configuration C_k is $f(k)$-rich with respect to the set of states S_k, with probability $1 - 3|\Lambda_n|/n$, the configuration C_{k+1} is $f(k+1)$-rich with respect to S_{k+1}.

For general $k \geq 0$, let us fix the interactions up to the beginning of phase k, and assume that configuration C_k is $f(k)$-rich with respect to the set of states S_k. Further, consider a state $q \in S_{k+1}$. We will aim to prove that, with probability $1 - O(1/n)$, the configuration C_{k+1} satisfies contains state q with count $\geq f(k)$.

First, we define the following auxiliary notation. For any node r and set of nodes I, and define this counts the number of interactions between r and nodes in the set I, i.e.

$$\mathsf{intcount}(I, r) = |\{ \text{ interaction } j \text{ in phase } k : \text{there exists } i \in I \text{ such that } e_j = (i, r) \}|.$$

Next, we define the set of nodes in a state s at the beginning of phase k as

$$W(s) = \{v : v \in V \text{ and } C_k(v) = s\}.$$

Finally, we isolate the set of nodes in state s at the beginning of phase k which *did not interact* during phase k as

$$W'(s) = \{v : v \in W(s) \textit{ and } \mathsf{intcount}(V, v) = 0\}.$$

Returning to the proof, there are two possibilities for the state q. The first is when $q \in S_k$, that is, the state is already present at the beginning of phase k. But then, by assumption, state q has count $\geq f(k)$ at the beginning of phase k. To lower bound its count at the end of phase k, it is sufficient to examine the size of the set $W'(q)$. For a node $v \in W(q)$, the probability that $v \in W'(q)$ is

$$\left(1 - \frac{1}{n}\right)^{n/2} \geq 1/2,$$

by Bernoulli's inequality. Therefore the expected size of $W'(q)$ is at least $|W(q)|/2$. Changing any interaction during phase k may change $|W'(q)|$ by at most 1, and therefore we can apply the method of bounded differences to obtain that

$$\Pr\left[|W'(q)| < \frac{|W(q)|}{2} - \sqrt{n \log n}\right] \leq \exp\left(-\frac{n \log n}{n}\right) = \frac{1}{n}.$$

Since, by assumption, $|W(q)| \geq f(k) \geq 10\sqrt{n \log n}$, it follows that

$$\Pr\left[|W'(q)| < \frac{2}{5} f(k)\right] \leq \frac{1}{n}.$$

Since $2f(k)/5 \geq f(k+1)$, we have that $\Pr[\#q(C_{k+1}) \geq f(k+1)] \geq 1 - 1/n$, which concludes the proof of this case.

It remains to consider the case when $q \in S_{k+1} - S_k$. Here, we know that there must exist states q_i and q_r in S_k such that $\delta(q_i, q_r) = q$. We wish to lower bound the number of interactions between nodes in state q_i and nodes in state q_r throughout phase k. To this end, we isolate the set R of nodes which are in state q_r at the beginning of phase k, and only interact once during the phase, i.e.

$$R = \{v : v \in W(q_r) \text{ and } \mathsf{intcount}(V, v) = 1\},$$

and the set of nodes R', which are in R, and only interacted once during phase k, with a node in the set $W'(q_i)$, i.e.

$$R' = \{v : v \in R \text{ and } \mathsf{intcount}(W'(q_i), v) = 1\}.$$

Notice that any node in the set R' is necessarily in state q at the end of phase $k+1$. In the following, we lower bound the size of this set.

First, a simple probabilistic argument yields that $E[|R|] \geq |W(q_r)|/4$. Since each interaction in this phase affects the size of R by at most 2 (since it changes the count of both interaction partners), we can again apply the method of bounded differences to obtain that

$$\Pr\left[|R| < \frac{|W(q_r)}{4} - 2\sqrt{n \log n}\right] \leq \frac{1}{n},$$

implying that

$$\Pr\left[|R| < \frac{1}{20} f(k)\right] \leq \frac{1}{n}.$$

To lower bound the size of R', we apply again the method of bounded differences. We have that $|W'(q)| \geq (2/5)f(k)$, and that $|R| \geq (1/20)f(k)$, we have that

$$\Pr\left[|R'| \leq \frac{1}{50}\left(\frac{f(k-1)^2}{n}\right) - \sqrt{n\log n}\right] \leq \frac{1}{n}.$$

At the same time, we have that

$$\frac{1}{50}\left(\frac{f(k)^2}{n}\right) - \sqrt{n\log n} \geq \frac{51}{50}f(k+1) - \frac{1}{50}f(k+1) = f(k+1),$$

which concludes the claim in this case as well. Finally, we can apply a union bound on the set of states to obtain that, with probability $\geq 1 - 3|\Lambda_n|/n$,

Final Argument.: According to the lemma statement, we are considering an initial configuration in which all states which are present have count $\geq n/M$, for some constant $M \geq 0$. Let k_0 be the first positive integer such that $n/M \geq f(k_0)$. We have that the initial configuration is $f(k_0)$-rich with respect to the set of initial states S_0. By a variant of the previous inductive claim, we obtain that, for any integer $0 \leq \ell \leq |\Lambda_n|$ satisfying $f(k_0+\ell) \geq \max(n^{1-\beta}, 10\sqrt{n\log n})$, at the beginning of phase ℓ, configuration C_ℓ is $f(k_0+\ell)$-rich with respect to S_ℓ.

It therefore follows that, with probability at least

$$(1 - 3|\Lambda_n|/n)^{|\Lambda_n|} \geq 1 - 3(c\log\log n)^2/n \geq 1 - 1/n^{1-\beta},$$

there exists an integer j such that the configuration reached after j steps is $n^{1-\beta}$-rich. □

Claim A.2. *In a system of n nodes, let $\gamma > 0, f : \mathbb{N} \to \mathbb{R}^+, c \in \mathbb{N}^{\Lambda_n}$, and $X, Y \subseteq \mathbb{N}^{\Lambda_n}$ such that $\Pr[c \Longrightarrow X] \geq \gamma$, and every transition sequence from every $x \in X$ to some $y \in Y$ has an f-bottleneck. Then $T[c \Longrightarrow Y] \geq \gamma\frac{n-1}{2f(n)|\Lambda_n|^2}$.*

Proof. We will prove that for any $x \in X$, $T[x \Longrightarrow Y] \geq \frac{n-1}{2f(n)|\Lambda_n|^2}$ holds, which implies the desired claim. By definition, every transition sequence from x to a configuration $y \in Y$ contains an f-bottleneck, so it is sufficient to lower bound the expected time for the first f-bottleneck transition to occur from x before reaching Y. In any configuration c reachable from x, for any pair of states $r_1, r_2 \in \Lambda_n$ such that $r_1, r_2 \to p_1, p_2$ is a f-bottleneck transition in c, the definition implies that $c(r_1) \cdot c(r_2) \leq f(n)$. Thus the probability that the next pair of agents selected to interact are in states r_1 and r_2, is at most $\frac{2f(n)}{n(n-1)}$. Taking an union bound over all $|\Lambda_n|^2$ possible such transitions, the probability that the next transition is f-bottleneck is at most $|\Lambda_n|^2\frac{2f(n)}{n(n-1)}$. Bounding by a geometric variable with success probability $\frac{2f(n)|\Lambda_n|^2}{n(n-1)}$, the expected number of interactions until the first f-bottleneck transition is at least $\frac{n(n-1)}{2f(n)|\Lambda_n|^2}$. The expected parallel time is this quantity divided by n, completing the argument. □

Lemma A.3. *Let $\mathcal{P} = (\Lambda, \delta)$ be a population protocol such that $|\Lambda_n| \leq 1/2\log\log n$ states for all sufficiently large n, and let I be a sequence of sets of dense initial configurations. Assume that there exists a Q, such that for all sufficiently large n, $\mathcal{P}$ Q-stabilizes in expected time $o(\frac{n}{f(n)|\Lambda_n|^2})$ from all $i_n \in I_n$. Then, for all sufficiently large $m \in \mathbb{N}$ there exists a configuration x_m with $|x_m| = m$ agents, reachable from some $i \in I_m$ and transition sequence p_m with the following properties,*

- *$x_m(s) \geq m^{1-\beta}$ for all $s \in \Lambda_m$, where $\beta = 1/100$ is a constant.*
- *$x_m \Longrightarrow_{p_m} y_m$, where y_m is Q-stable, and*
- *p_m has no f-bottleneck.*

Proof. Recall that, as defined earlier, Q is a sequence of subset of transitions from the respective elements of δ, i.e. $Q_n \subseteq \delta_n$ holds for all n. Also, I_n is a set of some legal initial configurations for n agents, which are all given to be dense. We kno that the expected time to reach a Q-stable configuration from i is finite. Hence if $i \Longrightarrow x_m$ for $i \in I_m$, then a Q-stable configuration y_m must be reachable from x_m through some transition sequence p_m, but we also need x_m and p_m to satisfy the first and third requirements.

We know $|\Lambda_n| \leq 1/2\log\log n$ for all large enough n. Hence, by Lemma A.1, for a constant $\beta = 1/100$, starting in any dense configuration $i_n \in I_n$, with probability at least $1 - (1/n)^{1-\beta}$, an $n^{1-\beta}$-rich

configuration is reachable. So $n > 2$, we get that $\Pr[i_n \Longrightarrow X_n] \geq 1/2$ where $X_n = \{x \mid i_n \Longrightarrow x$ and $(\forall s \in \Lambda_n)x(s) \geq n^{1-\beta}\}$.

Let Y_n be a set of all Q-stable configurations with n agents. Suppose that every transition sequence from every configuration $x \in X_n$ to some $y \in Y$ has an f-bottleneck. Then, using Claim A.2, the expected time to Q-stabilize from $i \in I_n$ is $T[i_n \Longrightarrow Y] \geq \frac{1}{2} \cdot \frac{n-1}{2f(n)|\Lambda_n|^2} = \Theta(\frac{n}{f(n)|\Lambda_n|^2})$. But we know that the protocol Q-stabilizes from $i \in I_n$ in time $o(\frac{n}{f(n)|\Lambda_n|^2})$, implying that for all sufficiently large m, we can find $x_m \in X_m$ from which it is possible to reach a Q-stable configuration in Y without an f-bottleneck. First requirement is satisfied by the definition of X_m, and we let p_m be the transition sequence from x_m to some $y \in Y$ without an f-bottleneck. □

Lemma A.4. *Let $b \in \mathbb{N}$ and $x, y \in \mathbb{N}^{\Lambda_n}$ in a system of n agents, such that $\forall s \in \Lambda_n : x(s) \geq b_2$ and $x \Longrightarrow_q y$ via a transition sequence q without a $(b_2)^2$-bottleneck, where $b_2 = |\Lambda_n|^2 \cdot b + |\Lambda_n| \cdot b$. Define*

$$\Delta = \{d \in \Lambda_n \mid y(d) \leq b\}$$

i.e. the set of states whose count in configuration y is at most b. Then, there is an order $\Delta = \{d_1, d_2, \ldots, d_k\}$, such that, for all $i \in \{1, \ldots, k\}$, there is a transition α_i of the form $d_i, s_i \to o_i, o_i'$ with $s_i, o_i, o_i' \notin \{d_1, \ldots, d_i\}$, and α_i occurs at least b times in q.

Proof. The argument is identical as in [CCDS14,DS15] and is described below for the sake of completeness.

Let $k = |\Delta|$ and define $\Delta_k = \Delta$. We will construct the ordering in reverse, i.e. we will determine d_i for $i = k, k-1, \ldots, 1$ in this order. At each step, we will define the next Δ_{i-1} as $\Delta_i - \{d_i\}$.

We start by setting $i = k$. For all i we define $\Phi_i : \mathbb{N}^{\Lambda_n} \to \mathbb{N}$ based on Δ_i as $\Phi_i(c) = \sum_{d \in \Delta_i} c(d)$, i.e. the number of agents in states from Δ_i in configuration c. Notice that once Δ_i is well-defined, so is Φ_i.

The following works for all $i \geq 1$ and lets us construct the ordering. Because $y(d) \leq b$ for all states in Δ, it follows that $\Phi_i(y) \leq i \cdot b \leq |\Lambda_n| \cdot b$. On the other hand, we know that $x(d) \geq b_2$ for all $d \in \Delta_i$, hence $\Phi_i(x) \geq b_2 \geq |\Lambda_n| \cdot b \geq \Phi_i(y)$. Let c' be the last configuration along q from x to y where $\Phi_i(c') \geq b_2$, and r be the suffix of q after c'. Then, r must contain a subsequence of transitions u each of which strictly decreases Φ_i, with the total decrease over all of u being at least $\Phi_i(c') - \Phi_i(y) \geq b_2 - |\Lambda_n| \cdot b \geq |\Lambda_n|^2 \cdot b$.

Let $\alpha : r_1, r_2 \to p_1, p_2$ be any transition in u. α is in u so it strictly decreases Φ_i, and without loss of generality $r_1 \in \Delta_i$. Transition α is not a $(b_2)^2$-bottleneck, since u (and q) do not contain such bottlenecks, and all configurations c along u have $c(d) < b_2$ for all $d \in \Delta_i$ by definition of r. Hence, we must have $c(r_2) > b_2$ meaning $r_2 \notin \Delta_i$. Exactly one state in Δ_i decreases its count in transition α, but α strictly decreases Φ_i, so it must be that both $p_1 \notin \Delta_i$ and $p_2 \notin \Delta_i$. We take $d_i = r_1, s_i = r_2, o_i = p_1$ and $o_i' = p_2$.

There are $|\Lambda_n|^2$ different types of transitions. As each transition in u decreases Φ_i by exactly one and there are at least $|\Lambda_n|^2 \cdot b$ such instances, at least one transition type must repeat in u at least b times, completing the proof. □

Lemma A.5. *Consider some population protocol in a system with any fixed number of agents n, and an arbitrary fixed function $h : \mathbb{N} \to \mathbb{N}^+$ such that $h(n) \geq 2^{|\Lambda_n|}$. Let $h'(n) = 2^{|\Lambda_n|}$. For all configurations $c, c' \in \mathbb{N}^{\Lambda_n}$, such that all agents in c are in states in $\Gamma_h(c)$ (i.e. $\forall d \in \Delta_h(c) : c(d) = 0$) and $\Gamma_h(c) \subseteq \Gamma_{h'}(c')$, any state producible from c is also producible from c'. Formally, if $c \Longrightarrow y : y(s) > 0$ for some $s \in \Lambda_n$ and $y \in \mathbb{N}^{\Lambda_n}$, then there also exists $y' \in \mathbb{N}^{\Lambda_n}$ such that $c' \Longrightarrow y'$ and $y'(s) > 0$.*

Proof. Since $h(n) \geq 2^{|\Lambda_n|}$, for any state from $\Gamma_h(c)$, its count in c is at least $2^{|\Lambda_n|}$. As $\Gamma_h(c) \subseteq \Gamma_{h'}(c')$, the count of each of these states in c' is also at least $h'(n) = 2^{|\Lambda_n|}$. We say two agents have the same type if they are in the same state in c. We will prove by induction that any state that can be produced by some transition sequence from c, can also be produced by a transition sequence in which at most $2^{|\Lambda_n|}$ agents of the same type participate (ever interact). Configuration c only has agents with types (states) in $\Gamma_h(c)$, and configuration c' also has at least $2^{|\Lambda_n|}$ agents for each of those types, the same transition sequence can be performed from c' to produce the same state as from c, proving the desired statement.

The inductive statement is the following. There is a $k \leq |\Lambda_n|$, such that for each $i = 0, 1, \ldots, k$ we can find sets $S_0 \subset S_1 \subset \ldots \subset S_k$ where S_k contains all the states that are producible from c. Let A_i be a set consisting of 2^i agents of each type in $\Gamma_h(c)$, out of all the agents in configuration c (we could also use c'), for the total of $2^i \cdot |\Gamma_h(c)|$ agents. There are enough agents of these types in c (and in c') as $i \leq k \leq |\Lambda_n|$. Then, for each $0 \leq i \leq k$ and each state $s \in S_i$, there exists a transition sequence from c in which only the agents in A_i ever interact and in the resulting configuration, one of these agents from A_i ends up in state s.

We do induction on i and for the base case $i = 0$ we take $S_0 = \Gamma_h(c)$. The set A_0 as defined contains one (2^0) agent of each type in $\Gamma_h(c) = S_0$[5]. All states in S_0 are immediately producible by agents in A_0 via an empty transition sequence (without any interactions).

Let us now assume inductive hypothesis for some $i \geq 0$. If S_i contains all the producible states from configuration c, then $k = i$ and we are done. We will have $k \leq \Lambda_n$, because $S_0 \neq \varnothing$ and $S_0 \subset S_1 \subset \ldots S_k$ imply that S_k contains at least k different states, and there are $|\Lambda_n|$ total. Otherwise, there must be some state $s \notin S_i$ that can be produced after an interaction between two agents both in states in S_i, let us say by a transition $\alpha : r_1, r_2 \to s, p$ with $r_1, r_2 \in S_i$. Also, as S_i contains at least i states out of $|\Lambda_n|$ total, and there is the state $s \notin S_i$, $i < |\Lambda_n|$ holds and the set A_{i+1} is well-defined. Let us partition A_{i+1} into two disjoint sets B_1 and B_2 where each contain 2^i agents from c for each type. Then, by induction hypothesis, there exists a transition sequence where only the agents in B_1 ever interact and in the end, one of the agents $b_1 \in B_1$ ends up in the state r_1. Analogously, there is a transition sequence for agents in B_2, after which an agent $b_2 \in B_2$ ends up in state r_2. Combining these two transition and adding one instance of transition α in the end between agents b_1 and b_2 (in states r_1 and r_2 respectively) leads to a configuration where one of the agents from A_{i+1} is in state s. Also, all the transitions are between agents in A_{i+1}. Hence, we can set $S_{i+1} = S_i \cup \{s\}$, completing the inductive step. □

Claim A.6. *There exist configurations $e, e' \in \mathbb{N}^{\Lambda_m}$ and $z_1, z_1' \in \mathbb{N}^{\Gamma_g}$, such that $e + x_m \implies z_1$, $e' + c_g + x_m \implies z_1'$. Moreover, we have an upper bound on the counts of states in e and e': $\forall s \in \Lambda_m : e(s) \leq 2^{|\Lambda_m|} \cdot g(m)$ and $e'(s) \leq 2^{|\Lambda_m|} \cdot g(m)$.*

Proof. The proof is analogous to [DS15], but we consider a subsequence of the ordered transitions $\Delta_b = \{d_1, \ldots, d_k\}$ obtained earlier by Lemma A.4. Since $b(m) \geq g(m)$, we can represent $\Delta_g = \{d_{i_1}, \ldots, d_{i_l}\}$, with $i_1 \leq \ldots \leq i_l$. We iteratively add groups of transitions at the end of transition sequence p_m, (p_m is the transition sequence from x_m to y_m), such that, after the first iteration, the resulting configuration does not contain any agent in d_{i_1}. Next, we add group of transitions and the resulting configuration will not contain any agent agent in d_{i_1} or d_{i_2}, and we repeat this l times. In the end, no agents will be in states from Δ_g.

The transition ordering lemma provides us with the transitions to add. Initially, there are at most $g(m)$ agents in state d_{i_1} in the system. So, in the first iteration, we add the same amount (at most $g(m)$) of transitions $d_{i_1}, s_{i_1} \to o_{i_1}, o'_{i_1}$, after which, as $s_{i_1}, o_{i_1}, o'_{i_1} \notin \{d_1, \ldots d_{i_1}\}$, the resulting configuration will not contain any agent in configuration d_{i_1}. If there are not enough agents in the system in state s_{i_1} already to add all these transitions, then we add the remaining agents in state in s_{i_1} to e (or e'). For the first iteration, we may need to add at most $g(m)$ agents.

For the second iteration, add transitions of type $d_{i_2}, s_{i_2} \to o_{i_2}, o'_{i_2}$ to the resulting transition sequence. Therefore, the number of agents in d_{i_2} that we may need to consume is at most $3 \cdot g(m)$, $g(m)$ of them could have been there in y_m, and we may have added $2 \cdot g(m)$ in the previous iteration, if for instance both o_{i_1} and o'_{i_1} were d_{i_2}. Otherwise, the iteration is analogous, and we may add $3 \cdot g(m)$ extra agents to e (or e').

If we repeat these iterations for all remaining $j = 3, \ldots, l$, in the end we will end up in a configuration z (or z') that contains all agents in states in Γ as desired, because of the property of transition ordering lemma that $s_{i_j}, o_{i_j}, o'_{i_j} \notin \{d_1, \ldots, d_{i_j}\}$. For any j, the maximum total number of agents we may need to add to e at iteration j is $(2^j - 1) \cdot g(m)$. The worst case is when o_{i_1} and o'_{i_1} are both d_{i_2}, and o_{i_2}, o'_{i_2} are both d_{i_3}, etc.

[5] In c, all the agents are in one of the states of $\Gamma_h(c)$, so as long as $n > 0$ there must be at least one agent per state (type). Also, if $\Gamma_h(c) = \varnothing$, then n must necessarily be 0, so nothing is producible $A_0 = \varnothing$, $k = 0$ and we are done

Finally, it must hold that $l < |\Lambda_m|$, because the final configuration contains m agents in states in Γ_g and none in $\{d_{i_1}, \ldots, d_{i_l}\}$, so Γ_g cannot be empty. Therefore, the total number of agents added to e (or e') are $g(m) \cdot \sum_{j=1}^{l} 2^l - 1 < 2^{l+1} \cdot g(m) \leq 2^{|\Lambda_m|} \cdot g(m)$. This completes the proof because $e(s)$ for any state s can be at most the number of agents in e, which is at most $2^{|\Lambda_m|} \cdot g(m)$. □

Claim A.7. *Let e be any configuration with the property that $\forall s \in \Lambda_m : e(s) \leq 2^{|\Lambda_m|} \cdot g(m)$. There exists configurations $p \in \mathbb{N}^{\Delta_b}$ and $w \in \mathbb{N}^{\Gamma_g}$, such that $p + x_m \implies p + w + e^{\Delta_g}$. Moreover, for counts in p, we have that $\forall s \in \Lambda_m : p(s) \leq b(m)$ and for counts in w^{Γ_g}, we have $\forall s \in \Gamma_g : w(s) \geq 2^{|\Lambda_m|} \cdot g(m)$.*

Proof. The following proof has some significant differences from its counterpart in [DS15]. Also, as in the proof of Claim A.6, we define a subsequence ($i_1 \leq i_l$), $\Delta_g = \{d_{i_1}, \ldots, d_{i_l}\}$ of $\Delta_b = \{d_1, \ldots, d_k\}$ that we obtained earlier by Lemma A.4. We again start by the transition sequence p_m from configuration x_m to y_m, and perform iterations for $j = 1, \ldots k$. At each iteration, we modify the transition sequence, possibly add some agents to configuration p, which we will define shortly, and consider the counts of all agents not in p in the resulting configuration. Configuration p acts as a buffer of agents in certain states that we can temporarily borrow. For example, if we need 5 agents in a certain state with count 0 to complete some iteration j, we will temporarily let the count to -5 (add 5 agents to p), and then we will fix the count of the state to its target value, which will also return the "borrowed" agents (so p will also appear in the resulting configuration). As in [DS15], this allows us let the counts of certain states temporarily drop below 0.

We will maintain the following invariants on the count of agents, excluding the agents in p, in the resulting configuration after iteration j:

1) The counts of all states (not in p) in $\Delta_g \cap \{d_1, \ldots, d_j\}$ match to the desired counts in e^{Δ}.
2) The counts of all states in $\{d_1, \ldots d_j\} - \Delta_g$ are at least $2^{|\Lambda_m|} \cdot g(m)$.
3) The counts in any state diverged by at most $(3^j - 1) \cdot 2^{|\Lambda_m|} \cdot g(m)$ from the respective counts in y_m.

These invariants guarantee that we get all the desired properties after the last iteration. Let us consider the final configuration after itaration k. Due to the first invariant, the set of all agents (not in p) in states Δ_g is exactly e^{Δ_g}. All the remaining agents (also excluding agents in p) are in w, and thus, by definition, the counts of states in Δ_g in configuration w will be zero, as desired. The counts of agents in states $\Delta_b - \Delta_g$ that belong to w will be at least $2^{|\Lambda_m|} \cdot g(m)$, due to the second invariant. Finally, the counts of agents in Γ_b that belong to w will also be at least $b(m) - 3^{|\Lambda_m|} \cdot 2^{|\Lambda_m|} \cdot g(m) \geq 2^{|\Lambda_m|} \cdot g(m)$, due to the third invariant and the fact that the states in Γ_b had counts at least $b(m)$ in y_m. Finally, the third invariant also implies the upper bound on counts in p. The configuration p will only contain the agents in states Δ_b, because the agents in Γ_b have large enough starting counts in y_m borrowing is never necessary.

In iteration d_j, we fix the count of state d_j. Let us first consider the case when d_j belongs to Δ_g. Then, the target count is the count of the state d_j in e^{Δ_g}, which we are given is at most $2^{|\Lambda_m|} \cdot g(m)$. Combined with the third invariant, the maximum amount of fixing required may be is $3^j \cdot 2^{|\Lambda_m|} \cdot g(m)$. If we have to reduce the number of d_j, then we add new transitions $d_j, s_j \to o_j, o'_j$, similar to Claim A.6 (as discussed above, not worrying about the count of s_j possibly turning negative). However, in the current case, we may want to increase the count of d_j. In this case, we remove instances of transition $d_j, s_j \to o_j, o'_j$ from the transition sequence. The transition ordering lemma tells us that there are at least $b(m)$ of these transitions to start with, so by the third invariant, we will always have enough transitions to remove. We matched the count of d_j to the count in e^{Δ_g}, so the first invariant still holds, and so does the second one. The third invariant also holds, because we performed at most $3^j \cdot 2^{|\Lambda_m|} \cdot g(m)$ transition additions or removals, each affecting the count of any other given state by at most 2, and hence the total count differ by at most

$$(3^j - 1) \cdot 2^{|\Lambda_m|} \cdot g(m) + 2 \cdot 3^j \cdot 2^{|\Lambda_m|} \cdot g(m) = (3^{j+1} - 1) \cdot 2^{|\Lambda_m|} \cdot g(m).$$

Now assume that d_j belongs to $\Delta_b - \Delta_g$. If the count of d_j is aleady larger than $2^{|\Lambda_m|} \cdot g(m)$, than we do nothing and move to the next iteration, and all the invariants will hold. If the count is smaller than $2^{|\Lambda_m|} \cdot g(m)$, then we set the target count to $2^{|\Lambda_m|} \cdot g(m)$ and add or remove transitions like in the previous case, thus the first two invariants will again hold after the iteration. Because d_j is in Γ_g, the its initial

count in y_m was at least $g(m)$, it diverged by at most $(3^j - 1) \cdot 2^{|\Lambda_m|} \cdot g(m)$, and ended up less than $2^{|\Lambda_m|} \cdot g(m)$ which we are fixing to $2^{|\Lambda_m|} \cdot g(m)$. Therefore, the maximum amount of fixing required is $(3^j - 1) \cdot 2^{|\Lambda_m|} \cdot g(m) + (2^{|\Lambda_m|} \cdot g(m) - g(m))$, which is at most $3^j \cdot 2^{|\Lambda_m|} \cdot g(m)$, as in the previous case, we will have enough transitions to remove and the third invariant will also hold. □

Corollary A.8. *Any population protocol that uses $|\Lambda_n| \leq 1/2 \log\log n$ states for for all sufficiently large number of agents n and stably computes majority decision among two initial states with majority advantage ϵn, must take $\Omega\left(\frac{n}{36^{|\Lambda_n|} \cdot |\Lambda_n|^6 \cdot \max(2^{|\Lambda_n|}, \epsilon n)^2}\right)$ expected parallel time to convergence.*

Proof. We set $g(n) = \max(2^{|\Lambda_m|+1}, 4\epsilon n)$. For majority computation, initial configurations consist of agents in one of two states, corresponding to two opinions, with the majority opinion holding an ϵn advantage in the counts. Therefore, the sum of two initial configurations of the same protocol is also a valid input configuration. The bound is nontrivial only in a regime $\epsilon n \in o(\sqrt{n})$, which we will henceforth assume without loss of generality. The initial configurations we consider in I_n will all have advantage ϵn, and therefore will all be dense.

Let us prove that for all sufficiently large m, in any final stable configuration y, strictly less than $2^{|\Lambda_m|} \leq g(m)/2$ agents will be in the initial minority state s_B. The reason is that if c is the initial configuration of all m agents in state s_B, the protocol must converge from c to a final configuration where the states correspond to decision s_B. By Lemma A.5, from any configuration that contains at least $2^{|\Lambda_m|}$ agents in s_B it would also be possible to reach a configuration where some agent supports decision s_B. Therefore, all stable final configuration y have at most $g(m)/2 - 1$ agents in initial minority state s_B. This allows us to let c_g be a configuration of $g(m)/2 + 1 \geq 2\epsilon m + 1$ agents in state s_B.

Assume, to the contrary, that the protocol converges in parallel time $o\left(\frac{n}{36^{|\Lambda_n|} \cdot |\Lambda_n|^6 \cdot \max(2^{|\Lambda_n|}, \epsilon n)^2}\right)$. Recall that I is the sequence of sets of dense initial configurations I_n. For all n, we let I_n contain configurations with $\frac{(1+\epsilon)n}{2}$ agents in state s_A and $\frac{(1-\epsilon)n}{2}$ agents in state s_B, i.e. having majority opinion s_A with advantage ϵn[6]. Using Theorem 3.1, we can find infinitely many configurations i and z' of at most $3m$ agents, such that (1) $i + c_g \implies z'$, (2) $i \in I_{2m}$ is an initial configuration for $2m$ agents, with majority opinion s_A and advantage $2\epsilon m$. (3) the same protocol is used for all number of agents between m and $3m$, implying $|\Lambda_m| = |\Lambda_{2m}| = |\Lambda_{2m+|c_g|}|$, and (4) in z' all agents are in states in $\Gamma_g(y)$, i.e. the states that have counts at least $g(m)$ in some stable final configuration y of m elements, reachable from an initial configuration in I_m.

To get the desired contradiction we will prove two things. First, z' is actually a stable final configuration for decision s_A (majority opinion in i), and second, $i + c_g$ is a valid initial configuration for the majority problem, but with majority opinion s_B (majority of agents in s_B). This will imply that the protocol converges to a wrong outcome, and complete the proof by contradiction.

If we could reach a configuration from z' with any agent in a state corresponding to the decision for s_B, then by Lemma A.5, from a configuration y (which contains $2^{|\Lambda_m|}$ agents in each of the states in $\Gamma_g(y)$) it is also possible to reach a configuration with an agent supporting s_B. This is impossible, as configuration y is a final stable configuration for an initial configuration in I_m, which has a majority of s_A.

Configuration $i \in I_{2m}$ contains $2\epsilon m$ more s_A states than s_B. Configuration c_g consists of at least $2\epsilon m + 1$ agents all in state s_B. Hence, $i + c_g$ is a legal initial configuration with a majority of s_B. □

B Analysis of the Majority Algorithm

The update rules in Figure 1 are chained, i.e. a cancel is followed by a join and a split. This is an optimization, applying as many possible reactions as possible. However, for the analysis we consider a slight modification, where we only apply split only if both join and cancel were unsuccessful.

For presentation purposes, we assume that n is a power of two, and when necessary, we assume that it is sufficiently large. Throughout this proof, we denote the set of nodes executing the protocol by V. We

[6]To clarify, I_n has to contain some dense initial configurations for n agents, but not necessarily all such configurations.

measure execution time in discrete steps (rounds), where each time step t corresponds to an interaction. The *configuration* at a given time t is a function $c : V \to Q$, where $c(v)$ is the state of the node v at time t. (We omit the explicit time t when clear from the context.)

Recall that a value of a state $\langle x, y \rangle$ is defined as $x - y$ and we will also refer to $\max(x, y)$ as the *level* of this node. We call $\langle x, y \rangle$ a *mixed* state if both x and y are non-zero, and a *pure* state otherwise. A mixed or pure node is a node in a mixed or a pure state, respectively.

The rest of this section is focused on proving the following result.

Theorem B.1. *The Split-Join algorithm will never converge to the minority decision, and is guaranteed to converge to the majority decision within $O(\log^3 n)$ parallel time, w.h.p.*

Correctness: We first prove that nodes never converge to the sign of the initial minority (safety), and that they eventually converge to the sign of the initial majority (termination).

The first statement follows since given the interaction rules of the algorithm, the sum of the encoded values stays constant as the algorithm progresses. The proof follows by the structure of the algorithm.

Invariant B.2. *The sum $\sum_{v \in V} value(c(v))$ never changes, for all reachable configurations c of the protocol.*

This invariant implies that the algorithm may never converge to a wrong decision value. For instance, if the initial sum is positive, then positive values must always exist in the system. Therefore we only need to show that the algorithm converges to a state where all nodes have the same sign, which we do via a rough convergence bound, assuming an arbitrary starting configuration.

Lemma B.6. *Let c be an arbitrary starting configuration. We define the (initial) sum of values as $S := \sum_{v \in V} value(c(v))$. By assumption, $S \neq 0$. With probability 1, the algorithm will reach a configuration $\hat{c}$ such that i) $sgn(\hat{c}(v)) = \mathrm{sgn}(S)$ for all nodes $v \in V$,* and, *ii) no node changes its sign in configurations c_e reachable from $\hat{c}$, i.e. $\forall c_e$ reachable from c and $v \in V : sgn(c_e(v)) = sgn(\hat{c}(v))$. For sufficiently large n, the convergence time to $\hat{c}$ is at most n^5 expected communication rounds, i.e. parallel time n^4.*

Convergence Time: Next, we bound the time until all nodes converge to the correct sign.

Claim B.3. *Consider a configuration where $n\delta$ out of the n nodes are in a mixed state, for $\delta \geq \frac{2(\log n - 1)}{n}$. In the next interaction round, the number of mixed nodes strictly decreases with probability at least $\frac{\delta^2}{2(\log n - 1)}$.*

Proof. Consider $\log n - 1$ buckets corresponding to values $1, 2, 4, \ldots, n/4$. Let us assign mixed nodes to these buckets according to their states, where node in state $\langle x, y \rangle$ goes into bucket $\min(x, y)$. All nodes fall into one of the $\log n - 1$ buckets because of the definition of (mixed) states.

If two nodes in the same bucket interact, either cancel or join will be successful, and since we consider the algorithm where split is not applied in this case, and the number of mixed nodes will strictly decrease. Thus, if there are $d_1, d_2, \ldots, d_{\log n - 1}$ nodes in the buckets, the number of possible interactions that decrease the number of mixed nodes is at least $\sum_{i=1}^{\log n - 1} \frac{d_i(d_i - 1)}{2} = \frac{(\sum d_i^2) - n\delta}{2}$.

By the Cauchy-Schwartz inequality, $\sum d_i^2 \geq \frac{n^2\delta^2}{\log n - 1}$. Combining this with the above and using $n\delta \geq 2(\log n - 1)$ we get that the there are at least $\frac{n^2\delta^2}{4(\log n - 1)}$ pairs of nodes whose interactions decrease the number of mixed nodes. The total number of pairs is $n(n-1)/2$, proving the desired probability bound. ☐

Claim B.4. *Suppose f is a function such that $f(n) \in O(\mathsf{poly}(n))$. For all sufficiently large n, the probability of having less than $\frac{n}{2^{11} \log n}$ pure nodes in the system at any time during the first $f(n)$ communication rounds is at most $1 - 1/n^5$.*

Proof. Assume that this number became less than $\frac{n}{2^{11} \log n}$ for the first time at time T after some number of communication rounds. Let t be the last time when the number of pure nodes was at least $\frac{n}{2^9 \log n}$ (such a

time exists since the initial number of pure nodes is n) and let α be the number of communication rounds between t and T. The number of mixed nodes increases by at most two in each round, so $\alpha \geq \frac{n}{2^{11}\log n}$.

By definition of t and T, at all times during the α communication rounds between t and T, at least $\frac{n(2^9\log n-1)}{2^9\log n} \geq \frac{n}{2}$ nodes are mixed. Thus, by Claim B.3 in each of these communication rounds, the number of mixed nodes decreases by at least one with probability at least $\frac{1}{8\log n}$. Let us describe by a random variable $X \sim \mathrm{Bin}(\alpha, \frac{1}{8\log n})$ at least how often the number of mixed nodes decreased. Each node is pure or mixed, and by Chernoff Bound, the probability that the number of pure nodes increased less than $\frac{\alpha}{16\log n}$ times is $\Pr\left[X \leq \frac{\alpha}{16\log n}\right] = \Pr\left[X \leq \frac{\alpha}{8\log n}(1-1/2)\right] \leq \exp\left(-\frac{\alpha}{8\log n\cdot 2^2\cdot 2}\right) \leq \exp\left(-\frac{n}{2^{17}\log^2 n}\right)$

On the other hand, in each of these α rounds, the number of pure nodes can decrease only if one of the interacting nodes was in a pure state. By definition of t and T, the number of such pairs is at most $\frac{n^2}{2^{18}\log^2 n} + 2\frac{n^2(2^9\log n-1)}{2^{18}\log^2 n} \leq \frac{2n^2}{2^9\log n}$. This implies that in each round the probability that the number of pure nodes will decrease is at most $\frac{1}{2^6\log n}$. Let us describe the (upper bound on the) number of such rounds by a random variable $Y \sim \mathrm{Bin}(\alpha, \frac{1}{2^6\log n})$. Since in each such round the number of pure nodes can decrease by at most 2, using Chernoff bound the probability that the number of pure nodes decreases by more than $\frac{\alpha}{16\log n}$ during the α communication rounds is at most $\Pr\left[Y \geq \alpha/(32\log n)\right] \leq \exp\left(-\frac{n}{2^{19}\log^2 n}\right)$

In order for the number of pure nodes to have decreased from $\frac{n}{2^9\log n}$ at time t to $\frac{n}{2^{11}\log n}$ at time T, either the number of mixed nodes must have increased by at most $\frac{\alpha}{16\log n}$, or the number of pure nodes must have decreased by at least $\frac{\alpha}{16\log n}$ during the α communication rounds between t and T. Otherwise, the increase in mixed nodes would be more than the decrease in pure nodes. However, by union bound, the probability of this is at most $\exp\left(-\frac{n}{2^{17}\log^2 n}\right) + \exp\left(-\frac{n}{2^{19}\log^2 n}\right)$.

We can now take union bound over the number of communication rounds until the number of pure nodes drops below $\frac{n}{2^{11}\log n}$ (time T). For at most $f(n) \in O(\mathsf{poly}(n))$ rounds, we get that the probability of the number of pure nodes ever being less than $\frac{n}{2^{11}\log n}$ is at most $1/n^5$ for all large enough n. □

Consider the high probability case of the above claim, where a fraction of pure nodes are present in every configuration in the execution prefix. We call a round a *negative-round* if, in the configuration c at the beginning of the round, there are at least $\frac{n}{2^{11}\log n}$ pure nodes and at least *half* of the pure nodes encode a non-positive value. Analogously, we call a round a *positive-round* if there are at least $\frac{n}{2^{11}\log n}$ pure nodes, at least half of which encode a non-negative value. A round can be simultaneously negative *and* positive, for instance when all pure nodes encode value 0. Next claim establishes the speed at which the maximum level in the system decreases. The proof, given in full in the Appendix, follows by bounding the probability that a node with the maximum level meets a pure node with value 0 or a value of the opposite sign. This results in a split (or cancel) reaction decreasing the level of the node, and we use Chernoff and Union Bounds to bound the probability that the node avoids such a meeting for significant time.

We get a condition for halving the maximum level (among positive or negative values) in the system with high probability. The initial levels in the system is n, which can only be halved $\log n$ times for each sign. Combining everything results in the following claim:

Claim B.8. *There exists a constant β, such that if during the first $2\beta n\log^3 n$ rounds the number of pure nodes is always at least $\frac{n}{2^{11}\log n}$, then with probability at least $1 - \frac{2\log n}{n^5}$, one of the following three events occurs at some point during these rounds:*

1. *Nodes only encode values in $\{-1, 0, 1\}$;*
2. *There are less than $\frac{n}{2^{12}\log n}$ nodes with non-positive values, all encoding 0 or -1,*
3. *There are less than $\frac{n}{2^{12}\log n}$ nodes with non-negative values, all encoding 0 or 1.*

Final Argument: To see how this claim can be used to obtain the convergence upper bound, let us assume without loss of generality that the initial majority of nodes was in A (positive) state, i.e. $a > b$.

Setting β as in Claim B.8, by Claim B.4, with high probability, we have at least $\frac{n}{2^{11}\log n}$ pure nodes during the first $2\beta n \log^3 n$ rounds. Thus, w.h.p. during these rounds the execution reaches a configuration where one of the three events from Claim B.8 holds. Consider this point T in the execution.

By our assumption about the initial majority and Invariant B.2, $\sum_{v\in V} value(c(v)) = \epsilon n^2$ holds in every reachable configuration c. The third event is impossible, because the total sum would be negative. In the first event, the total sum is $\epsilon n^2 \geq n$ of n encoded values each being -1, 0 or 1. Therefore, in this case, all nodes must be in state $\langle 1, 0\rangle$ and we are done.

In the second event implies there are at least $\frac{n(2^{12}\log n - 1)}{2^{12}\log n} \geq \frac{2n}{3}$ nodes encoding strictly positive values. Hence, at time T during the first $2\beta n \log^3 n$ rounds there are at least $n/3$ more strictly positive than strictly negative values. Moreover, -1's are the only strictly negative values of the nodes at point T, and this will be the case for the rest of the execution because of the update rules. After time T, we have

Claim B.9. *Consider a configuration where all nodes with strictly negative values encode -1, while at least $\frac{2n}{3}$ nodes encode strictly positive values. The number of rounds until convergence is $O(n \log n)$ in expectation and $O(n \log^2 n)$ with high probability.*

Using this, and by Union Bound over Claim B.4 and Claim B.8, with probability $1 - \frac{\log n + 1}{n^5}$ the number of communication rounds to convergence is thus $2\beta n \log^3 n + O(n \log^2 n) = O(n \log^3 n)$.

In the remaining low probability event, with probability at most $\frac{\log n + 1}{n^5}$, the remaining number of rounds is at most $O(n^5)$ by Lemma B.6. Therefore, the same $O(n \log^3 n)$ bound also holds in expectation,

Claim B.5. *There are at most $2n^2$ split reactions in any execution.*

Proof. A level of a node in state $s = \langle x, y\rangle$ is defined as $level(s) = \max(x, y)$. Consider a node in a state with level l. Then, we say that the *potential of the node* is $\phi(l) = 2l$ for $l > 0$ and $\phi(0) = 1$. In any configuration c, the *potential of the system* is $\Phi(c) = \sum_{i=1}^{n} \phi(level(s_i))$.

Then, the potential of the system in the initial configuration is $\sum(2n) = 2n^2$, and it can never fall below $\sum(1) = n$. By the interaction rules of the algorithm, potential of the system never increases after an interaction, and it decreases by at least one after each successful split interaction. This implies the claim. □

Lemma B.6. *Let c be an arbitrary starting configuration. Define $S := \sum_{v\in V} value(c(v)) \neq 0$. With probability 1, the algorithm will reach a configuration $\hat{c}$ such that $sgn(\hat{c}(v)) = \mathrm{sgn}(S)$ for all nodes $v \in V$. Moreover, in all later configurations c_e reachable from $\hat{c}$, no node can ever have a different sign, i.e. $\forall v \in V : sgn(c_e(v)) = sgn(\hat{c}(v))$. For sufficiently large n, the convergence time to $\hat{c}$ is at most n^5 expected communication rounds, i.e. parallel time n^4.*

Proof. Assume without loss of generality that the sum S is positive.

We estimate the expected convergence time by splitting the execution into three phases. The first phase starts at the beginning of the execution, and lasts until either i) no node encodes a strictly negative value *or* ii) each node encodes a value in $\{-1, 0, 1\}$, i.e. all nodes are in states $\langle 1, 0\rangle$, $\langle 0, 0\rangle^+$. $\langle 0, 0\rangle^-$ or $\langle 0, 1\rangle$.

Due to Invariant B.2, at least one node encodes a strictly positive value. Also, by definition, during the first phase there is always a node encoding a strictly negative value. Moreover, there is a node in state $\langle x, y\rangle$ with $\max(x, y) > 1$. Assume that $x > y$ for this node. Then, if there is another node in state $\langle 0, y_2\rangle$ for any y_2, then with probability at least $1/n^2$ these two nodes interact in the next round resulting in a split reaction. Otherwise, every node $\langle x_1, y_1\rangle$ that encodes a strictly negative value must have $\min(x_1, y_1) > 0$. At least one such node exists and if there is another node in state $\langle x_2, 0\rangle$ for any x_2, then again with probability at least $1/n^2$ a split reaction occurs in the next round. The case of $x < y$ is analogous and we get that during the first phase, if there is no pair whose interaction would result in a split reaction, all nodes must be in states $\langle x, y\rangle$ with $\min(x, y) > 0$, i.e. in mixed states. By Claim B.3, with probability at least $\frac{1}{2(\log n - 1)}$

a pure node appears after the next communication round and by the above argument, if the first phase has not been completed, in the subsequent round a split reaction will occur with probability at least $1/n^2$. Therefore, during the first phase, the expected number of rounds until the next split reaction is at most $4n^2(\log n - 1)$. By Claim B.5, there can be at most $2n^2$ split reactions in any execution, thus the expected number of communication rounds in the first phase is at most $8n^4(\log n - 1)$.

The second phase starts immediately after the first, and ends when no node encodes a strictly negative value. Note that if this was already true when the first phase ended, then the second phase is trivially empty. Consider the other case when all nodes encode values -1, 0 and 1 at the beginning of the second phase. Under these circumstances, because of the update rules, no node will ever be in a state $\langle x, y\rangle$ with $\max(x, y) > 1$ in any future configuration. Also, the number of nodes encoding non-zero values can only decrease. In each round, with probability at least $1/n^2$, two nodes with values 1 and -1 interact, becoming $\langle 0, 0\rangle^+$ and $\langle 0, 0\rangle^-$. Since this can only happen $n/2$ times, the expected number of communication rounds in the second phase is at most $n^3/2$.

The third phase lasts until the system converges, that is, until all nodes with value 0 are in state $\langle 0, 0\rangle^+$. By Invariant B.2, $S > 0$ holds throughout the execution, so there is at least one node with a positive sign and non-zero value. There are also at most $n - 1$ *conflicting* nodes with negative sign, all in state $\langle 0, 0\rangle^-$. Thus, independently in each round, with probability at least $1/n^2$, a conflicting node meets a node with strictly positive value and becomes $\langle 0, 0\rangle^+$, decreasing the number of conflicting nodes by one. The number of conflicting nodes can never increase and when it becomes zero, the system has converged to the desired configuration $\hat{c}$. Therefore, the expected number of rounds in the third phase is at most n^3.

Combining the results and using the linearity of expectation, the total expected number of communication rounds before reaching $\hat{c}$ is at most $n^3(8n(\log n - 1) + 1/2 + 1) \leq n^5$ for sufficiently large n. Finite expectation implies that the algorithm converges with probability 1. Finally, when two nodes with positive sign meet, they both remain positive, so any configuration c_e reachable from $\hat{c}$ has the correct signs. □

Claim B.7. *Let $w > 1$ be the maximum level among the nodes with a negative (resp., positive) sign. There is a constant β, such that after $\beta n \log^2 n$ positive-rounds (resp., negative-rounds) the maximum level among the nodes with a negative (resp., positive) sign will be at most $\lfloor w/2 \rfloor$ with probability at least $1 - \frac{1}{n^5}$.*

Proof. We will prove the claim for nodes with negative values. (The converse claim follows analogously.) Fix a round r, and recall that $w > 1$ is the maximum level of a node with a negative value at the beginning of the round. Let U be the set of all nodes with negative values and the same level w at the beginning of the round, and let $u = |U|$. We call these nodes *target nodes*.

By the structure of the algorithm, the number of target nodes never increases and decreases by one in every *eliminating* round where a target node meets a pure node with a non-negative value, due to a split or cancel reaction. Consider a set of $\alpha n \log n$ consecutive positive-rounds after r, for some constant $\alpha > 2^{12}$. In each round, if there are still at least $\lceil u/2 \rceil$ target nodes, then the probability of this round being eliminating is at least $\frac{\lceil u/2 \rceil}{2^{12} n \log n}$ (since in a positive round at least half of $\frac{n}{2^{11} \log n}$ pure nodes have non-negative value). Let us describe the process by considering a random variable $Z \sim \mathrm{Bin}(\alpha n \log n, \frac{\lceil u/2 \rceil}{2^{12} n \log n})$, where each success event corresponds to an eliminating round. By a Chernoff Bound, the probability of having $\alpha n \log n$ iterations with at most $\lceil u/2 \rceil$ eliminations is at most:

$$\Pr\left[Z \leq \lceil u/2 \rceil\right] = \Pr\left[Z \leq \frac{\alpha \lceil u/2 \rceil}{2^{12}} \left(1 - \frac{\alpha - 2^{12}}{\alpha}\right)\right] \leq \exp\left(-\frac{\alpha \lceil u/2 \rceil (\alpha - 2^{12})^2}{2^{13} \alpha^2}\right)$$

For sufficiently large α and $u \geq \log n$, the probability of this event is at most $\frac{1}{n^6}$ for $\alpha n \log n$ positive-rounds. Applying the same rationale iteratively as long as $u \geq \log n$, we obtain by using a Union Bound that the number of target nodes will become less than $\log n$ within $\alpha n \log n(\log n - \log\log n)$ positive-rounds, with probability at least $1 - \frac{\log n - \log\log n}{n^6}$.

Finally, we wish to upper bound the remaining number of positive-rounds until no target node remains. Again for sufficiently large α, but when $u < \log n$, we get from the same argument as above that the number

of target nodes is reduced to $\lfloor u/2 \rfloor$ within $\frac{\alpha n \log^2 n}{u}$ consecutive positive-rounds with probability $1/n^6$. So we consider increasing numbers of consecutive positive-rounds, and obtain that no target nodes will be left after at most $\alpha n \log n + 2\alpha n \log n + \ldots + \alpha n \log^2 n \leq 2\alpha n \log^2 n$ positive-rounds, with probability at least $1 - \frac{\log \log n}{n^6}$, where we have taken the union bound over $\log \log n$ events. The original claim follows by setting $\beta = 3\alpha$, taking Union Bound over the above two events ($u \geq \log n$ and $u < \log n$) and $\log n \leq n$. □

Claim B.8. *There exists a constant β, such that if during the first $2\beta n \log^3 n$ rounds the number of pure nodes is always at least $\frac{n}{2^{11} \log n}$, then with probability at least $1 - \frac{2 \log n}{n^5}$, one of the following three events occurs at some point during these rounds:*

1. *Nodes only encode values in $\{-1, 0, 1\}$;*
2. *There are less than $\frac{n}{2^{12} \log n}$ nodes with non-positive values, all encoding 0 or -1,*
3. *There are less than $\frac{n}{2^{12} \log n}$ nodes with non-negative values, all encoding 0 or 1.*

Proof. We take a constant β that works for Claim B.7. Since there are at least $\frac{n}{2^{11} \log n}$ pure node at all times during the first $2\beta n \log^3 n$ rounds, each round during this interval is a negative-round, a positive-round, or both. We call maximum positive (resp. negative) level the maximum level among all the nodes encoding non-negative (resp. non-positive) values. Unless the maximum positive level in the system is ≤ 1, by Claim B.7, a stretch of $\beta n \log^2 n$ negative-rounds halves the maximum positive level, with probability at least $1 - \frac{1}{n^5}$. The same holds for stretches of $\beta n \log^2 n$ positive-rounds and the maximum negative level.

Assume that none of the three events hold at any time during the first $2\beta n \log^3 n$ rounds. In that case, each round can be classified as either:

- a negative-round where the maximum positive level is strictly larger than 1, or
- a positive-round where the maximum negative level is strictly larger than 1.

To show this, without a loss of generality consider any positive-round (we showed earlier that each round is positive-round or a negative-round). If the maximum negative level is > 1 then the round can be classified as claimed, thus all non-positive values in the system must be 0 or -1. Now if there are less than $\frac{n}{2^{12} \log n}$ such nodes, then we have the second event, so there must be more than $\frac{n}{2^{12} \log n}$ nodes encoding 0 and -1. However, all these nodes are pure, so the round is simultaneously a negative-round. Now if the maximum positive level is > 1 then the round can again be classified as claimed, and if the maximum positive level is at most 1, then all nodes in the system are encoding values -1, 0 or 1 and we have the first event.

Thus, each round contributes to at least one of the stretches of $\beta n \log^2 n$ rounds that halve the maximum (positive or negative) level, w.h.p. However, this may happen at most $2 \log n$ times. By applying Claim B.7 $2 \log n$ times and the Union Bound we get that after the first $2\beta n \log^3 n$ rounds, with probability at least $1 - \frac{2 \log n}{n^5}$ only values -1, 0 and 1 may remain. However, this is the same as the first event above. Hence, the probability that none of these events happen is at most $\frac{2 \log n}{n^5}$. □

Claim B.9. *Consider a configuration where all nodes with strictly negative values encode -1, while at least $\frac{2n}{3}$ nodes encode strictly positive values. The number of rounds until convergence is $O(n \log n)$ in expectation and $O(n \log^2 n)$ with high probability.*

Proof. In any configuration, let us call *conflicting* any node that encodes -1, and *target* node any node that has a strictly positive value. Because of the structure of the algorithm, and that in configuration c the only nodes with non-positive sign encode -1 or 0, in all configurations reachable from c nodes with negative values will also only encode -1 or -0. Moreover, the number of conflicting nodes can never increase after an interaction. Observe that the number of conflicting nodes decreases by one after an interaction where a target node (with a stritly positive value) meets a node with value -1, while the number of target nodes may also decrease by at most 1. This is because a split reaction happens on the positive component of the target node (since the positive component of the conflicting node is 0) and both nodes get value ≥ 0 after

the interaction. There are at least $n/3$ more target nodes than conflicting nodes in c, therefore, in every later configuration, there must always be at least $n/3$ target nodes.

Let us estimate the number of rounds until each conflicting node has interacted with a target node, at which point no more conflicting nodes may exist. Let us say there were x conflicting nodes in configuration c. The expected number of rounds until the first conflicting node meets a target node is at most $\frac{3n}{x}$, since the probability of such an interaction happening in each round is at least $\frac{x}{n} \cdot \frac{n}{3n}$. The expected number of rounds for the second node is then $\frac{3n}{(x-1)}$, and so on. By linearity of expectation, the expected number of rounds until all conflicting nodes are eliminated is $O(n \log x) \leq O(n \log n)$.

At this point, all nodes that do not have a positive sign must be in state $\langle 0, 0 \rangle^-$. If we redefine *conflicting* to describe these nodes, it is still true that an interaction of a conflicting node with a target node brings the conflicting node to state $\langle 0, 0 \rangle^+$, decreasing the number of conflicting nodes. As we discussed at least $n/3$ target nodes are still permanently present in the system. By the structure of the algorithm no interaction can increase the number of conflicting nodes, and the system converges when all conflicting nodes are eliminated. This takes expected $O(n \log n)$ rounds by exactly the same argument as above.

To get the high probability claim, simply observe that when there are x conflicting nodes in the system, a conflicting node will interact with a target node within $\frac{3nO(\log n)}{x}$ rounds, with high probability. The same applies for the next conflicting node, etc. Taking Union Bound over these events gives the desired result. □

C Synthetic Coins

Claim C.1. $\mathbb{E}[X_{i+m} \mid X_i = x] = n/2 + (1 - 4/n)^m \cdot (x - n/2)$.

Proof. If two agents both with coin values one are selected, the number of ones decreases by two. If both coin values are zero, it increases by two, and otherwise stays the same. Hence, we have that

$$\begin{aligned}\mathbb{E}[X_{i+m} \mid X_{i+m-1} = t] &= (t-2) \cdot \Pr[X_{i+m} = t-2] + t \cdot Pr[X_{i+m} = t] + (t+2) \cdot \Pr[X_{i+m} = t+2] \\ &= (t-2) \cdot \frac{t(t-1)}{n(n-1)} + t \cdot \frac{2t(n-t)}{n(n-1)} + (t+2) \cdot \frac{(n-t)(n-t-1)}{n(n-1)} \\ &= t + \frac{2}{n(n-1)} \cdot \left(n^2 - 2nt - n + 2t\right) = t \cdot \left(1 - \frac{4}{n}\right) + 2\end{aligned}$$

Thus, we get a recursive dependence $\mathbb{E}[X_{i+m}] = \mathbb{E}[X_{i+m-1}] \cdot (1 - 4/n) + 2$, that gives

$$\mathbb{E}[X_{i+m}] = 2 \cdot \sum_{j=0}^{m-1} \left(1 - \frac{4}{n}\right)^j + \mathbb{E}[X_i] \cdot \left(1 - \frac{4}{n}\right)^m = \frac{n}{2} + \left(1 - \frac{4}{n}\right)^m \left(x - \frac{n}{2}\right)$$

by telescoping. □

D Analysis of the Leader Election Algorithm

Lemma D.1. *All nodes can never be minions. A configuration with $n-1$ minions must have a stable leader, meaning that the non-minion node will never become a minion, while minions will remain minions.*

Proof. Assume for contradiction that all nodes are minions at some time T, and let u be a maximum $(.payoff, .level)$ pair (lexicographically) among all the minions at this time. No node in the system could ever have had a larger pair, because no interaction can decrease a pair. The minions only record the values of such pairs they encounter, and never increase them, so there must have been a contender in the system with a payoff and level pair u that turned minion by time T. Among all such contenders, consider the one that turned minion the last. It could not have interacted with a minion, because no minion (and no node) in the system ever held a larger pair. On the other hand, even if it interacted with another contender, the contender also could not have held a larger pair. Thus, it could only have been an interaction with another contender, that held the same pair and a larger coin value used as a tie-breaker. However, that interaction partner would

remain a contender and and must have turned minion later, contradicting our assumption that the interaction we considered was the last one where a contender with a pair u got eliminated.

By the structure of the algorithm, minions can never change their mode. In any configuration with $n-1$ minions, the only non-minion must remain so forever, and thus be a stable leader, because otherwise we would get n minions and violate the above argument. □

Lemma D.2. *Assume that the maximum number that the parameter $payoff$ can hold is $\sqrt{m}$. Then, the expected number of interactions until all nodes have $.mode =$ tournament or $.mode =$ minion is $O(n \log^2 n)$. Moreover, once that happens, with probability at least $1 - 5/n^3$, at most $12 \log n$ nodes will hold the maximum $payoff$ value, which will be at least $\log n/4$ and at most $16 \log n$.*

Proof. During the first $2n$ interactions, any given agent is expected to interact twice. The probability that during this period, a given agent interacts more than $4 \log n$ times is at most $1/n^4$. Taking an union bound, all agents interact at most $4 \log n$ times with probability at least $1 - 1/n^3$.

Let us call the interactions after the first $2n$ the *relevant* interactions. Consider the relevant for any fixed agent. By Theorem 4.1, during any later interaction, with probability at least $1 - \exp(-n/4)$, there are at least $n/3$ and at most $2n/3$ agents holding each possible coin value. Taking an union bound, this holds for all of the first n^4 relevant interactions with probability at least $1 - \frac{n^4}{\exp(n/4)}$. From now on, let us assume this high probability event.

During any later interaction, with probability at least $(n/3 - 1)/n > 1/4$ the agent observes 0 and changes its $mode$ to the tournament. Hence, the probability that it increases its payoff more than $12 \log n$ times is at most $(3/4)^{12 \log n} \leq 1/n^4$. Taking the union bound over all the agents gives that the increase in payoffs in the later interactions for all agents are less than $12 \log n$ with probability at least $1 - 2/n^3$. In total, with probability at least $1 - 3/n^3$, all agents will have payoffs at most $16 \log n$.

Since every agent stays in seeding mode for four interactions, we can find at least $n/2$ agents, who move to $lottery$ mode after their first $2n$ interactions. Consider any one of the $n/2$ agents. By assumption, the agent will have probability at least $1/4$ of finalizing the $payoff$ and moving to tournament mode. The probability that the payoff of this agent will be larger than $\log n/4$ is thus at least $1/\sqrt{n}$. If the payoff is indeed larger, we are done, otherwise, we can find another agent among the $n/2 - \log n/4$ whose interactions we have not yet considered, and analogously get that with probability at least $1/\sqrt{n}$, it would get a payoff at least $\log n/4$. We can continue this process, and will end up with about $2n/\log n$ agents, whose interactions were completely independent, and because of the bias, each of them had a probability of at least $1/\sqrt{n}$ of getting a larger payoff than $\log n/4$. If we described this process as a random variable $\mathrm{Bin}\left(\frac{2n}{\log n}, \frac{1}{\sqrt{n}}\right)$, we get by the Chernoff Bound that the probability of no node actually getting more than $\log n/4$ payoff must be extremely low (because the expectation is $2\sqrt{n}/\log n$), in particular, lower than $1/n^3$.

Again, considering all the high probability events from above, we know that the maximum payoff in the system is between $\log n/4$ and $16 \log n$. Consider any fixed payoff k in this interval, and let us say it is the maximum. Then, any agent that reaches this payoff, has to flip 0, but they might flip 1 with probability at least $1/4$. Thus, the probability that at least $12 \log n$ agents will stop exactly at payoff k is at most $(3/4)^{4 \log n} \leq 1/n^4$. Taking the union bound over at most $16 \log n < n$ payoffs, and the above high probability events, we get that with probability at most $1 - 5/n^3$, at most $12 \log n$ agents will have the maximum payoff, which will be between $\log n/4$ and $16 \log n$. Next, we look at the expected maximum payoff.

With probability at most $\frac{n^4}{\exp(n/4)}$, the maximum payoff in the system can be as high as $\sqrt{m}$. Otherwise, all coin flips have the bias in $[1/4, 3/4]$ during the n^4 later interactions. Moreover, the same argument as above shows that the probability that the maximum payoff among all agents is larger than k is at most $n \cdot (3/4)^k \leq n \cdot 2^{-k/3}$. This means that with extra probability of at most $2^{-n^4/3}$, the maximum payoff in the system can be as much as $\sqrt{m}$. When this does not happen and we have proper bias during the n^4 later

interactions, the expected maximum payoff is at most $16\log n + n \cdot \sum_{i=16\log n+1}^{n^4} 2^{-i/3} = O(\log n)$. Thus, the expected maximum payoff is actually $O(\log n) + \sqrt{m} \cdot (n^4 \cdot \exp(-n/4) + 2^{-n^4/3}) = O(\log n)$.

Assuming that the maximum payoff is k, the expected number of interactions before every node has left seeding or lottery mode is of at most $2n(k+4)\log n$. We can count this as the expected number of interactions until every node has interacted $k+4$ times, as, once that has happened, no node can be in seeding or lottery mode. The reason is that seeding only lasts four interactions per node, and no node did more than k interactions in lottery, because then they would have a larger payoff. The expected number of interactions before every node interacts once is $1 + n/(n-1) + n/(n-2) + \ldots + n \leq 2n\log n$. The expected number of interactions before every node interacts $k+4$ times is at most the amount of interactions it would take for every node to interact once, then reset and count interactions until they would all interact again, etc, $k+4$ times, which gives $2n(k+4)\log n$. If maximum payoff is k_i with probability p_i, then the expected number of interactions until all nodes are in the tournament or minion mode is then at most order of $\sum 2p_i k_i n \log n$ which is the same as $2n\log n$ multiplied by the expected maximum payoff, which we know is $O(\log n)$. Thus, we have shown that the expected number of interactions until no node is in the seeding or lottery mode is $O(n\log^2 n)$. □

Lemma D.3. *With probability at least $1 - \frac{O(\log n)}{n^3}$, only one contender reaches level $\ell = \frac{3\log n}{\log 18 + \log\log n}$, and for each level up to ℓ, it takes at most $O(n \cdot \log^9 n)$ interactions before some contender gets to a larger level. Conditioned on this high probability event, the expected number of interactions before having $n-1$ minions is $O(n\log^9 n)$.*

Proof. By our assumption on m, it holds that $\frac{m}{3\log m} \geq \frac{3\log n}{\log 18 + \log\log n}$, so this value of level can always be reached. We will assume the high probability case in Lemma D.2, which occurs with probability $\geq 1 - 5/n^3$. Hence, we have to prove that the probability that either more than one competitor reaches level ℓ is at most $\frac{2+O(\log n)}{n^3}$.

Consider some competitor v which just increased the maximum level among competitors in the system. Until some other competitor reaches the same level, v will turn every interaction partner into its minion. Furthermore, as in epidemic spreading, these minions will also turn their interaction partners into minions of the highest level contender v. Let the $.payoff, .level$ pair of this competitor be u. Also, we call a node whose pair also at least u an *up-to-date* node; the node is *out-of-date* otherwise. Initially, only the contender v that reached the maximum level is up-to-date.

We will show that if in some configuration $x < n$ nodes are up-to-date, after a *phase* of $\frac{16n(n-1)\log n}{4x(n-x)}$ interactions, at least $x+1$ nodes will be up-to-date with probability at least $1 - \frac{1}{n^4}$. Up-to-date nodes may never become out-of-date. On the other hand, an out-of-date node becomes up-to-date itself after an interaction with any up-to-date node. If we have x up-to-date nodes, in each round, the probability that an out-of-date node interacts with an up-to-date node increasing the number of up-to-date nodes to $x+1$, is $\frac{2x(n-x)}{n(n-1)}$. To upper bound the probability that such an interaction never happens during a phase of $\frac{16n(n-1)\log n}{4x(n-x)}$ rounds, we can consider a random variable $Y \sim \mathrm{Bin}\left(\frac{16n(n-1)\log n}{4x(n-x)}, \frac{2x(n-x)}{n(n-1)}\right)$ and establish that:

$$\Pr[Y \leq 0] \leq 2^{-4\log n} < \frac{1}{n^4},$$

An Union Bound over at most n phases gives that with probability at least $1 - 1/n^3$, after at most

$$\sum_{x=1}^{n-1} \frac{16n(n-1)\log n}{4x(n-x)} \leq \frac{16(n-1)\log n}{4} \sum_{x=1}^{n-1} \left(\frac{1}{x} + \frac{1}{n-x}\right) \leq 16n\log^2 n$$

rounds, all nodes will have value at least u. Taking an union bound over all possible levels $\ell < \log n$, we get that with probability at least $1 - \log n/n^3$, once a contender reaches some level, unless some other contender

reaches the same level within the next $16n \log^2 n$ interactions, the original node will turn every other node into minions and become a stable leader.

Once some contender has increased the maximum level, it needs to observe $6 \log \log n + 12$ consecutive ones to increase a level. W.h.p, the probability at each iteration is at least $(1/2 - 1/16)$ for each observed coin, so the probability that a stretch of $6 \log \log n + 12$ consecutive interactions results in a level increase is at least $\Theta(1/\log^{7.8} n)$. If we consider an interval containing $O(n \log^9 n)$ interactions. Then, with high probability, the node will perform a level increase for at least $\log^9 n / \log^8 n$ times during such a stretch, unless it has already become a minion.

For the rest of the argument, we will bound the probability that any of the other contenders, whose number is at most $16 \log n$ by Lemma D.2, will be able to increment their level during an arbitrary stretch of $6 \log \log n + 12$ consecutive interactions. This probability will be low, and therefore it is likely that the process will terminate after a level increase.

More precisely, once a new level is reached after a level increment, the nodes have $16n \log^2 n$ interactions to increment to the same level, or they will soon all become minions. To do so, they should all have at least one iteration of observing $6 \log \log n + 12$ consecutive ones, because all contenders have the maximum payoff from the first stage; by Lemma D.2, the maximum payoff was at least $\log n/4$.

Hence, there can be at most $16 \log n \cdot \Theta(\log^2 n / \log \log n)$ such interaction intervals, with probability at least $1 - 1/n^3$. Each interaction interval has probability at most $(1/2 + 1/16)^{6 \log \log n + 12} \leq 1/\log^4 n$ of success. Hence, by taking sufficiently large n, we can make the expectation of the number of successful iterations be less than $1/\log n$ divided by a large constant. More precisely, by fixing the constants, we can show that the probability that even one of the iterations is successful is at most $\frac{1}{18 \log n}$.

Hence, the probability that there is no second survivor among contenders at each level (which would correspond to a stable leader being elected) is at most $1/18 \log n$, every time the maximum level is incremented. The probability that this does not happen for all $\ell = \frac{3 \log n}{\log 18 + \log \log n}$ levels is then at most $\left(\frac{1}{18 \log n}\right)^{\ell} \leq \frac{1}{n^3}$.

On the other hand, clearly since with more than constant probability a single contender remains after each level, only constantly many levels will be used in expectation. Moreover, as we have seen above, any contender with the maximum level has at least $\Theta(1/\log^{7.8} n)$ probability at each interval to increase the level. Therefore, the maximum level is expected to increase every $n \log^8 n \log \log n$ interactions, and the maximum reachable level should be attained after $O(n \log^9 n)$ interactions. The claim then follows. □

Corollary D.4. *The algorithm converges in expected parallel time $O(\log^9 n \log \log n)$ and with high probability in parallel time $O(\log^{10} n)$.*

Proof. The combination of Lemma D.2 and Lemma D.3 gives the high probability claim, combined with the observation that when the second stage of the algorithm starts, all non-contenders become minions within parallel time $O(\log^2 n)$ with high probability, because they get exposed to higher payoff values. The formal proof of this is exactly the same as the argument in Lemma D.3, that shows that once a contender gets to a new maximum level, the information about it is propagated to the whole system within $O(n \log^2 n)$ interactions with high probability.

To get the bound on expectation, first we sum up the expectations from both stages, and get the dominant term $O(\log^9 n)$ parallel time. Then similar to above, we add the expected parallel time after the start of the second stage to the point when all non-contenders become minions, i.e. when all nodes with less than the maximum payoff learn about a larger value. This takes $O(\log n)$ parallel time due to the following reason. Call the nodes that know about the maximum payoff value a *up-to-date*, and *out-of-date* otherwise. At time T, at least one node is up-to-date. Before an arbitrary interaction round where we have x up-to-date nodes, the probability that an out-of-date node interacts with an up-to-date node, increasing the number of up-to-date nodes to $x+1$, is $\frac{2x(n-x)}{n(n-1)}$. By a Coupon Collector argument, the expected number of rounds until every node is up-to-date is then $\sum_{x=1}^{n-1} \frac{n(n-1)}{2x(n-x)} \leq \frac{(n-1)}{2} \sum_{x=1}^{n-1} \left(\frac{1}{x} + \frac{1}{n-x}\right) \leq 2n \log n$.

Finally, we need to incorporate the expected time in the low probability event. Notice that the expectation in Lemma D.2 is not conditioned, so we only need to care about the low probability events that happen during the second stage. But recall that, during the second stage, non-minions can always eliminate each other in direct interactions comparing their payoffs, levels, and the coin as a tie-breaker. So, for any given two non-minion nodes x and y, in every interaction round, there is a probability of at least $1/n^2$ that they meet, and one could eliminate each other for certain if they had different coin values. If not, then with probability at least $1/n$, one of the nodes, say x, interacts with some other node in this interaction, and then immediately afterward, interacts with y, this time with different coin values. Hence, in every two rounds, with probability at least $1/n^3$, the number of contenders decreases by at least one. Hence, even in the low probability case, the expectation is at most $O(n^3)$, which does not affect the dominant term of the $O(\log^9 n)$ time. □