Title
Computation in population protocols: exact majority, uniform computation, and the dynamic model

Author
Eftekhari, Mahsa

Publication Date
2022

Peer reviewed|Thesis/dissertation

Computation in population protocols:
exact majority, uniform computation, and the dynamic model

By

MAHSA EFTEKHARI

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

David Doty, Chair

Vladimir Filkov

Mohammad Sadoghi

Committee in Charge

2022

To all students on PS752 whose journeys ceased.

Contents

Abstract

We study population protocols, a model of distributed computing appropriate for modeling well-mixed chemical reaction networks and other physical systems where agents exchange information in pairwise interactions, but have no control over their schedule of interaction partners.

Concretely, *population protocols* [10] are asynchronous, complete networks that consist of computational entities called *agents* with no control over the schedule of interactions with other agents. In a population of n agents, repeatedly a random pair of agents are chosen to interact, each observing the state of the other agent before updating its own state.

Population protocols are equivalent to the model of chemical reaction networks, describing abstract chemical reactions such as $A + B \to C + D$, when the latter is subject to the restriction that all reactions have two reactants and two products, and all rate constants are 1.

In this thesis, we demonstrate an algorithmic advance for the exact majority problem: starting in a population of n agents, each with one of two opinions A or B, the agents must determine whether there are more A, more B, or a tie. The proposed algorithm is *nonuniform*: assumes that the agents are initialized with an approximate estimate of n (specifically, a value that is $\geq \lfloor \log n \rfloor$).

We introduce *uniform* population protocols: networks of anonymous agents whose pairwise interactions are chosen at random, where each agent uses an *identical* transition algorithm that does not depend on the population size n. Moreover, we study almost optimal protocols that solve static counting problem: the *counting* problem is that of designing a protocol so that n agents, all starting in the same state, eventually converge to states where each agent encodes in its state an exact or approximate description of population size n.

Finally, we introduce population protocols with dynamic size and finish this thesis with a novel counting protocol robust to an adversary who can add or remove agents.

Acknowledgments

I appreciate my supervisor, Dave Doty, for his help in many ways. For letting me pursue a research area that I am passionate about, being an excellent coauthor, mentor, and supervisor while respecting me as a colleague, always open to discussing research, and supporting me generously throughout my entire Ph.D.

I am grateful to all the many coauthors that were part of this work: Talley Amir, James Aspnes, Leszek Gąsieniec, Othon Michail, Eric Severson, Paul G. Spirakis, Grzegorz Stachowiak, Michail Theofilatos, and Przemyslaw Uznański. I appreciate the community in the computer science department at UC Davis: all the staff, professors, classmates, and friends who were always there for me during this journey.

I am also very thankful to my supportive brother, Mohsen Eftekhari, who always supports and encourages me during graduate school and life challenges.

The financial support for the work in this dissertation came from NSF award 1900931 and CAREER award 1844976.

Previously published work

Parts of this thesis is based on previously published papers that are listed below. All author names are sorted in alphabetical order.

- Dynamic size counting in population protocols. David Doty, Mahsa Eftekhari. In the 1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2022).
- A Time and Space Optimal Stable Population Protocol Solving Exact Majority. David Doty, Mahsa Eftekhari, Leszek GKasieniec, Eric Severson, Grzegorz Stachowiak, and Przemysław Uznański. In the 62nd Annual of IEEE Symposium on Foundations of Computer Science (FOCS 2021).
- A survey of size counting in population protocols. David Doty, Mahsa Eftekhari. Theoretical Computer Science Journal (TCS 2021).
- Message complexity of population protocols. Talley Amir, James Aspnes, David Doty, Mahsa Eftekhari, and Eric Severson. In the 34th International Symposium on Distributed Computing (DISC 2020).
- Efficient size estimation and impossibility of termination in uniform dense population protocols. David Doty, Mahsa Eftekhari. In the 38th ACM Symposium on Principles of Distributed Computing (PODC 2019).
- Brief announcement: Exact size counting in uniform population protocols in nearly logarithmic time. David Doty, Mahsa Eftekhari, Othon Michail, Paul G. Spirakis, and Michail Theofilatos. In the 32nd International Symposium on Distributed Computing (DISC 2018).

CHAPTER 1

Introduction

1.1. Introduction to population protocols

With recent advances in DNA computation,[1] scientists are on the verge of developing smart molecules that can communicate, perform tasks, and control matter at the nano level. For example, intelligent molecules with self-counting ability could improve medical diagnostic tests. I do not look at molecular computing as a model that competes with the traditional silicon based on speed or other measures of computational power but as a direct molecular manipulation that can conduct computation at the molecular level. So, we can use them as biosensors to detect cellular irregularities and combine them with our existing detection and treatment techniques for various medical conditions.

To help biological computation scientists achieve their goals, we study population protocols: an abstract distributed computing model of computation that models the natural physical systems and captures their dynamics. Population protocols are an appropriate model for electronic computing scenarios such as sensor networks and for "fast-mixing" physical systems such as animal populations [**93**], gene regulatory networks [**33**], and chemical reactions [**87**], the latter increasingly regarded as an implementable "programming language" for molecular engineering, due to recent experimental breakthroughs in DNA nanotechnology [**39**, **88**].

A population is a network of n anonymous and identical entities (called *agents*) each holding a *state* representing its entire memory.[2] The agents communicate through a sequence of randomly chosen pairwise interactions, i.e., with no control over whom they interact.

[1] Microsoft implemented a DNA version of the approximate majority protocol on a mixed population of DNA agents [**39**].

[2] Using message-passing terminology, each agent sends its entire state of memory as the message.

The computation happens through a sequence of pairwise interactions among randomly chosen pairs of agents [10]. Concretely, in an interaction, the scheduler selects two different agents uniformly at random. Each agent observes the other agent's entire memory (state) and updates its state according to the transition function defined by a protocol.

A protocol is defined with 2-input, 2-output *transitions* such as $(a, b) \to (c, d)$ that determine if two agents with the state a and b interact should update their state to c and d, respectively.

In the original model of population protocols [10], the states and transitions are constant with respect to the population size n. However, recent studies use a variant of the model: allowing the number of states and transitions to grow with n [2,3,5,20,24,25,26,27,28,30,34,47,50,51,57,60, 61, 79, 91], leading to time- and space-optimal solutions for leader election [26] and majority [50] problems.

One motivation to study population protocols with $\omega(1)$ states is the existence of impossibility results showing that no constant-state protocol can stabilize in sublinear time with probability 1 for problems such as leader election [53], majority [2], or computation of more general predicates and integer-valued functions [18]. Note that the known time lower bounds [2, 19, 53] are on stabilization, not convergence time. Intuitively, we say a protocol *converges* when it reaches the correct output without subsequently changing it—though it may remain change*able* for some time after converging—whereas it *stabilizes* when the output becomes unchangeable. See [25, 53] for a discussion of the distinction between stabilization and convergence.

For a concrete example, consider a simple leader election transition is $(L, L) \to (L, F)$ with all agents starting in state L. This protocol *stabilizes*, meaning that with probability 1, it reaches a configuration that is both correct (has exactly one agent in state L) and *stable*: every subsequently reachable configuration is also correct.

Recently Kosowski and Uznanski [69] achieved a breakthrough result, showing constant-state protocols for leader election and all decision problems computable by population protocols (the *semilinear* predicates), which converge with high probability in polylog(n) time, and for any $\epsilon > 0$, probability 1 protocols for the same problems that converge in $O(n^\epsilon)$ expected time. The latter protocols require $\Omega(n)$ time to stabilize, as would any constant-state protocol due to the cited time lower bounds.

1.1.1. A note on uniform computation. Most of the recent algorithmic advances using non-constant states [2, 3, 5, 20, 24, 25, 26, 27, 30, 50, 61, 79, 91] propose a nonuniform protocol. Rather than being a single set of transitions, a nonuniform protocol represents a *family* of protocols, where the set of transition rules (i.e., the protocol) used is allowed to depend on the population size n.

The typical way that nonuniformity appears in a protocol is that the agents have an estimate of n (for example the value $\lceil \log n \rceil$) appearing in the transitions. [3] When expressed in pseudocode, this is often realized by a "hard-coded" constant $\lceil \log n \rceil$ to which the code has access (i.e., each agent receives the value $\lceil \log n \rceil$ as "advice"). However, in measuring the state complexity of the protocol, the space required to store this value does not count against the memory usage. Note that this concern is relevant because we count memory complexity by counting the number of states, rather than the number of bits necessary to represent the state. Adding a field with $O(\log n)$ different values does not asymptotically change the bit usage, but it does asymptotically increase the number of states. As an example of a nonuniform protocol using its estimate of $\log n$, consider the following nonuniform leaderless phase clock rules: each agent, independently, counts its number of interactions and uses it as a timer by comparing to $12 \ln n$. In this protocol, each agents increment the value in its `count` field until it reaches a threshold dependent on $\ln n$ (note that "$-$" stands for an arbitrary value in $\mathbb{N}$ for `count`):

$$\begin{cases} (\mathsf{A} = \mathsf{F}, \texttt{count} = i), (-,-) \longrightarrow (\mathsf{A} = \mathsf{F}, \texttt{count} = i+1), (-,-) & \textbf{if } i < 12 \ln n \\ (\mathsf{A} = \mathsf{F}, \texttt{count} = i), (-,-) \longrightarrow (\mathsf{A} = \mathsf{T}, \texttt{count} = 0), (-,-) & \textbf{if } i = 12 \ln n \end{cases} \tag{1.1}$$

In the above leaderless phase clock, no agent will set their A field to T before $6 \ln n$ time has passed with probability at least $1 - 1/n$.[4] However, no uniform protocol can achieve this same task: in any uniform protocol, some agent will set its A field to T in constant time with high probability [47, Theorem 4.1].

In a *uniform* protocol, by contrast, the transitions are not dependent on the population size n, i.e., agents lack any knowledge of n.

[3] Some of these protocols crucially rely on an estimate of $\log n$ for correctness.

[4] We can compute the error probability with a straightforward Chernoff bound application on binomial random variables.

The original $O(1)$-state model [10, 11, 12] is uniform, since there is a single transition function for all population sizes. However , uniform protocols are not required to have constant states. For example, starting from n agents in state L_1, the protocol defined by transitions $L_i, L_j \to L_{i+j}, F$ for all natural numbers i, j, in a population of size n, can produce all values of L_i for i between 1 and n. However, note that the (infinitely many) transitions are "uniformly specified": no transition makes reference to an estimate of n. (This is formalized by requiring the transition function to be computable by a single Turing machine; see section 1.2 for more details.)

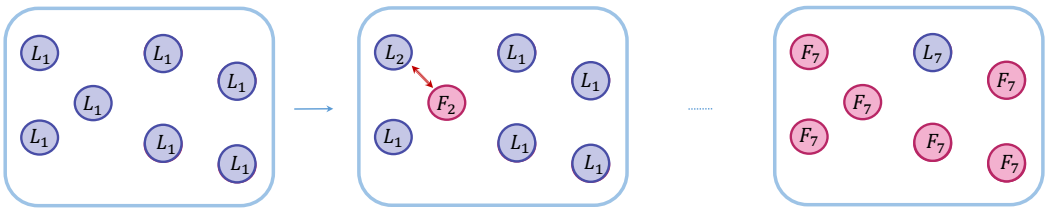


FIGURE 1.1. Example of a uniform protocol that solves the exact counting problem in which the agents calculate the population size n. The protocol $(L_i, L_j) \longrightarrow (L_{i+j}, F_{i+j})$ and $(F_i, F_j) \longrightarrow (F_{\max(i,j)}, F_{\max(i,j)})$ counts the population size starting from all agents in state L_1. In a population of size n, the agents eventually reach a configuration with one agent in the leader state and $n-1$ agents in the follower state, all having the same value n.

Throughout this thesis, n denotes the number of agents in the population. Also, we define the *(parallel) time* for some event to happen as the number of interactions, divided by n, until the event happens.[5] This represents a natural model of time complexity in which each individual agent experiences expected $O(1)$ interactions per unit time, hence across the whole population, $\Theta(n)$ total interactions occur per unit time. All references to "time" in this thesis refer to parallel time.

All problems solvable with zero error probability by a constant-state population protocol are solvable in $O(n)$ time [12, 52]. The benchmark for "efficient" computation is thus sublinear time, ideally polylog(n), with $\Omega(\log n)$ time a lower bound on most nontrivial computation, since a simple coupon collector argument shows that is the time required for each agent to have at least one interaction.

[5] A simplified approximation for modeling well-mixed systems assumes all pairs of agents have equal probability.

To have probability 0 of error, a protocol must eventually *stabilize*: reach a configuration where all agents agree on the correct output, which is *stable*, meaning no subsequently reachable configuration can change the output.[6]

As a simple example of time complexity, suppose we want to design a protocol to decide whether at least one x exists in an initial population of x's and q's. The single transition $x, q \to x, x$ indicates that if agents in states x and q interact, the q agent changes state to x. If x outputs "yes" and q outputs "no", this takes expected time $O(\log n)$ to reach a consensus of all x's (i.e., $O(n \log n)$ total interactions, including null interactions between two x's or between two q's). However, the transitions $x, x \to y, y; \quad y, x \to y, y; \quad y, q \to y, y$, where x, q output "no" and y outputs "yes", which computes whether at least *two* x's exist, is exponentially slower: expected time $O(n)$. The worst-case input is exactly 2 x's and $n - 2$ q's, where the first interaction between the x's takes expected $\binom{n}{2} = \Theta(n^2)$ interactions, i.e., $\Theta(n)$ time.

It is conventional in population protocols to measure the memory usage by counting the total number $s(n)$ of states agents can store in population size n, instead of the number of bits required to represent these states, which is about $\log s(n)$.

In this work, we advance the theoretical understanding of the computational power of population protocols and introduce a dynamic variation of this model. This initiative is to help fill in the gap between molecular computing and the current abstract computation models. In chapter 2, we discuss recent algorithmic advances in population protocols on majority, counting, and other relate problems. In chapter 3 we discuss the well-studied majority problem, in which in a population of n agents, each starting with one of two votes, A or B; the goal of the majority problem is to determine if there are more As, more Bs, or even if it is a tie. More than a decade-long line of research reduced the state and time complexity for majority protocols [**2**,**3**,**20**,**24**,**25**,**30**,**73**,**78**,**79**]. In this chapter, we present the first protocol that is asymptotically optimal in both time and state complexity. Then in chapter 4 and chapter 5, we consider the problem of computing the number of agents in a population protocol. Both exact (computing n) and approximate counting (computing $\lceil \log n \rceil$ or $\lfloor \log n \rfloor$, which gives $2^{\lceil \log n \rceil}$ or $2^{\lfloor \log n \rfloor}$ as a multiplicative factor-2 estimate of n) are considered in this thesis (chapter 4 and chapter 5 respectively). Finally, in chapter 6 we study population protocols under an adversarial setting. We introduce population protocols with dynamic population size that

[6]Technically this connection between probability 1 correctness and reachability requires the number of producible states for any fixed population size n to be finite, which is the case for our protocol.

is in contrast to most work, which assume the population size n is fixed over time. We model an adversary that can add or remove agents arbitrarily and repeatedly during the computation.

1.2. Definitions and Notation

In this thesis, we write n to denote the number of agents in the population. Additionally, we use $\log n$ to denote $\log_2 n$, and $\ln n$ to denote the natural (base-e) logarithm. We write $x \sim y$ to denote that x and y are asymptotically equivalent (implicitly in the population size n), meaning $\lim_{n\to\infty} \frac{x(n)}{y(n)} = 1$.

A *population protocol* is a pair $\mathcal{P} = (\Lambda, \Delta)$, where Λ is a finite set of *states*, and $\Delta \subseteq (\Lambda \times \Lambda) \times (\Lambda \times \Lambda)$ is the *transition function*. (Often this is defined as a function $\delta : \Lambda \times \Lambda \to \Lambda \times \Lambda$, but we allow randomized transitions, where the same pair of inputs can randomly choose among multiple outputs.)

A *transition* (a.k.a., *reaction*) is a 4-tuple $\alpha = (r_1, r_2, p_1, p_2)$, written $\alpha : r_1, r_2 \to p_1, p_2$, such that $((r_1, r_2), (p_1, p_2)) \in \Delta$. If an agent in state r_1 interacts with an agent in state r_2, then they can change states to p_1 and p_2. This notation omits explicit probabilities; our main protocol's transitions can be implemented so as to always have either one or two possible outputs for any input pair, with probability $1/2$ of each output in the latter case.[7] For every pair of states r_1, r_2 without an explicitly listed transition $r_1, r_2 \to p_1, p_2$, there is an implicit *null* transition $r_1, r_2 \to r_1, r_2$ in which the agents interact but do not change state.

Since in each interaction, the scheduler picks an *ordered* pair of agents to interact, we denote these agents in the pseudocode as receiver (rec) and sender (sen) respectively. In other words, unlike many models of distributed computing, population protocols typically are defined to be able to break symmetry "for free".[8]

For our protocols, we specify transitions formally with pseudocode that indicate how agents alter each independent field in their state.

[7]For the purpose of representation, we make an exception in our protocol, when we show agents generate a geometric random variable in one line (see Protocol 37). However, we can assume a geometric random variable is generated through $O(\log n)$ consecutive interactions with each selecting out of two possible outputs (**H** or **T**).

[8]There are examples of interesting protocols using only symmetric transitions, and under certain circumstances, asymmetric protocols can be simulated by symmetric ones [**32**].

A *configuration* $\mathbf{c}$ of a population protocol is a multiset over Λ of size n, giving the states of the n agents in the population. For a state $s \in \Lambda$, we write $\mathbf{c}(s)$ to denote the count of agents in state s.

An *execution* is a sequence of configurations $\mathbf{c}_0, \mathbf{c}_1, \ldots$ such that for all i, applying a transition to $\mathbf{c}_i$ results in $\mathbf{c}_{i+1}$.

More formally, given a configuration $\mathbf{c}$ and transition $\alpha : r_1, r_2 \to p_1, p_2$, we say that α is *applicable* to $\mathbf{c}$ if $\{r_1, r_2\} \subseteq \mathbf{c}$, i.e., $\mathbf{c}$ contains 2 agents, one in state r_1 and one in state r_2. If α is applicable to $\mathbf{c}$, then we write $\mathbf{c} \Rightarrow_\alpha \mathbf{c}'$, where $\mathbf{c}' = \mathbf{c} - \{r_1, r_2\} + \{p_1, p_2\}$ is the configuration that results from applying α to $\mathbf{c}$. We write $\mathbf{c} \Rightarrow \mathbf{c}'$, and we say that $\mathbf{c}'$ is *reachable* from $\mathbf{c}$, if applying 0 or more transitions to $\mathbf{c}$ results in $\mathbf{c}'$. Concretely, if there is a sequence $(\alpha_0, \alpha_1, \ldots, \alpha_k)$ of transitions such that $\mathbf{c} \Rightarrow_{\alpha_0} \mathbf{c}_1 \Rightarrow_{\alpha_1} \ldots \Rightarrow_{\alpha_k} \mathbf{c}'$. This notation omits mention of $\mathcal{P}$; we always deal with a single protocol at a time, so it is clear from context which protocol is defining the transitions.

The *time* until some event is measured as the number of interactions until the event occurs, divided by n, also known as parallel time. Since we would like to model that many interactions can happen in parallel, with $O(1)$ interactions per agent per unit of time, we define n interactions as one unit of time.

All references to "time" in this thesis refer to parallel time. This definition coincides with the time defined in the standard Gillespie kinetic model for chemical reaction networks [**62**], of which population protocols are a special case describing n molecules reacting in a volume proportional to n.

A protocol *stably* solves a problem if the agents eventually reach a correct configuration with probability 1, and no subsequent interactions can move the agents to an incorrect configuration; i.e., the configuration is *stable*. A population protocol is self-stabilizing if from any initial configuration, the agents stably solve the problem.

A protocol *converges* in a given random execution when the output stops changing, though it could take longer to subsequently stabilize.

In a *uniform* protocol (such as protocols in chapter 4, chapter 5, and chapter 6), the transitions are independent from the population size n. In other words, a single protocol computes the output correctly when applied on any population size. In contrast, in *nonuniform* protocols a different Λ

and δ are allowed for different population sizes n (one for each possible value of $\lceil \log n \rceil$), but we abuse terminology and refer to the whole family as a single protocol. In all cases (as with similar nonuniform protocols), the nonuniformity is used to embed the value $\lceil \log n \rceil$ into each agent; the transitions are otherwise "uniformly specified".

To formally define *uniform* computation in population protocols, the agents' transition algorithm is modeled as a 2-tape Turing machine with "state tape" as tape 1 and "computation tape" as tape 2. Our protocol describes a constant number of integer fields comprising each agent's state, which could all be stored in the state tape and separated by a special symbol. When two agents interact they both append the content of the tape 1 of each other into their own tape 1. (We assume the Turing machine state and tape head positions are not transmitted, since they can be assumed WLOG identical in every initial configuration.) The space usage (in bits) s is defined as normal for Turing machines: the maximum number of tape cells that are written during the computation. The number of states is then 2^s, where s is the maximum space usage of any agent during an execution of the protocol. For ease of understanding, we will use standard population protocol terminology and not refer explicitly to details of the Turing machine definition except where needed. Therefore a *state* $s \in \Lambda$ always refers to the TM tape content of an agent (leaving out TM state and tape head positions since these are identical in all initial configurations), where Λ is the set of all states, a *configuration* $\mathbf{c} \in \mathbb{N}^\Lambda$ is a vector indexed by a state, where $\mathbf{c}(s)$ is the *count* of state s in the population. We set the output of our protocol the value stored in a special field labeled "output". This definition is not indisputable since the output of a protocol can also be defined as a function of the fields stored in an agent's memory. Therefore, agents do not necessarily carry the output in their memory. Though the latter definition does not reduce our protocol's space usage it might benefit other algorithms to save memory. We leave the discussion that which definition is more suited to the readers.

The traditional definition of population protocols assumes a deterministic transition function. Several papers [2, 27] indicate how to use the randomness built into the interaction scheduler to provide nearly uniform random bits to the agents, using various *synthetic coin* techniques, showing that the deterministic model can effectively simulate a model with a randomized transition. For brevity and simplicity of presentation, we will simply assume in the model that each agent has

access to a source of uniformly random bits. We assume that each agent has access to independent uniformly random bits, pre-written on a read-only tape, allowing the Turing machine to be deterministic even though it is computing a nondeterministic relation.

When discussing random events in a protocol of population size n, we say event E happens *with high probability* (WHP) if $\Pr[\neg E] = O(n^{-c})$, where c is a constant that depends on our choice of parameters in the protocol, where c can be made arbitrarily large by changing the parameters. In other words, the probability of failure can be made an arbitrarily small polynomial. For concreteness, we will write a particular polynomial probability such as $O(n^{-2})$, but in each case we could tune some parameter (say, increasing the time complexity by a constant factor) to increase the polynomial's exponent. We say event E happens *with very high probability* if $\Pr[\neg E] = O(n^{-\omega(1)})$, i.e., if its probability of failure is smaller than any polynomial probability.

1.3. Useful Techniques and Time Bounds

In this section, we present the most commonly used bounds in this thesis. We will use the following standard multiplicative Chernoff bound:

THEOREM 1.3.1. *Let $X = X_1 + \ldots + X_k$ be the sum of independent $\{0,1\}$-valued random variables, with $\mu = \mathbb{E}[X]$. Then for any $\delta > 0$, $\Pr[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu}{2+\delta}\right)$.*

We will also use the following two concentration bounds on heterogeneous sums of geometric random variables, due to Janson [67, Theorems 2.1, 3.1]:

THEOREM 1.3.2. *Let X be sum of k independent geometric random variables with success probabilities $p_1, ..., p_k$, let $\mu = \mathbb{E}[X] = \sum_{i=1}^{k} \frac{1}{p_i}$, and let $p^* = \min_{1 \leq i \leq k} p_i$. For all $\lambda \geq 1$, $\Pr[X \geq \lambda\mu] \leq e^{-p^*\mu(\lambda - 1 - \ln\lambda)}$. For all $\lambda \leq 1$, $\Pr[X \leq \lambda\mu] \leq e^{-p^*\mu(\lambda-1-\ln\lambda)}$.*

The expression $\lambda - 1 - \ln\lambda$ is hard to work with if we are not fixing exact constant values for λ. The following Corollary gives asymptotic approximation to the error bound:

COROLLARY 1.3.3. *Let X be sum of k independent geometric random variables with success probabilities $p_1, ..., p_k$, let $\mu = \mathbb{E}[X] = \sum_{i=1}^{k} \frac{1}{p_i}$, and let $p^* = \min_{1 \leq i \leq k} p_i$. For any $0 < \epsilon < 1$, we have $(1-\epsilon)\mu \leq X \leq (1+\epsilon)\mu$ with probability $1 - \exp\left(-\Theta(\epsilon^2 p^* \mu)\right)$.*

PROOF. Setting $\lambda = 1 + a$ in theorem 1.3.2, where $a = \epsilon$ in the upper bound and $a = -\epsilon$ in the lower bound, by Taylor series approximation of $\ln(1+a)$, $\lambda - 1 - \ln(\lambda) = a - \ln(1+a) \geq a - (a - \frac{a^2}{2} + \frac{a^3}{3}) = a^2(\frac{1}{2} - \frac{a}{3}) = \Theta(\epsilon^2)$. The stated inequality then follows from theorem 1.3.2. □

1.3.1. Epidemic time bounds. The following lemma is essentially Lemmas 1 and 2 from the paper [12]. However, that paper does not state how the various constants are related, which we require for our proofs. We recapitulate their proof, deriving those relationships explicitly. For all $n \in \mathbb{N}^+$, let $H_n = \sum_{k=1}^{n} \frac{1}{k}$ denote the n'th harmonic number, recalling that $H_n \approx \ln n$.

LEMMA 1.3.4 ([12]). *Let T denote the time to complete an epidemic. Then $\mathbb{E}[T] = \frac{n-1}{n} H_{n-1}$, $\Pr\left[T < \frac{1}{4} \ln n\right] < 2e^{-\sqrt{n}}$, and for any $\alpha_u > 0$, $\Pr[T > \alpha_u \ln n] < 4n^{-\alpha_u/4+1}$.*

PROOF. We begin by showing $\Pr[T > \alpha_u \ln n] < 4n^{-\alpha_u/4+1}$. Suppose we have $\alpha_u n \ln n$ interactions, starting with one infected agent. We want to bound the probability that any agent remains uninfected. The second half of an epidemic (after exactly $n/2$ agents are infected) has equivalent distribution to the first, so we analyze just the second half, bounding the probability it requires more than $(\alpha_u/2)n \ln n$ interactions. When half of the agents are infected, each interaction picks an infected sender with probability at least $1/2$. The number of interactions to complete the epidemic is then stochastically dominated by a binomial random variable $\mathcal{B}((\alpha_u/2)n \ln n, 1/2)$, equal to the number of heads after $(\alpha_u/2)n \ln n$ coin flips if $\Pr[\text{heads}] = 1/2$.

Let $\mu = \mathbb{E}[\mathcal{B}((\alpha_u/2)n \ln n, 1/2)] = (\alpha_u/4)n \ln n$ and $\delta = 2/\sqrt{n}$. By the Chernoff bound [77, Corollary 4.10],

$$\Pr[\mathcal{B}((\alpha_u/2)n \ln n, 1/2) < (1-\delta)\mu] \quad < \quad e^{-\delta^2 \mu} = e^{-(4/n)(\alpha_u/4)n \ln n} = e^{-\alpha_u \ln n} = n^{-\alpha_u}.$$

So with probability at least $1 - n^{-\alpha_u}$, more than $((1-\delta)\alpha_u/4)n \ln n > (\alpha_u/8)n \ln n$ (since $\delta < 1/2$) interactions involve an infected sender.

To complete the proof of the time upper bound, we need to bound the probability that these $(\alpha_u/8)n \ln n$ interactions fail to infect all agents. Conditioned on each interaction having an infected sender, the random variable giving the number of interactions until all agents are infected is equivalent to the number of collections required to collect the last $n/2$ coupons out of n total. Angluin, Aspnes, and Eisenstat [12, Lemma 1] showed that for any β, it takes more than $\beta(n/2)\ln(n/2) < (\beta/2)n \ln n$ collections to collect n coupons with probability at most $n^{-\beta+1}$. Let

$\beta = \alpha_u/4$. Then $\Pr[(\alpha_u/8)n \ln n$ interactions fail to infect every agent$] \leq n^{-\beta+1} < n^{-\alpha_u/4+1}$. By the union bound on the events *"fewer than* $(\alpha_u/8)n \ln n$ *interactions involved an infected sender"* and *"*$(\alpha_u/8)n \ln n$ *interactions fail to infect every agent"*, the second half of the epidemic fails to complete within $(\alpha_u/2)n \ln n$ interactions with probability at most $n^{-\alpha_u} + n^{-\alpha_u/4+1} < 2n^{-\alpha_u/4+1}$.

Again by the union bound on the events *"first half of the epidemic takes more than* $(\alpha_u/2)n \ln n$ *interactions"* and *"second half of the epidemic takes more than* $(\alpha_u/2)n \ln n$ *interactions"*, the whole epidemic takes more than $\alpha_u n \ln n$ interactions with probability at most $4n^{-\alpha_u/4+1}$.

To show $\Pr\left[T < \frac{1}{4} \ln n\right] < 2e^{-\sqrt{n}}$, we note that Lemma 1 of [12] shows that if S_n is the number of times a coupon must be collected to collect all coupons, then $\Pr\left[S_n < \frac{1}{4} n \ln n\right] < 2e^{-\sqrt{n}}$. The proof says $2e^{-\Theta(\sqrt{n})}$, but inspection of the argument reveals that the big-Θ constant can be assumed to be 1. In this case, applying the coupon collector argument to the epidemic, since we are proving a time lower bound, if we assume that *every* interaction involves an infected sender, this process stochastically dominates the real epidemic. Thus $\Pr\left[T < \frac{1}{4} \ln n\right] < 2e^{-\sqrt{n}}$.

To analyze the expected time, starting from one infected agent, we require exactly $n - 1$ interactions between an infected and an uninfected agent. When k agents are infected, the probability that the next interaction is of this form is

$$\frac{k(n-k)}{\binom{n}{2}} = \frac{2k(n-k)}{n(n-1)},$$

so the expected time until such an interaction is

$$\frac{n-1}{2k(n-k)}.$$

By linearity of expectation, the total expected time for all $n-1$ interactions is

$$\begin{aligned}
\sum_{k=1}^{n-1} \frac{n-1}{2k(n-k)} &= \frac{n-1}{2} \sum_{k=1}^{n-1} \frac{1}{k(n-k)} \\
&= \frac{n-1}{2} \sum_{k=1}^{n-1} \left(\frac{1}{nk} + \frac{1}{n(n-k)} \right) \\
&= \frac{n-1}{2n} \left(\sum_{k=1}^{n-1} \frac{1}{k} + \sum_{k=1}^{n-1} \frac{1}{n-k} \right) \\
&= \frac{n-1}{2n} \left(2 \sum_{k=1}^{n-1} \frac{1}{k} \right) \\
&= \frac{n-1}{n} H_{n-1}.
\end{aligned}$$

□

Note that this assumes a receiver can infect a sender and vice versa. A one-way epidemic in which only senders can infect receivers would have exactly twice the expected time: $\frac{2(n-1)}{n} H_{n-1}$.

The following lemmas give applications of theorem 1.3.2 and theorem 1.3.3 to common processes that we will repeatedly analyze. When the fractions we consider are distinct and independent of n, theorem 1.3.3 applies to give tight time bounds with very high probability. We also consider cases where we run a process to completion and bring a count to 0. Here we must use theorem 1.3.2 and only get a high probability bound with times at most a constant factor above the mean.

First we consider the epidemic process, $i, s \to i, i$, moving from a constant fraction infected to another constant fraction infected.

LEMMA 1.3.5. *Let $0 < a < b < 1$. Consider the epidemic process starting from a count of $a \cdot n$ infected agents. The expected parallel time t until there is a count $b \cdot n$ of infected agents is*

$$\mathbb{E}[t] = \frac{\ln\left(b - \frac{1}{n}\right) - \ln\left(1 - b + \frac{1}{n}\right) - \ln(a) + \ln(1-a)}{2} \sim \frac{\ln(b) - \ln(1-b) - \ln(a) + \ln(1-a)}{2}.$$

Let $c = \min(a, 1 - b + \frac{1}{n})$ and $0 < \epsilon < 1$. Then $(1-\epsilon)\mathbb{E}[t] < t < (1+\epsilon)\mathbb{E}[t]$ with probability at least $1 - \exp\left(-\Theta(\epsilon^2 \mathbb{E}[t]\, nc)\right)$.

PROOF. When there are i infected agents, the probability the next reaction infects another agent and increases this count is $\frac{i(n-i)}{\binom{n}{2}}$. The number of interactions T for the count to increase

from $a \cdot n$ to $b \cdot n$ is a sum of geometric random variables, with expected value

$$\begin{aligned}\mathbb{E}[T] &= \sum_{i=a\cdot n}^{b\cdot n-1} \frac{\binom{n}{2}}{i(n-i)} \sim \frac{1}{2}\sum_{i=a\cdot n}^{b\cdot n-1} \frac{1}{\frac{i}{n}(1-\frac{i}{n})} \sim \frac{n}{2}\int_{x=a}^{b-\frac{1}{n}} \frac{dx}{x(1-x)} \\ &= \frac{n}{2}[\ln(x) - \ln(1-x)]_a^{b-\frac{1}{n}} = n \cdot \frac{\ln\left(b-\frac{1}{n}\right) - \ln\left(1-b+\frac{1}{n}\right) - \ln(a) + \ln(1-a)}{2},\end{aligned}$$

giving the stated value of $\mathbb{E}[t]$ after converting from interactions to parallel time. The minimum probability $p^* = \Theta(\min(a, 1-b+\frac{1}{n}))$, so using theorem 1.3.3 we have Then $(1-\epsilon)\mathbb{E}[t] < t < (1+\epsilon)\mathbb{E}[t]$ with probability at least $1 - \exp\left(-\Theta(\epsilon^2 \mathbb{E}[T]\, p^*)\right)$. □

Note that if $a, (1-b), \epsilon$ are all constants independent of n, then the bound above is with very high probability. If we wanted to consider the complete epidemic process (which starts with $a = 1/n$ and ends with $(1-b) = 1/n$), we would have $\mathbb{E}[t] = \Theta(\log n)$ and minimum probability $p^* = \Theta(\frac{1}{n})$. Then the probability bound would become $1 - \exp(-\Theta(\log n)) = 1 - n^{-\Theta(1)}$, which is high probability, but requires carefully considering the constants to get the exact polynomial bound. In our proofs, we only use the very high probability case. For tight bounds on a full epidemic with precise constants, see [34, 82].

1.3.2. Fast averaging protocol. The averaging technique discussed in this section is used in the subsequently discussed counting protocols in chapter 4, section 2.3.2, section 2.3.3, and section 2.4.6.

The averaging technique, also known as randomized load balancing [21, 29, 86], was first introduced in population protocols in [7] to solve the exact majority problem.

Each agent's state is an integer; for intuition, assume the integers represent a "load"[9] that each agent holds. The averaging rules allow each selected pair of agents to exchange loads to balance (as best they can) their values, e.g., $(2, 11) \to (6, 7)$. This leads the population to a configuration in which all agents have almost equal values: concretely, if the total load among the population is m, then after stabilization, each agent holds either the value $\lfloor m/n \rfloor$ or $\lceil m/n \rceil$. Stabilization can take $\Theta(n)$ time in the worst case, but it takes only $O(\log n)$ time for all agents to hold *three* consecutive values (two of which are $\lfloor m/n \rfloor$ or $\lceil m/n \rceil$) [21, 78]. The averaging technique has been crucial in several polylogarithmic-time protocols for problems such as population size

[9] In this thesis, whenever the load values are nonnegative, we use "tokens" instead to present a more intuitive explanation of the protocols.

counting [28, 51] and majority related problems [7, 24, 78, 79], and its time complexity has been tightly analyzed [21, 29, 79, 81].

Notably, Mocquard, Anceaume, Aspnes, Busnel, and Sericola [79] used the averaging technique to solve a generalization of the exact majority problem. Considering a population with n_a, n_b initialized agents in states A and B, i.e., $n_a + n_b = n$, the authors designed an averaging-based protocol that counts the *exact* difference between the number of agents in the A and B (computing the value of $n_a - n_b$).

In this protocol, the A and B agents start with $+m$ and $-m$ values respectively, where m is a large integer with respect to the population size n. Thus the population as a whole starts with a total of $m(n_a) - m(n_b)$ load. The protocol is designed to almost equally distribute the load among the agents while *preserving the total sum*. In this protocol, the agents update their state according to the "discrete averaging" rule described in Protocol 0.

Protocol 0 `DiscreteAveraging`(rec, sen)
Initialization:
if `agent.input` = A: `agent.ave` $\leftarrow m$
if `agent.input` = B: `agent.ave` $\leftarrow -m$

1: rec.`ave`, sen.`ave` $\leftarrow \left\lceil \frac{\text{rec.ave}+\text{sen.ave}}{2} \right\rceil, \left\lfloor \frac{\text{rec.ave}+\text{sen.ave}}{2} \right\rfloor$
2: rec.`output` $\leftarrow \left\lfloor \frac{n \times \text{rec.ave}}{m} + \frac{1}{2} \right\rfloor$

Initializing Protocol 0 with n_a, n_b agents in states A and B, it is shown that the agents' `ave` value converges to $\frac{(n_a - n_b)m}{n}$ quickly. In fact, the authors of [79] proved for $m = \left\lceil \frac{\sqrt{2}n^{3/2}}{\sqrt{\delta}} \right\rceil$, after $O(n \log n)$ interactions with probability $1-\delta$, the `output` field of agents will be equal to $n_a - n_b$. (e.g. to achieve a high probability result, one can set $\delta = 1/n$ and start the protocol with $m = O(n^2)$).

The protocol given in [79] is nonuniform; it is assumed that the population size is known in advance. Crucially, the protocol requires all agents to store the exact value of n in their memory to compute the `output`. In a separate paper [78], Mocquard, Anceaume, and Sericola show how to remove this assumption and make the protocol uniform. Their protocol computes the ratio of A agents with respect to n within a multiplicative factor error $(1 + \epsilon)$ of the true proportion for any precision $\epsilon > 0$ using $2\left\lceil \frac{3}{4\epsilon} \right\rceil + 1$ states.

1.4. Thesis contribution

The majority problem: In chapter 3, we focus on the exact majority protocol of [50]. In the exact majority problem, in a population of n agents, each agent starts with one of two votes, A or B; the goal of the majority problem is to determine if there are more As, more Bs, or even if it is a tie. Computing the majority is a useful primitive for more complicated computations [12, 13]. More than a decade-long line of research reduced the state and time complexity for majority protocols [2, 3, 20, 24, 25, 30, 73, 78, 79].

Our presented protocol solves this problem in optimal $O(\log n)$ time using $O(\log n)$ states. By introducing a new synchronization scheme (fixed-resolution phase clock) that allows the agents to synchronize within "hours" of $O(1)$ time (previous synchronization techniques allow hours of $O(\log n)$ time [7, 12, 60]). Still, we prove this looser synchronization keeps a large constant fraction of agents synchronized every hour.

Exact and approximate size counting: Nonuniform protocols in the literature [2, 3, 5, 20, 24, 25, 26, 27, 30, 50, 61, 79, 91] initialize each agent with an estimate of the population size (e.g., $\lfloor \log n \rfloor$). Hence we study the problem of computing the population size. In our papers [47, 51] (covered in chapter 4 and chapter 5 respectively), we asked *how the agents in a population (with no prior knowledge about the population size) could calculate the population size?*

In chapter 4, we focus on exact counting: computing n; while in chapter 5 we discuss the approximate population size counting problem, i.e., computing an approximation of $\log n$.

The recent algorithmic advances for population size counting problem provide composable building blocks that simplify the (uniform) solution of other problems: compute an estimate of $\log n$, and use this value where a nonuniform protocol would use the hard-coded constant $\lceil \log n \rceil$. We can adopt a counting technique as a black box and compose it with a nonuniform protocol through a restarting scheme [28, 47, 60] to obtain a uniform protocol.

We presented the first sublinear time protocol to solve exact counting. Our protocol relies on the discrete averaging idea of [79], which has been extensively used in recent studies [7, 24, 28, 78, 79].

For approximate size counting, we proposed a protocol that computes $\log n \pm 5.7$, using $O(\log^2 n)$ time and $O(\log^4 n)$ states, both WHP. We extended a standard coin flip technique that generates an estimate $\Theta(\log n)$ [2] to $\log n + O(1)$ by taking an average of $K = \log n$ independent estimates. We

utilized advanced techniques to overcome two main challenges: first, storing all K values takes too much memory, and second, normal Chernoff bounds don't apply to unbounded random variables.

Berenbrink et al. [28] improved upon both our result by presenting almost optimal time-space optimal counting protocols (for both exact and approximate counting problems) while using similar techniques to ours.

Population protocols with dynamic size In the conventional population protocols model, we assume the population size is constant throughout the computation. However, it is crucial to consider dynamic changes that happen to the distributed sensor networks or dynamic domains like biological applications. Hence, in chapter 6 we study population protocols with a population size that is dynamically changing. Perhaps the most realistic scenario is sensor networks in which the agents can arrive and leave a network continuously. In this model, the protocols should converge to their output despite the network changes. This constraint forces protocols to be uniform and requires the population to adapt "quickly" to the new size. My goal is to study the limits of computation in the dynamic model.

This model allows an adversary that changes the population size infinitely many times and adds or removes an arbitrary number of agents. Suppose the adversary changes the population size from n to n'; In that case, we say a protocol P compute function F if protocol P converges to the correct value after $\text{polylog}(n')$ time. Of course, the adversary might change the population size many times; however, the protocol must converge to the correct answer if the population size is stable for $\Omega(\text{polylog}(n'))$ time.

A fundamental question to ask in the dynamic population size model is "counting" the number of agents in the population size. Note that any uniform protocol stably converges to the correct output with a weaker adversary who can only insert agents but not delete existing ones since the transitions in a uniform protocol are independent of the population size and can tolerate inserting new agents. However, if we increase the power of the adversary to insert agents with arbitrary internal states, existing uniform protocols fail to compute the function. In chapter 6, we introduced a protocol that approximates the population size with a polynomial number of states [49].

Closest to our setting, but assuming the agents communicate in *synchronous rounds*, Goldwasser et al. [63] introduced a protocol that "maintains" the population size.

CHAPTER 2

Population Protocols Literature Review

In this section, we review recent related results that have been shown in population protocols. See also [6, 48, 57] for in-depth surveys on recent algorithmic advances.

2.1. Averaging technique

Key to many fast computation in population protocols is a protocol, due to Mocquard, Anceaume, Aspnes, Busnel, and Sericola [79] (see section 4.2.3). Despite the title of that paper ("Counting with Population Protocols"), it actually solves a different problem, a generalization of the majority problem: count the exact difference between "blue" and "red" agents in the initial population. The protocol assumes an initial leader and that each agent initially stores n exactly. In a follow-up work [78], Mocquard et al. showed a *uniform* protocol that, for any $\epsilon > 0$, computes an approximation of the relative proportion (but not exact number) of "blue" nodes in the population, within multiplicative factor $(1 + \epsilon)$ of the true proportion. The approximation precision ϵ depends on a constant number m, which is encoded in the initial state. They also describe a protocol to find the number of "blue" nodes in the population, However, like [79], this latter protocol is not uniform since the transition function encodes the exact value of n.

In a different network model, Jelasity and Montresor [68] use a similar technique to ours that involves a fast "averaging" similar to [78,79]. However, they do arbitrary-precision rational number averaging, so have a larger memory requirement (not analyzed). They also assume each agent initially has a unique ID. Goldwasser, Ostrovsky, Scafuro, Sealfon [63] study a related problem in a synchronous variant of population protocols: assuming that both an adversary and the agents themselves have the ability to create and destroy agents (similar to the more general model of chemical reaction networks), using polylog(n) states, they *maintain* the population size within a multiplicative constant of a target size. This is likely relevant to the exact and approximate size counting problems, since the protocol of [63] must "sense" when the population size is too large or small and react.

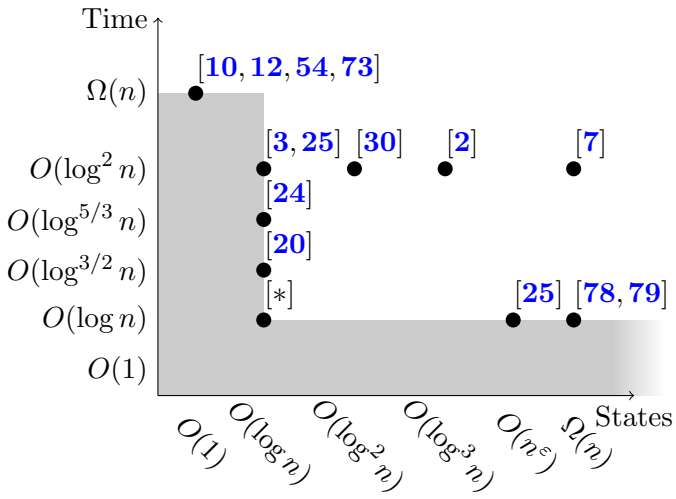


TABLE 2.1. Summary of results on the stable exact majority problem in population protocols, including the protocol in chapter 3 [∗]. Gray regions are provably impossible: $o(\log\log n)$ state, $o(n)$ time unconditionally [2], $o(\log n)$ state, $O(n^{1-\varepsilon})$ time for monotone, output-dominant protocols [3], and $o(\log n)$ time unconditionally.

2.2. Majority computation

Angluin, Aspnes, and Eisenstat [13] showed a protocol they called *approximate majority*, which means that starting from an initial population of n agents with opinions A or B, if $|A - B| = \omega(\sqrt{n}\log n)$ (i.e., the *gap* between the initial majority and minority counts is greater than roughly $\sqrt{n}$), then with high probability the algorithm stabilizes to all agents adopting the majority opinion in $O(\log n)$ time. A tighter analysis by Condon, Hajiaghayi, Kirkpatrick, and Maňuch [40] reduced the required gap to $\Omega(\sqrt{n\log n})$.

Mertzios, Nikoletseas, Raptopoulos, and Spirakis [73], and independently Draief and Vojnović [54], showed a 4-state protocol that solves *exact* majority problem, i.e., it identifies the majority correctly, no matter how small the initial gap.[1] We henceforth refer to this simply as the *majority* problem. The protocol of [54, 73] is also *stable* in the sense that it has probability 1 of getting the correct answer. However, this protocol takes $\Omega(n)$ time in the worst case: when the gap is $O(1)$. Known work on the stable majority problem is summarized in table 2.1. GKasieniec, Hamilton, Martin, Spirakis, and Stachowiak [59] investigated $\Omega(n)$ time protocols for majority and the more general "plurality consensus" problem. Blondin, Esparza, Jaax, and Kučera [31] show a similar stable (also $\Omega(n)$ time) majority protocol that also reports if there is a tie.

[1]The 4-state protocol doesn't identify ties, (gap = 0), but this can be handled with 2 more states; see Stable Backup.

Alistarh, Gelashvili, and Vojnović [7] showed the first stable majority protocol with worst-case polylogarithmic expected time, requiring $\Omega(n)$ states. A series of positive results reduced the state and time complexity for stable majority protocols [2,3,20,24,25,30,78,79]. Ben-Nun, Kopelowitz, Kraus, and Porat showed the current fastest stable sublinear-state protocol [20] using $O(\log^{3/2} n)$ time and $O(\log n)$ states. The current state-of-the-art protocols use alternating phases of *cancelling* (two biased agents with opposite opinions both become unbiased, preserving the difference between the majority and minority counts) and *splitting* (a.k.a. *doubling*): a biased agent converts an unbiased agent to its opinion; if all biased agents that didn't cancel can successfully split in that phase, then the count difference doubles. The goal is to increase the count difference until it is n; i.e., all agents have the majority opinion. See [6, 57] for relevant surveys.

Some non-stable protocols solve exact majority with high probability but have a small positive probability of incorrectness. Berenbrink, Elsässer, Friedetzky, Kaaser, Kling, and Radzik [25] showed a protocol that with initial gap α uses $O(s + \log\log n)$ states and WHP converges in $O(\log n \log_s(\frac{n}{\alpha}))$ time.[2] Setting $\alpha = 1$ and $s = O(1)$, their protocol uses $O(\log\log n)$ states and converges in $O(\log^2 n)$ time. Kosowski and Uznański [69] showed a protocol using $O(1)$ states and converging in poly-logarithmic time with high probability.

On the negative side, Alistarh, Aspnes, and Gelashvili [3] showed that any stable majority protocol taking (roughly) less than linear time requires $\Omega(\log n)$ states if it also satisfies two conditions (satisfied by all known stable majority protocols, including ours): *monotonicity* and *output dominance*. These concepts are discussed in section 3.8. In particular, the $\Omega(\log n)$ state bound of [3] applies only to *stable* (probability 1) protocols; the high probability protocol of [25], for example, uses $O(\log\log n)$ states and $O(\log^2 n)$ time.

Counting in population protocols

Population size counting is the problem of computing the number of agents in a population protocol. Both exact [28, 51] (computing n) and approximate counting [2, 28, 47, 76] (computing $\lceil \log n \rceil$ or $\lfloor \log n \rfloor$, which gives $2^{\lceil \log n \rceil}$ or $2^{\lfloor \log n \rfloor}$ as a multiplicative factor-2 estimate of n) have been considered in the literature. Considering the size counting problem in a nonuniform model of population protocol is trivial since we can provide the agents the values of n or $\lceil \log n \rceil$ as advice. Thus, all cited papers solving this problem use the uniform variant of the model [2, 28, 47, 51, 76].

[2] This protocol is SimpleMajority in [25], which they then build on to achieve multiple stable protocols. The stable protocols require either $\Omega(n)$ stabilization time or $\Omega(\log n)$ states to achieve sublinear stabilization time.

In this section, we will discuss the existing counting protocols and draw attention to their time-space tradeoff. We will cover both the exact and approximate counting problems, since, for most protocols, having an approximation of $\log n$ suffices. table 2.2 and table 2.3 summarize both exact and approximate counting protocols in the conventional model of population protocols (with initialized population).

Exact counting protocols					
ref.	sec.	prob.	time	states	comments
[51]	2.3.2	1	$O(\log n \log\log n)$	$O(n^{60})$	stable
[28]	2.3.3	$1-\frac{O(1)}{n}$	$O(\log n)$	$O(n \log n)$	-
[28]	2.3.3	1	$O(\log n)$	$O(n \log n \log\log n)$	stable

TABLE 2.2. Summary of existing protocols for the exact counting problem. Note that "stable" means correct with probability 1. For all the stable protocols, the stated time bounds the stated time bounds are proven both with high probability and in expectation. However, the state complexity for all the protocols is correct with high probability. We also mention the correctness probability for each protocol under the "prob." column. We also discuss counting in population protocols with constant message size and in the self-stabilizing model in section 2.3.4 and section 2.3.5 respectively.

Approximate counting protocols						
ref.	sec.	output value (range)	prob.	time	states	comments
[28, 50]	2.4.1	$\lfloor \log n \rfloor$	1	$O(n \log n)$	$O(\log^2 n)$	stable
[2]	2.4.2	$[\frac{1}{2}\log n, 9\log n]$	$1-\frac{O(1)}{n^3}$	$O(\log n)$	$O(\log^9 n)$	deterministic
[2, 47, 91]	2.4.2	$[\log n - \log\ln n, 2\log n]$	$1-\frac{O(1)}{n}$	$O(\log n)$	$O(\log^2 n)$	-
[25, 60]	2.4.3	$[\frac{\log n}{16}, 256\log n]$	$1-\frac{O(1)}{n}$	$O(\log n)$	$O(\log\log n)$	deterministic
[47]	2.4.4	$[\log n - 5.7, \log n + 5.7]$	$1-\frac{O(1)}{n}$	$O(\log^2 n)$	$O(\log^4 n)$	-
[28]	2.4.6	$\lfloor \log n \rfloor$ or $\lceil \log n \rceil$	$1-\frac{O(\log n)}{n^2}$	$O(\log^2 n)$	$O(\log n \log\log n)$	-
[28]	2.4.6	$\lfloor \log n \rfloor$ or $\lceil \log n \rceil$	1	$O(\log^2 n)$	$O(\log^2 n \log\log n)$	stable

TABLE 2.3. Summary of existing leaderless protocols for the approximate counting problem. The approximation factor of each protocol is implied under the "output value". The columns follow the same convention as table 2.2. One leader-driven protocol is discussed in section 2.4.5.

2.3. Exact population size counting with a fixed size population

In the exact size counting problem, the agents aim to compute their population size n. In some protocols, to reduce the space complexity, the agents report their estimate of n as a function of their internal fields [28, 51] rather than storing the population size explicitly in their memory. This trick helps the agent to describe numbers that exceeds their memory limit. For example, the agents

might store $\mathtt{a} = \lfloor \log n \rfloor$ but set their output as $2^{\mathtt{a}}$ without explicitly computing the value of $2^{\mathtt{a}}$ to keep their memory usage $\Theta(\log n)$ instead of using linear states.

2.3.1. Naïve slow protocol. A naïve protocol can count the number of agents in a population using a modified version of the slow leader election protocol. All of the agents start in the `active` state holding 1 token in their `count` variable. For consistency with other counting protocols in this section, we name the leader `active`, retaining the standard pairwise leader elimination $(\mathtt{active}, \mathtt{active}) \to (\mathtt{active}, \mathtt{not\ active})$. We also change the leader election protocol so that the final remaining `active` agent accumulates all the n tokens. The rules of this protocol preserve the total sum of the active tokens. When two active agents interact one of them becomes inactive, and both change their `count` value to the sum of their accumulated tokens. Initially, the agents start with n scattered tokens and eventually, there will remain one `active` agent having all the tokens. Protocol 1 describes how the agents update their state at each interaction.

Protocol 1 `IntegerTokenPassing`(rec, sen)
Initialization:
`agent.count` $\leftarrow 1$; `agent.active` $\leftarrow$ True

```
1: if rec.active = True & sen.active = True then
2:     rec.count, sen.count ← rec.count + sen.count
3:     sen.active ← False
4: if rec.active = False & sen.active = False then
5:     rec.count, sen.count ← max(rec.count, sen.count)
```

The transitions of Protocol 1 require $O(n)$ time to converge to the exact population size n.[3] $\Omega(n)$ is a clear lower bound on the number of states needed for any protocol that requires agents to store the value n, since $\lceil \log n \rceil$ bits are required merely to write the number n. Protocol 1 solves this problem using $2n$ states.

2.3.2. Fast exact counting with polynomial states. Doty, Eftekhari, Michail, Spirakis, and Theofilatos [**51**] devised the first sublinear time protocol for the exact population size counting problem. Although their protocol heavily relies on the idea of the averaging protocol of [**79**] (explained in section 1.3.2), they managed to eliminate the "advance knowledge of n" assumption.

[3] This is a standard analysis in population protocols; for instance, see [**10**, Section 6]. One way to see it requires $\Omega(n)$ time is to observe that once exactly two agents have `active` = True, since there are $\binom{n}{2} = \Theta(n^2)$ total pairs of agents, it takes expected $\Theta(n^2)$ interactions, i.e., $\Theta(n)$ expected parallel time, for the two `active` agents to interact and reduce their count to one.

Their protocol achieves uniformity by putting together one phase of leader election and approximate counting before the averaging phase (see chapter 4 for more details).

Their protocol is stabilizing (correct with probability 1) and converges to the correct value of n after $\Theta(\log n \log\log n)$ time both WHP. (probability at least $1 - \frac{O(\log\log n)}{n}$) and in expectation. The error of having multiple leaders always will be detected (since all agents eventually have unique `codename`s) and the agents replace their `output` value resulting an always correct protocol.

2.3.3. Fast exact counting with linear states. Berenbrink, Kaaser, and Radzik [28] improved the space complexity as well as the time complexity of the exact counting protocol of [51]. They start their exact counting with a subprotocol of [60] that elects not a single leader but a "junta" of n^ϵ leaders, for $0 \leq \epsilon < 1$. In the junta election protocol of [60] (see section 2.4.3 for more details), each agent computes a `level` value, and the maximum `level` among the agents is an approximation of $\log\log n$: assuming l^* is the maximum level, $\log\log n - 4 \leq l^* \leq \log\log n + 8$ with probability at least $1 - O(1/n)$ [25, Lemma 4] [60, Theorem 3]. Having a junta of size n^ϵ, opens the possibility of simulating a "phase clock" that allows agents to stay synchronized within phases of length $\Theta(\log n)$ for $\text{poly}(n)$ time [6, 12, 25, 60]. In addition to the junta, the exact counting protocol of [28] requires having one unique leader.

They use the leader election protocol from [25] that uses constant number of phases to elect a leader: in every even phase, each remaining leader generates a sequence of $\Theta(\log n)$ random bits. In the odd phases, they broadcast the maximum bitstring by epidemic and if a leader encounters a larger bitstring than its own, it updates its state to follower. This leader election protocol is a generalization of the $O(\log^2 n)$ time protocol described in [60], which allows remaining leaders to generate and broadcast 1 random bit in each phase and continues for $O(\log n)$ phases of each $O(\log n)$ time.

In contrast to the protocol of [51], where the leader starts with $\text{poly}(n)$ tokens (n^c for $3 \leq c \leq 9$), at every stage of the protocol of [28], the leader starts with no more than n tokens. Once the agents have almost equal tokens because of the averaging phase, the entire population multiplies their `ave` value (tokens) by another factor of $\approx n$. This trick puts an upper bound of n over the range of possible values of `ave` but achieves having a total of $\text{poly}(n)$ tokens among the population. The protocol of [28] uses $O(n \log n)$ states and converges in $O(\log n)$ time both WHP. However, this protocol has a small probability of error; i.e., it is not *stable* (See section 1.2). It is explained in [28]

how to achieve stabilization in $O(\log n)$ time using $O(n \log n \log\log n)$ states with error detection schemes that point agents to switch to the naïve slow (but stable) section 2.3.1 as a backup.

2.3.4. Fast exact counting with constant size messages. Amir, Aspnes, Doty, Eftekhari, and Severson [9] studied the exact counting problem in population protocols with large memories but limited (constant) message size. Considering the exact population size counting problem in this model, the authors of [9] proposed a leader-driven protocol to count the exact population size that converges in $O(\log^2 n)$ time using $O(n \log^2 n)$ states with probability at least $1 - O(1/n)$. They also proposed a leaderless protocol that counts the exact number of agents in a population using $O(\log^2 n)$ time and $O(n \text{ polylog } n)$ states with probability at least $1 - O(1/n)$. They also demonstrated protocols that *approximate* the population size, also using $O(1)$ messages. See section 2.4 for a definition of approximate population size counting.

The following large-message protocol allows agents to compute n: the leader starts with value 1, and agents conduct a rational-number variant of the averaging protocol (e.g., $1, 0 \to \frac{1}{2}, \frac{1}{2}$; $\frac{1}{2}, 0 \to \frac{1}{4}, \frac{1}{4}$; $\frac{1}{4}, \frac{1}{8} \to \frac{3}{16}, \frac{3}{16}$) until all agents hold dyadic values close enough to $\frac{1}{n}$ that they can uniquely identify the size n. The protocol of [9] simulates this in $O(\log n)$ phases (synchronized via a leader-driven phase clock), averaging together only *constantly* many values at a time, narrowing the interval of values stored internally by agents, until it contains a unique integer reciprocal $\frac{1}{n}$.

2.3.5. Self-stabilizing counting with a fixed size population. So far we have discussed the *initialized* setting, where we assume the protocol is permitted to designate a set of *valid* initial configurations. In the case of the counting problem, we identify a special state x_0, where valid initial configurations have all agents in state x_0. In contrast, a population protocol is *self-stabilizing* if, from *any* initial configuration, it reaches to a correct stable configuration. This is an extreme form of fault tolerance, modeling errors that can alter states arbitrarily, at any time during the execution of a protocol, requiring the protocol to be able to recover from any number of such transient errors, by considering the "initial" configuration to be the (arbitrary) configuration just after the *last* such transient error.

It is worth observing why counting, as defined previously, is impossible in this strict setting. Suppose that a population of n agents has stabilized on output n. Then for any $k < n$, in the self-stabilizing setting, any configuration of a sub-population of k of these agents is a valid starting

configuration for population size k. Then this size-k population must eventually change their output from n to k. However, the interactions that achieve this are possible in the size-k sub-population of the original size-n population, contradicting its stability.

To circumvent this impossibility, protocols for the self-stabilizing counting problem have considered adding one exceptional entity, called the *base station*, such that the adversary is not permitted to affect its memory [15, 16, 17, 66]. Furthermore, only the base station is required to know the count after stabilization; thus it is possible for other agents to have fewer than n states. In these protocols, the base station stably computes the exact number n of agents in the population, called *counted* agents. Assuming a known upper bound P on the population size n, Beauquier, Clement, Messika, Rosaz, and Rozoy [17] proposed a protocol that solves the exact counting problem using $4P$ states. This result improved by Izumi, Kinpara, Izumi, and Wada [66] to $2P$ states space per counted agent. In both protocols, the base station assigns unique names to the counted agents. Beauquier, Burman, Clavière, and Sohier [16] proposed a space-optimal protocol that solves the exact counting problem using 1-bit memory for each counted agent in $O(2^n)$ time. Later on, Aspnes, Beauquier, Burman, and Sohier reduced the exponential time complexity to $O(n \log n)$ time which is also proven to be optimal while still using 1-bit memory for the counted agents [15].

2.3.6. Loosely-stabilizing. Sudo, Nakamura, Yamauchi, Ooshita, Kakugawa, and Masuzawa [90] introduce loose-stabilization as a relaxation for the self-stabilizing model in which starting from any configuration, the population will reach a correct configuration (with respect to the problem) within a short time. After that, the agents remain in the correct configuration a long time but not forever (in contrast with self-stabilization). On the positive side, for the leader election problem, the agents no longer need to know the exact population size to solve the loosely-stabilizing leader election, but a rough upper bound suffices. Loosely-stabilizing leader election has been studied, providing a time-optimal protocol that solves the leader election problem [89] and a tradeoff between the holding and convergence times [65, 92].

2.4. Approximate population size counting with a fixed size population

The study of the counting problem is partially motivated by the existence of nonuniform protocols. Most of these nonuniform protocols require not n exactly, but an approximation, e.g., the value $\lceil \log n \rceil$. In the approximate size counting problem, the agents compute an approximation

of n , e.g., $2^{\lceil \log n \rceil}$, the smallest power of two greater than n, rather than the exact value n. This freedom opens room for protocols with exponentially smaller space complexity.

2.4.1. Naïve slow protocol. Recall that Protocol 1 solves the exact counting problem via pairwise elimination of active agents, passing all the tokens (where each agent starts with one token) to the remaining active agent. A simple modification to Protocol 1 can solve the approximate counting problem using $O(n \log n)$ time [**25**, **50**]. In the protocol presented next, token counts are restricted to powers of two, thus using only $\Theta(\log n)$ states. All agents start in the `active` state with one token stored in their `exponent` field (initially set to 0 representing integer 2^0). When two `active` agents with the same `exponent` value equal to i (integer value of 2^i) interact, one of them becomes `not active`, and both update their `exponent` to $i+1$ (integer value of 2^{i+1}). Additionally, all `not active` agents help propagating the maximum value of `exponent` they have seen (described in Protocol 2).

Protocol 2 `PowersOfTwoTokenPassing`(rec, sen)
Initialization:
`agent.exponent` $\leftarrow 0$; `agent.active` $\leftarrow$ True

```
1: if (rec.active & sen.active) & (rec.exponent = sen.exponent) then
2:     rec.exponent, sen.exponent ← rec.exponent + 1
3:     sen.active ← False
4: if rec.active = False & sen.active = False then
5:     rec.exponent, sen.exponent ← max(rec.exponent, sen.exponent)
```

Although Protocol 2 is slow and takes $O(n \log n)$ time, it utilizes almost optimal space complexity. This protocol uses $O(\log n)$ states having $n - O(\log n)$ agents store $\lfloor \log n \rfloor$; however, requiring all agents to store $\lfloor \log n \rfloor$ results in a $O(\log^2 n)$ state protocol [4]. Note that $\lceil \log \log n \rceil$ bits (equivalently $\Theta(\log n)$ states) are needed to write the number $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$ for any protocol that reports an estimation of $\log n$ as its output.

In the following, we overview the fast protocols that considered the approximate counting problem. Commonly, the output of these protocols is an approximation of $\log n$.

[4]For $n = 2^k, k \in \mathbb{N}$, the population converges to having one unique `active` agent, and all `not active` agents will store the floor of $\log n$. For other values of $n \neq 2^k, k \in \mathbb{N}$, the population converges to $O(\log n)$ `active` agents each having a different value of $\{0, \ldots, \lfloor \log n \rfloor\}$ that results in all null interactions. Note that the interaction between (`active`, 2^3), (`active`, 2^5), concludes with both agents having the same states.
To be concrete, exactly b_1 `active` agents will remain, such that b_1 is the number of 1s in the binary expansion of n. Each of the b_1 `active` agents hold one of the values $i_1, \ldots, i_{b_1}$ for all the indices that have 1 in the binary expansion of n. Thus, to enforce "all" agents (both `active` and `notactive`) report the value of $\log n$, the protocol needs at most $O(\log^2 n)$ states per agent.

2.4.2. Maximum of n geometric random variables. Assuming a randomized protocol, i.e., agents have access to independent, unbiased random bits, there is a simple method for obtaining a constant-factor approximation of $\log n$, i.e, a polynomial factor approximation of n. Recall that a $\frac{1}{2}$-*geometric* random variable is the number of flips of a fair coin until the first heads. It is known that the maximum of n independent $\frac{1}{2}$-geometric random variables is in the interval $[\log n - \log \ln n, 2 \log n]$ with probability at least $1 - O(1/n)$ [47, 56]. Each agent flips a fair coin on each interaction, incrementing a counter until the first heads,[5] and then moves to a "propagate the maximum" stage where the maximum counter value obtained by any agent is spread by epidemic throughout the population, i.e., $i, j \to i, i$ if $i > j$.

2.4.2.1. *Synthetic coins.* Since it may be desirable to use a deterministic transition function, some work has been done on techniques for simulating randomized transitions with a deterministic transition function. Alistarh, Aspnes, Eisenstat, Gelashvili, Rivest [2] proposed a general technique, known as *synthetic coins*, that synthesizes "almost" independent and unbiased random coin flips in a deterministic protocol, "extracting" randomness from the random scheduler. Each agent uses this synthetic coin technique to simulate generating a $\frac{1}{2}$-geometric random variable G_i. Their protocol provides an approximation of $\log n$ in the interval $[1/2 \log n, 9 \log n]$ with probability at least $1 - O(1)/n^3$, i.e., worse bounds than obtained with independent, unbiased coin flips, but still within a constant factor of $\log n$.

Recently, Sudo, Ooshita, Izumi, Kakugawa, and Masuzawa [91] proposed an improved implementation of synthetic coins: independent and unbiased coins (as with [2], using only symmetric transitions). The method of Sudo et al. [91] works as follows for any protocol where "population splitting" can be used. (See [6, Section 4.3].) Create a subpopulation of "coin" agents whose only job is to provide random bits to the remaining "main" agents. Main agents build up a list of random bit values to use in the main algorithm, which they obtain when interacting with coin agents. Coin agents start (after first being assigned to the coin subpopulation) in state J, with the following transitions: $J, J \to K, K$; $K, K \to J, J$; $J, K \to C_0, C_1$. When a main agent interacts with C_b, it appends bit b to its list of random bits. Since the above transitions ensure that there are exactly the same number of C_0 and C_1 agents at any time, the bits are unbiased. Since the scheduler ensures that, conditioned on an interaction being between a main and a coin agent, the

[5] In more powerful variants of the model, each agent runs a randomized Turing machine [37, 47, 51]. In this case the $\frac{1}{2}$-geometric random variable can be generated in one step.

choice of coin agent is independent of other main-coin interactions, the bit values built up in main agents are independent.[6]

2.4.3. Arbitrary biased coins. One can approximate $\log\log n$ with access to random bits with arbitrary bias. We explained above how to approximate $\log n$ with a series of $\frac{1}{2}$-biased coins. Consider instead a special coin whose initial bias (probability of tail, i.e., continuing to flip) of $\frac{1}{2}$ is squared after each coin flip. In other words, the bias is $\frac{1}{2}$ for the first flip, $\frac{1}{4}$ for the second flip, $\frac{1}{16}$ for the third flip, etc. Similarly to the previous protocol, let each agent independently flip this special coin until a head appears and store the number of consecutive tails. In this process, the fraction of agents who get a tail and continue flipping is approximately squared after each flip, so the maximum stored value among n agents is an approximation of $\log\log n$. The junta election protocol of [60] (also explained in [25]) simulates this process without using any coin flips. We describe the modified version of protocol [60] for simplicity [25]. In this protocol the agents store their current coin number using a `level` variable. Initially, all agents start in state ($\texttt{level} = 0, \texttt{active} = \mathsf{True}$), and eventually, all will set their `active` to False. Agents increase their `level` value via the *asymmetric* transition rules indicated in Protocol 3.

Protocol 3 `JuntaElection`(rec, sen)
Initialization:
`agent.level` $\leftarrow 0$; `agent.active` $\leftarrow$ True

```
1: if rec.active & rec.level = 0 then
2:     rec.level ← 1                          ▷ happens at the first interaction
3: if sen.active & sen.level = 0 then
4:     sen.active ← False                     ▷ happens at the first interaction
5: else if rec.active & rec.level > 0 then
6:     if rec.level < sen.level then
7:         rec.level ← rec.level + 1
8:     else if rec.level ≥ sen.level then
9:         rec.active ← False
```

The combination of the first two if statements in Protocol 3 acts similarly to the $\frac{1}{2}$-bias coin. About $n/2$ agents participate in their first interaction as receiver and increase their `level` by 1. Intuitively, in Protocol 3, with αn agents (such that $0 < \alpha < 1$) having $\texttt{level} \geq i \geq 1$, there will be about $\alpha^2 n$ with $\texttt{level} \geq i + 1$.

[6]The only difference with a truly randomized protocol is that main agents may have to wait to build up random bits before being allowed to do a randomized transition with another agent. This does introduce some dependence in the main protocol, which means this is not a fully black-box technique for replacing a randomized protocol with a deterministic protocol.

It is proven that the maximum `level` value in the population (l^*) is an additive approximation of $\log\log n$. More precisely, $\log\log n - 4 \leq l^* \leq \log\log n + 8$ with probability at least $1 - O(1/n)$ [25,60]. This yields a multiplicative factor approximation of $\log n$; see table 2.3.

The above protocol is also a so-called *junta election* protocol: the number of agents who obtain the maximum `level` is $O(\sqrt{n \log n})$ with high probability. These agents can be used, for example to create a "junta-driven phase clock" [60], useful for synchronization.

2.4.4. Fast protocol with constant additive error. Doty and Eftekhari [47] presented a protocol that improves the approximation factor of the protocol of [2], which approximates $\log n$ using maximum of n geometric random variables, from a multiplicative to an additive factor approximation. Their protocol converges to an estimation of $\log n$ in the interval $[\log n - 5.7, \log n + 5.7]$ after $O(\log^2 n)$ time using $O(\log^4 n)$ number of states per agent. They extend one round of taking the maximum of n geometric random variables of [2] to K rounds of taking maximums and computing their average as an approximation of $\log n$. See chapter 5 for an in-depth discussion.

The protocols we have discussed so far for approximating the population size are leaderless. They are not dependent on the existence of a unique leader. In a leaderless protocol, all agents are initially equivalent, and there is no distinguished leader.

2.4.5. Leader-driven, epidemic-based protocol. A leader-driven protocol for approximating the population size was introduced in [76]. The basic idea of their protocol relies on the completion time of an epidemic process. Specifically, the leader triggers an epidemic process (infecting exactly one agent $L, Q \to L^*, A$) and keeps track of the number of infected (c_a) and uninfected (c_q) agents without infecting more agents. The followers (initialized in state Q) participates in this protocol via the one-way epidemic rule ($A, Q \to A, A$). As soon as the number of infected and uninfected agents becomes equal, the leader terminates the protocol and reports the approximation $n' = 2^{c_a+1}$. In [76] it was shown that $c_a \leq 2\log n$ with high probability.[7] This protocol approximates the population size using $\Theta(\log n)$ states for leader while the followers use constant states.

[7] Theorem 1 of [76] states that $\log n \leq c_a$ with high probability. However, this does not appear to hold in simulation. It seems likely that a bound of $c \cdot \log n$ can be proven for some $c > 0$ based on known results lower bounding times for epidemics to spread [82].

2.4.6. Discrete averaging with powers of two. Berenbrink, Kaaser, and Radzik [28] introduced a new averaging protocol via modifying the rules of Protocol 0. Recall that Protocol 0 works by pairwise averaging of nonnegative integer values held by each agent; the modified rules restrict the agents to use numbers that are a perfect power of two. The authors carefully developed the protocol such that the new rules of the protocol still preserve the total sum among the agents. In this variation of the averaging protocol, shown in Protocol 4, the agents can store either a perfect power of two or zero. The constraint helps to reduce the space complexity via representing an integer 2^x with x. To show the exact value of 0, the agents use -1. Using a similar approach to [28, 51] for the exact counting problem, a leader starts an averaging process with a large (with respect to n) positive value, and all the followers start with zero ($\texttt{ave} = -1$).

Protocol 4 `PowersOfTwoAveraging`(rec, sen)
Initialization:
if `agent.leaderBit` = True: `agent.ave` $\leftarrow m$
if `agent.leaderBit` = False: `agent.ave` $\leftarrow -1$

1: **if** rec.ave $= -1$ & sen.ave > 0 **then**
2: rec.ave, sen.ave $\leftarrow$ sen.ave $- 1$
3: **else if** sen.ave $= -1$ & rec.ave > 0 **then**
4: rec.ave, sen.ave $\leftarrow$ rec.ave $- 1$

Utilizing the restricted version of the discrete average process, they proposed an approximate counting protocol that outputs the value $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$ with high probability, using $O(\log^2 n)$ time and $O(\log n \log \log n)$ states. To stably solve the approximate counting problem, they used multiple always correct error detection schemes that point the population to the slow token-passing Protocol 2 if an error occurs. With an overhead of $O(\log n)$ states, their protocol stabilizes to $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$ using $O(\log^2 n \log \log n)$ states after $O(\log^2 n)$ time.

2.5. Dynamic population protocols

2.5.1. Size regulation in a dynamically sized population. Goldwasser, Ostrovsky, Scafuro, and Sealfon [63] consider the *size regulation problem*: approximately maintaining a target size (hard-coded into each agent) using $O(\log \log n)$ bits of memory per agent, despite an adversary that (like ours) adds or removes agents. That paper assumes a model variation in which:

- The agents can replicate or self-destruct.

- The computation happens through synchronized rounds of interactions. At each round the scheduler selects a random matching of size $k = O(n)$ agents to interact.
- The adversary's changes to the population size are limited. The adversary can insert or delete a total of $o(n^{1/4})$ agents within each round.

However, they use a *synchronous* variation of population protocols: in one round of the computation, a constant fraction of agents interact (at most once) via a random matching of size $k = O(n)$. Observe that, unlike the asynchronous scheduler, the synchronous scheduler prevents any agent having multiple interactions per k total interactions. Despite this difference in definitions, it is conceivable that techniques used in the analysis of [63] could be applied to the standard population protocol model. It is also noteworthy that their model of agents being created and destroyed is expressible in the model of chemical reaction networks [87], of which population protocols are a special case.

2.5.2. Approximate population size counting in a dynamic size population. Some of the previous counting protocols would still solve the counting problem in the presence of an adversary who can only add agents (see theorem 6.2.3). However, these protocols fail in the presence of an adversary who can also remove agents, since they work only in the initialized setting and rely on reaching a stable configuration (see theorem 6.2.4). Doty and Eftekhari [49] presented a population size counting protocol robust to an adversary who can add or remove agents. See chapter 6 for more details.

2.5.3. Computation with dynamically changing inputs. Alistarh, Töpfer, and Uznański [8] consider the dynamic variant of the comparison problem. In the comparision problem, a subset of population are in the input states X and Y and the goal is to compute if $X > Y$ or $X < Y$. In the dynamic variant of the comparision problem, they assume an adversary who can change the counts of the input states at any time. The agents should compute the output as long as the counts remain untouched for sufficiently long time. They propose a protocol that solves the comparision problem in $O(\log n)$ time using $O(\log n)$ states per agent, assuming $|X| \geq C_2 \cdot |Y| \geq C_1 \log n$ for some constants $C_1, C_2 > 1$.

Berenbrink, Biermeier, Hahn, and Kaaser [22] consider the adaptive majority problem (generalization of the comparison problem [8]). At any time every agent has an opinion from $\{X, Y\}$

or undecided and their opinions might change adversarially. The goal is to have agreement in the population about the majority opinion. They introduce a non-uniform loosely-stabilizing leaderless phase clock that that uses $O(\log n)$ states to solve the adaptive majority problem. This is similar to having an adversary who can add or remove agents with different opinion. However, all agents are assumed already to have an estimate of $\log n$ that remains untouched. Thus it is not straightforward to use their protocol to solve our problem of obtaining this estimate.

2.6. Population protocols with faulty agents

Most population protocols[8] assume that agents have perfect memory throughout the computation.[9] A single bit of corruption can invalidate the entire computation. In population protocols with $O(1)$ states, Angluin, Aspnes, and Eisenstat [13] introduced an approximate majority protocol robust to $O(\sqrt{n})$ faulty agents. Recently, a population protocol that tolerates linear number of Byzantine agents has been introduced [35]; however, no research has considered other problems such as leader election and counting in this model. Therefore, I plan to consider $\omega(1)$-states population protocols in the presence of an adversary who can corrupt a bounded number of agents at any time by either stopping them or changing their memory arbitrarily. Note that this model is different from the self-stabilizing systems that assume that any number of processes can have corrupted states, requiring that the system eventually returns to a correct configuration. My work on population protocols with faulty agents will support continuous changes by an adversary to capture changes in real-world physical systems. I plan to continue my research on size counting in the presence of faulty agents. In addition, I want to work on leader election and consider other problems.

In a slightly different model, the power of $O(1)$-state protocols with Byzantine agents has been studied [64]. In this model, the agents have unique IDs immune to adversarial changes (Byzantine agents cannot alter their IDs). However, with $\omega(1)$ (but still $o(n)$) state population protocols, one cannot assign IDs to the agents since it requires at least n states. In a related study [44], the authors consider $O(1)$ crash failures in the original model of population protocols with constant state

[8] With the exception of [13, 35, 44].

[9] Self-stabilizing protocols assume that errors eventually stop, so are not robust to *continual* errors from malicious agents.

memory (no IDs). However, it is still open if population protocols tolerate $\omega(1)$ failures. A series of relevant studies consider population protocols with a faulty model of communication [45, 46].

CHAPTER 3

A Time and Space Optimal Stable Protocol Solving Exact Majority

3.1. Introduction

The well-studied *majority* problem is that of determining in an initial population of n agents, each with one of two opinions A or B, whether there are more A, more B, or a tie.

Computing the majority is a useful primitive for more complicated computations [12, 13]. More than a decade-long line of research reduced the state and time complexity for majority protocols [2, 3, 20, 24, 25, 30, 73, 78, 79].

A *stable* protocol solves this problem with probability 1 by eventually entering a configuration in which all agents agree on a correct consensus decision of A, B, or T, from which the consensus cannot change. We describe a protocol that solves this problem using $O(\log n)$ states ($\log\log n + O(1)$ bits of memory) and optimal expected time $O(\log n)$. The number of states $O(\log n)$ is known to be optimal for the class of polylogarithmic time stable protocols that are "output dominant" and "monotone" [3]. These are two natural constraints satisfied by our protocol, making it simultaneously time- and state-optimal for that class. We introduce a key technique called a "fixed resolution clock" to achieve partial synchronization.

3.1.1. Our contribution. We show a stable population protocol solving the exact majority problem in optimal $O(\log n)$ time (in expectation and with high probability) that uses $O(\log n)$ states. Our protocol is both monotone and output dominant (see section 3.8 or [3] for discussion of these definitions), so by the $\Omega(\log n)$ state lower bound of [3], our protocol is both time and space optimal for the class of monotone, output-dominant stable protocols.

A high-level overview of the algorithm is given in sections 3.3.1 and 3.3.2, with a full formal description given in section 3.5. Like most known majority protocols using more than constant space (the only exceptions being in [25]), our protocol is *nonuniform*: agents have an estimate of the value $\lceil \log n \rceil$ embedded in the transition function and state space. section 3.7 describes how to

modify our main protocol to make it uniform, retaining the $O(\log n)$ time bound, but increasing the state complexity to $O(\log n \log\log n)$ in expectation and with high probability. That section discusses challenges in creating a uniform $O(\log n)$ state protocol.

3.2. Preliminaries

There are many modes of computation considered in population protocols: computing integer-valued functions [18,38,52] where the number of agents in a particular state is the output, Boolean-valued predicates [11, 12] where each agent outputs a Boolean value as a function of its state and the goal is for all agents eventually to have the correct output, problems such as leader election [2, 5, 26, 27, 53, 60, 61], and generalizations of predicate computation, where each agent individually outputs a value from a larger range, such as reporting the population size [28, 47, 51]. Majority computation is Boolean-valued if computing the predicate "$A \geq B$?", where A and B represent the initial counts of two opinions A and B. We define the slightly generalized problem that requires recognizing when there is a tie, so the range of outputs is $\{\mathsf{A}, \mathsf{B}, \mathsf{T}\}$.

Formally, if the set of states is Λ, the protocol defines a disjoint partition of $\Lambda = \Lambda_{\mathsf{A}} \cup \Lambda_{\mathsf{B}} \cup \Lambda_{\mathsf{T}}$. For $u \in \{\mathsf{A}, \mathsf{B}, \mathsf{T}\}$, if $a \in \Lambda_u$ for all $a \in \mathbf{c}$, we define *output* $\phi(\mathbf{c}) = u$ (i.e., all agents in $\mathbf{c}$ agree on the output u). Otherwise $\phi(\mathbf{c})$ is undefined (i.e., agents disagree on the output). We say $\mathbf{o}$ is *stable* if $\phi(\mathbf{o})$ is defined and, for all $\mathbf{o}_2$ such that $\mathbf{o} \Rightarrow \mathbf{o}_2$, $\phi(\mathbf{o}) = \phi(\mathbf{o}_2)$, i.e., the output cannot change.

The protocol identifies two special *input states* $A, B \in \Lambda$. A *valid* initial configuration $\mathbf{i}$ satisfies $a \in \{A, B\}$ for all $a \in \mathbf{i}$. We say the *majority opinion* of $\mathbf{i}$ is $M(\mathbf{i}) = \mathsf{A}$ if $\mathbf{i}(A) > \mathbf{i}(B)$, $M(\mathbf{i}) = \mathsf{B}$ if $\mathbf{i}(A) < \mathbf{i}(B)$, and $M(\mathbf{i}) = \mathsf{T}$ if $\mathbf{i}(A) = \mathbf{i}(B)$. The protocol *stably computes majority* if, for any valid initial configuration $\mathbf{i}$, for all $\mathbf{c}$ such that $\mathbf{i} \Rightarrow \mathbf{c}$, there is a stable $\mathbf{o}$ such that $\mathbf{c} \Rightarrow \mathbf{o}$ and $\phi(\mathbf{o}) = M(\mathbf{i})$. Let $O_{\mathbf{i}} = \{\mathbf{o} : \phi(\mathbf{o}) = M(\mathbf{i})\}$ be the set of all correct, stable configurations. In other words, for any reachable configuration, it is possible to reach a correct, stable configuration, or equivalently reach a strongly connected component in $O_{\mathbf{i}}$.

3.2.1. Time complexity. In any configuration the next interaction is chosen by selecting a pair of agents uniformly at random and applying an applicable transition, with appropriate probabilities for any randomized transitions. Thus the sequence of transitions and configurations they reach are random variables. To measure time we count the total number of interactions (including null transitions such as $a, b \to a, b$ in which the agents interact but do not change state),

and divide by the number of agents n. In the population protocols literature, this is often called "parallel time": n interactions among a population of n agents equals one unit of time.

If the protocol stably computes majority, then for any valid initial configuration $\mathbf{i}$, the probability of reaching a stable, correct configuration, $\Pr[\mathbf{i} \Rightarrow O_{\mathbf{i}}] = 1$.[1] We define the *stabilization time* S to be the random variable giving the time to reach a configuration $\mathbf{o} \in O_{\mathbf{i}}$.

3.3. Nonuniform majority algorithm description

The goal of section 3.3 through section 3.6.7 is to show the following theorem:

THEOREM 3.3.1. *There is a nonuniform population protocol Nonuniform Majority, using $O(\log n)$ states, that stably computes majority in $O(\log n)$ stabilization time, both in expectation and with high probability.*

3.3.1. High-level overview of algorithm. In this overview we use "pseudo-transitions" such as $A, B \to \mathcal{O}, \mathcal{O}$ to describe agents updating a portion of their states, while ignoring other parts of the state space.

Each agent initially has a `bias`: $+1$ for opinion A and -1 for opinion B, so the population-wide sum $g = \sum_v v.\texttt{bias}$ gives the *initial gap* between opinions. The majority problem is equivalent to determining $\text{sign}(g)$. Transitions redistribute biases among agents but, to ensure correctness, maintain the population-wide g as an invariant. Biases change through *cancel reactions* $+\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$ and *split reactions* $\pm\frac{1}{2^i}, 0 \to \pm\frac{1}{2^{i+1}}, \pm\frac{1}{2^{i+1}}$, down to a minimum $\pm\frac{1}{2^L}$. The constant $L = \lceil \log_2(n) \rceil$ ensures $\Theta(\log n)$ possible states. The *gap* is defined to be $\sum_v \text{sign}(v.\texttt{bias})$, the difference in counts between majority and minority biases. Note the gap should grow over time to spread the correct majority opinion to the whole population, while the invariant g should ensure correctness of the final opinion.

The cancel and split reactions average the bias value between both agents, but only when the average is also a power of 2, or 0. If we had averaging reactions between all pairs of biases (also allowing, e.g., $\frac{1}{2}, \frac{1}{4} \to \frac{3}{8}, \frac{3}{8}$), then all biases would converge to $\frac{g}{n}$, but this would use too many states.[2] With our limited set $\{0, \pm\frac{1}{2}, \pm\frac{1}{4}, \ldots, \pm\frac{1}{2^L}\}$ of possible biases, allowing all cancel and split reactions simultaneously does not work. Most biases appear simultaneously across the population,

[1] Since population protocols have a finite reachable configuration space, this is equivalent to the stable computation definition that for all $\mathbf{c}$ reachable from $\mathbf{i}$, there is a $\mathbf{o}' \in O_{\mathbf{i}}$ reachable from $\mathbf{c}$.

[2] This was effectively the approach used for majority in [7, 78], for an $O(n)$ state, $O(\log n)$ time protocol.

reducing the count of each bias, which slows the rate of cancel reactions. Then the count of unbiased 0 agents is reduced, which slows the rate of split reactions, see fig. 3.1a. Also, there is a non-negligible probability for the initial minority opinion to reach a much greater count, if those agents happen to do more split reactions, see fig. 3.1b. Thus using only the count of positive versus negative biases will not work to solve majority even with high probability.

To solve this problem, we partially synchronize the unbiased agents with a field `hour`, adding $\log n$ states $0_0, 0_1, 0_2, \ldots, 0_L$. The new split reactions

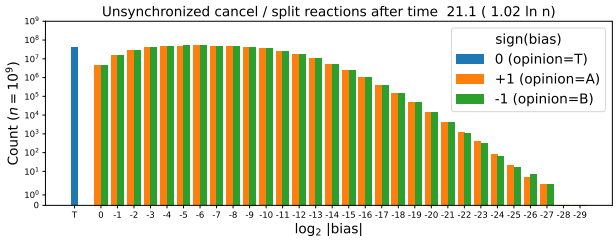
$$\pm\frac{1}{2^i}, 0_h \to \pm\frac{1}{2^{i+1}}, \pm\frac{1}{2^{i+1}} \quad \text{if } h > i$$

will wait until $\texttt{hour} \geq h$ before doing splits down to $\texttt{bias} = \pm\frac{1}{2^h}$. We could use existing phase clocks to perfectly synchronize `hour`, by making each hour use $\Theta(\log n)$ time, enough time for all opinionated agents to split. Then WHP all agents would be in states $\{0_h, +\frac{1}{2^h}, -\frac{1}{2^h}\}$ by the end of hour h, see fig. 3.1c. The invariant $g = \sum_v v.\texttt{bias}$ implies that all minority opinions would be eliminated by hour $\lceil \log_2 \frac{1}{g} \rceil \leq L$. This would give an $O(\log n)$-state, $O(\log^2 n)$-time majority algorithm, essentially equivalent to [**3**, **25**].

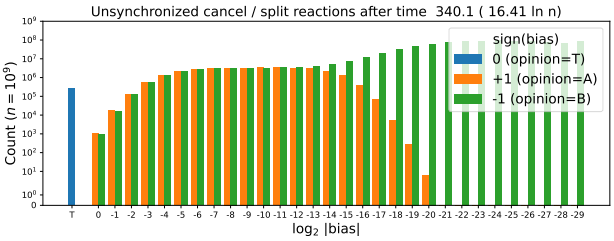
The main idea of our algorithm is to use these rules with a faster clock using only $O(1)$ time per hour. The `hour` field of unbiased agents is synchronized to a separate subpopulation of clock agents, who use a field `minute`, with k consecutive minutes per hour. Minutes advance by *drip reactions* $C_i, C_i \to C_i, C_{i+1}$, and catch up by *epidemic reactions* $C_i, C_j \to C_{\max(i,j)}, C_{\max(i,j)}$. See fig. 3.2 for an illustration of the clock `minute` and `hour` dynamics.

Since $O(1)$ time per hour is not sufficient to bring all agents up to the current hour before advancing to the next, we now have only a large constant fraction of agents, rather than all agents, synchronized in the current hour. Still, we prove this looser synchronization keeps the values of `hour` and `bias` relatively concentrated, so by the end of this phase, we reach a configuration as shown in fig. 3.1d. Most agents have the majority opinion (WLOG positive), with three consecutive biases $+\frac{1}{2^l}, +\frac{1}{2^{l+1}}, +\frac{1}{2^{l+2}}$.

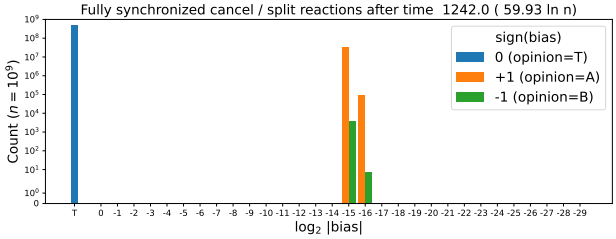
Detecting ties. This algorithm gives an elegant way to detect a tie with high probability. In this case, $g = 0$, and with high probability, all agents will finish the phase with $\texttt{bias} \in \left\{0, \pm\frac{1}{2^L}\right\}$. Checking this condition *stably* detects a tie (i.e., with probability 1, if this condition is true, then there is a tie), because for any nonzero value of g, there must be some agent with $|\texttt{bias}| > \frac{1}{2^L}$.



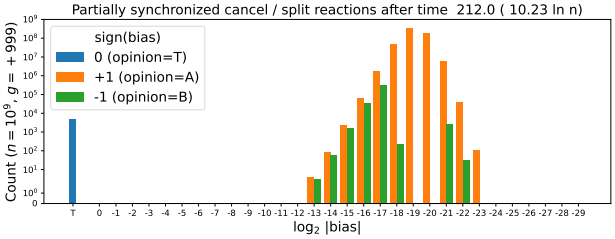
(a) Cancel/split reactions with no synchronization. All states become present, many in about equal counts. Rate of cancel reactions and fraction of 0 agents are $\Theta(\frac{1}{\log n})$.



(b) Later snapshot of the simulation in fig. 3.1a. The initial minority B now has a much larger count, because those agents happened to undergo more split reactions.



(c) Cancel/split reactions, fully synchronized into $O(\log n)$ time hours, at the beginning of hour 16. All minority are eliminated by hour $\log n$ in $O(\log^2 n)$ time.



(d) Main phase of our protocol, split reactions partially synchronized using the clock in fig. 3.2, at the end of this $O(\log n)$ time phase. Most agents are left with `bias` $\in \{+\frac{1}{2^{18}}, +\frac{1}{2^{19}}, +\frac{1}{2^{20}}\}$. Later phases eliminate the remaining minority agents.

FIGURE 3.1. Cancel / split reactions with no synchronization (fig. 3.1a,fig. 3.1b), perfect synchronization (fig. 3.1c), and partial synchronization (fig. 3.1d) via the fixed-resolution phase clock of our main protocol. Plots generated from [41].

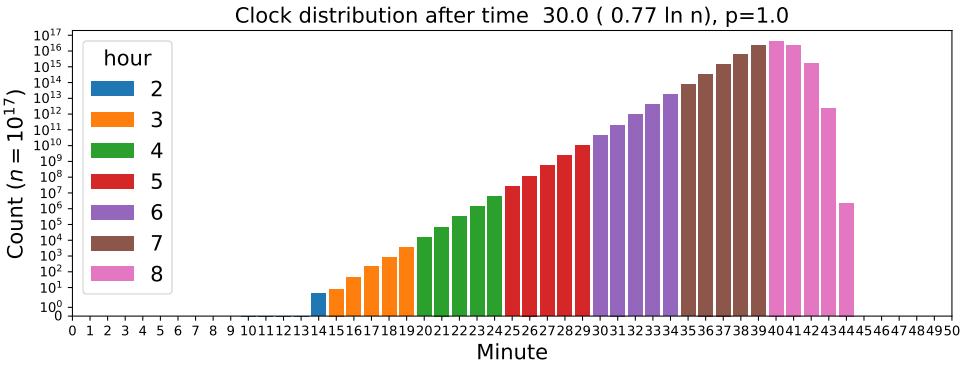


FIGURE 3.2. Clock rules of our protocol, showing a travelling wave distribution over minutes, on a larger population of size $n = 10^{17}$ to emphasize the distribution. The distribution's back tail decays exponentially, and its front tail decays doubly exponentially. A large constant fraction of agents are in the same two consecutive `hour`'s (here 7 and 8). Plot generated from [41].

Cleanup Phases. We must next eliminate all minority opinions, while still relying on the invariant $g = \sum_v v.\texttt{bias}$ to ensure correctness. Note that is it possible with low probability to have a greater count of minority opinions (with smaller values of `bias`), so only relying on counts of positive and negative biases would give possibilities of error.

We first remove any minority agents with large bias, by using an additional subpopulation of Reserve agents that enable additional split reactions for large values of $|\texttt{bias}| > \frac{1}{2^l}$. Then after cancel reactions with the bulk of majority agents, the only minority agents left must have $|\texttt{bias}| < \frac{1}{2^{l+2}}$.

To then remove minority agents with small bias, we allow agents with larger bias to "consume" agents with smaller bias, such as an interaction between agents $+\frac{1}{4}$ and $-\frac{1}{256}$. Here the positive agent can be thought to hold the entire bias $+\frac{1}{4} - \frac{1}{256} = +\frac{63}{256}$, but since this value is not in the allowable states, it can only store that its bias is in the range $+\frac{1}{8} \leq \texttt{bias} \leq +\frac{1}{4}$. Without knowing its exact `bias`, this agent cannot participate in future averaging interactions. However, we show there are enough available majority agents to eliminate all remaining minority via these *consumption reactions*. Thus with high probability, all minority agents are eliminated.

A final phase checks for the presence of both positive and negative `bias`, and if one has been completely eliminated, it stabilizes to the correct output. In the case where both are present, this is a detectable error, where we can move to a slow, correct backup that uses the original inputs. Due to the low probability of this case, it contributes negligibly to the total expected time.

3.3.2. Intuitive description of each phase. Our full protocol is broken up into 11 consecutive phases. We describe each phase intuitively before presenting full pseudocode in section 3.5. Note that some further separation of phases was done to create more straightforward proofs of correctness, so simplicity of the proofs was optimized over simplicity of the full protocol pseudocode. It is likely possible to have simpler logic that still solves majority via the same strategy.

Phase 0: "Population splitting" [6] divides agents into roles used in subsequent phases: Main, Reserve, Clock. In timed phases (those not marked as *Untimed* or *Fixed-resolution clock*, including the current phase), Clock agents count from $\Theta(\log n)$ to 0 to cause the switch to the next phase after $\Theta(\log n)$ time.

"Standard" population splitting uses reactions such as $x, x \to r_1, r_2$ to divide agents into two roles r_1, r_2. This takes $\Theta(n)$ time to converge, which can be decreased to $\Theta(\log n)$

time via $r_1, x \to r_1, r_2$ and $r_2, x \to r_2, r_1$, while maintaining that $\#r_1$ and $\#r_2$ are both $n/2 \pm \sqrt{n}$ WHP. However, since all agents initially have an opinion, but Clock and Reserve agents do not hold an opinion, agents that adopt role Clock or Reserve must first pass off their opinion to a Main agent.

From each interacting pair of unassigned agents, one will take the Main role and hold the opinions of both agents, interpreting A as $+1$ and B as -1. This Main agent will then be allowed to take at most one other opinion (in an additional reaction that enables rapid convergence of the population splitting), and holding 3 opinions can end up with a bias in the range $\{-3, -2, -1, 0, +1, +2, +3\}$.

Phase 1: Agents do "integer averaging" [5] of biases in the set $\{-3, \ldots, +3\}$ via reactions $i, j \to \lfloor \frac{i+j}{2} \rfloor, \lceil \frac{i+j}{2} \rceil$. Although taking $\Theta(n)$ time to converge in some cases, this process is known [83] to result in three consecutive values in $O(\log n)$ time. If those three values are detected to be $\{-1, 0, +1\}$ in the next phase, the algorithm continues.

Phase 2: (*Untimed*) Agents propagate the set of opinions (signs of biases) remaining after Phase 1 to detect if only one opinion remains. If so, we have converged on a majority consensus, and the algorithm halts here (see fig. 3.6). At this point, this is essentially the exact majority protocol of [73], which takes $O(\log n)$ time with an initial gap $\Omega(n)$, but longer for sublinear gaps (e.g., $\Omega(n)$ time for a gap of 1). Thus, if agents proceed beyond this phase (i.e., if both opinions A and B remain at this point), we will use later that the gap was smaller than $0.025 \cdot \#\mathsf{Main}$. With low probability both opinions remain but some agent has $|\texttt{bias}| > 1$, in which case we proceed directly to a slow stable backup protocol in Phase 10.

Phase 3: (*Fixed-resolution clock*) The key goal at this phase is to use cancel and split reactions to average the bias across the population to give almost all agents the majority opinion. Biased agents hold a field $\texttt{exponent} \in \{-L, \ldots, -1, 0\}$, describing the magnitude $|\texttt{bias}| = 2^{\texttt{exponent}}$, a quantity we call the agent's *mass*. Cancel reactions eliminate opposite biases $+\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$ with the same exponent; cancel reactions strictly reduce total mass. Split reactions $\pm\frac{1}{2^i}, 0 \to \pm\frac{1}{2^{i+1}}, \pm\frac{1}{2^{i+1}}$ give half of the bias to an unbiased agent, decrementing the exponent; split reactions preserve the total mass. The unbiased $\mathcal{O}$ agents, with $\texttt{role} = \mathsf{Main}$, $\texttt{opinion} = \texttt{bias} = 0$, act as the fuel for split reactions.

We want to obtain tighter synchronization in the exponents than fig. 3.1a, approximating the ideal synchronized behavior of the $O(\log^2 n)$ time algorithm of fig. 3.1c while using only $O(\log n)$ time. To achieve this, the Clock agents run a "fixed resolution" clock that keeps them roughly synchronized (though not perfectly; see fig. 3.2) as they count their "minutes" from 0 up to $L' = kL$, using $O(1)$ time per minute. This is done via "drip" reactions $C_i, C_i \to C_i, C_{i+1}$ (when minute i gets sufficiently populated, pairs of C_i agents meet with sufficient likelihood to increment the minute) and $C_j, C_i \to C_j, C_j$ for $i < j$ (new higher minute propagates by epidemic).[3] If randomized transitions are allowed, by lowering the probability p of the drip reaction, the clock rate can be slowed by a constant factor. Although we prove a few lemmas about this generalized clock, and some of our simulation plots in section 3.4 use $p < 1$, our proofs work even for $p = 1$, i.e., a deterministic transition function, although this requires constant-factor more states (by increasing the number of "minutes per hour", explained next).

Now the $\mathcal{O}$ agents will use $\Theta(\log n)$ states to store an "hour", coupled to the C clock agents via $C_{\lfloor i/k \rfloor}, \mathcal{O}_j \to C_{\lfloor i/k \rfloor}, \mathcal{O}_{\lfloor i/k \rfloor}$ if $\lfloor i/k \rfloor > j$, i.e., every consecutive k Clock minutes corresponds to one Main hour, and clock agents drag $\mathcal{O}$ agents up to the current hour (see fig. 3.3). Our proofs require $k = 45$ minutes per hour when $p = 1$, but smaller values of k work in simulation. For example, the simulation in fig. 3.1d showing intended behavior of this phase used only $k = 3$ minutes per hour with $p = 1$.

This clock synchronizes the `exponent`s because agents with `exponent` $= -i$ can only split down to `exponent` $= -(i+1)$ with an $\mathcal{O}$ agent that has `hour` $\geq i + 1$. This prevents the biased agents from doing too many splits too quickly. As a result, during hour i, most of the biased agents have $|\texttt{bias}| = \frac{1}{2^i}$, so the cancel reactions $+\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$ happen at

[3]This clock is similar to the power-of-two-choices leaderless phase clock of [3], where the agent with smaller (or equal) minute increments their clock ($C_j, C_i \to C_j, C_{i+1}$ for $i \leq j$), but increasing the smaller minute by only 1. Similarly to our clock, the maximum minute can increase only with both agents at the same minute. A similar process was analyzed in [23], and in fact was shown to have the key properties needed for our clock to work—an exponentially-decaying back tail and a double-exponentially-decaying front tail—so it seems likely that a power-of-two-choices clock could also work with our protocol.
The randomized variant of our clock with drip probability p is also similar to the "junta-driven" phase clock of [60], but with a linear number $2pn$ of agents in the junta, using $O(1)$ time per minute, rather than the $O(n^\epsilon)$-size junta of [60], which uses $O(\log n)$ time per minute. There, smaller minutes are brought up by epidemic, and only an agent in the junta seeing another agent at the same minute will increment. The epidemic reaction is exactly the same in both rules. The probability p of a drip reaction can be interpreted as the probability that one of the agents is in the junta. For similar rate of $O(1)$ time per minute phase clock construction see also Dudek and Kosowski work [55].

a high rate, providing many $\mathcal{O}$ agents as "fuel" for future split reactions. We tune the constants of the clock to ensure hour i lasts long enough to bring most biased agents down to $\texttt{exponent} = -i$ via split reactions and then let a good fraction do cancel reactions (see fig. 3.4c).

The key property at the conclusion of this phase is that unless there is a tie, WHP most majority agents end up in three consecutive exponents $-l, -(l+1), -(l+2)$, with a negligible mass of any other Main agent (majority agents at lower/higher exponents, minority agents at any exponent, or $\mathcal{O}$ agents).[4] Phases 10-12 use this fact to quickly push the rest of the population to a configuration where *all* minority agents have exponents strictly below $-(l+2)$; Phase 8 then eliminates these minority agents quickly.

Phase 4: (*Untimed*) The special case of a tie is detected by the fact that, since the total bias remains the initial gap g, if all biased agents have minimal exponent $-L$, g has magnitude less than 1:

$$|g| = \left| \sum_{a.\texttt{role}=\mathsf{Main}} a.\texttt{bias} \right| \leq \sum_{a.\texttt{role}=\mathsf{Main}} |a.\texttt{bias}|$$
$$\leq \sum_{a.\texttt{role}=\mathsf{Main}} \frac{1}{2^L} < \frac{n}{2^{\lceil \log_2(n) \rceil}} \leq 1.$$

The initial gap g is integer valued, so $|g| < 1 \implies g = 0$. Thus this condition implies there is a tie with probability 1; the converse that a tie forces all biased agents to exponent $-L$ holds with high probability. If only exponent $-L$ is detected, the algorithm halts here with all agents reporting output T (see fig. 3.8). Otherwise, the algorithm proceeds to the next phase.

Phase 5, Phase 6: Using the key property of Phase 3, these phases WHP pull all biased agents above exponent $-l$ down to exponent $-l$ or below using the Reserve R agents. The R's activate themselves in Phase 5 by sampling the exponent of the first biased agent they meet. This ensures WHP that sufficiently many Reserve agents exist with exponents $-l, -(l+1), -(l+2)$ (distributed similarly to the agents with those exponents). Then in Phase 6, they act as fuel for splits, via $R_i, \pm\frac{1}{2^j} \to \pm\frac{1}{2^{j+1}}, \pm\frac{1}{2^{j+1}}$ when $|i| > |j|$. The reserve agents, unlike the $\mathcal{O}$ agents in Phase 3, do not change their exponent in response

[4] l is defined such that if all biased agents were at exponent $-l$, the difference in counts between majority and minority agents would be between $0.4 \cdot \#\mathsf{Main}$ and $0.8 \cdot \#\mathsf{Main}$.

to interactions with Clock agents. Thus sufficiently many reserve agents will remain to allow the small number of biased agents above exponent $-l$ to split down to exponent $-l$ or below.[5]

Phase 7: This phase allows more general reactions to distribute the dyadic biases, allowing reactions between agents up to two exponents apart, to eliminate the opinion with smaller exponent: $\frac{1}{2^i}, -\frac{1}{2^{i+1}} \to \frac{1}{2^{i+1}}, 0$ and $\frac{1}{2^i}, -\frac{1}{2^{i+2}} \to \frac{1}{2^{i+1}}, \frac{1}{2^{i+2}}$ (and the equivalent with positive/negative biases swapped). Since all agents have exponent $-l$ or below, and many more majority agents exist at exponents $-l, -(l+1), -(l+2)$ than the total number of minority agents anywhere, these (together with standard cancel reactions $\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$) rapidly eliminate all minority agents at exponents $-l, -(l+1), -(l+2)$, while maintaining $\Omega(n)$ majority agents at exponents $\geq -(l+2)$ and $< 0.01n$ total minority agents, now all at exponents $\leq -(l+3)$.

Phase 8: This phase eliminates the last minority agents, while ensuring that if any error occurred in previous phases, some majority agents remain, to allow detecting the error by the presence of both opinions.[6]

The biased agents add a Boolean field `full`, initially False, and *consumption* reactions that allow an agent at a larger exponent i to consume (set to mass 0 by setting it to be $\mathcal{O}$) an agent at an arbitrarily smaller exponent $j < i$. Now the remaining agent represents some non-power-of-two mass $m = 2^i - 2^j$, which it lacks sufficient memory to track exactly. Thus setting the flag `full` = True corresponds to the agent having an uncertain mass m in the range $2^{i-1} \leq m < 2^i$. Because of this uncertainty, full agents are not allowed to consume other smaller levels. However, there are more than enough high-exponent majority agents by this phase to consume all remaining lower exponent minority agents.

[5]The reason we do this in two separate phases is to ensure that the Reserve agents have close to the same distribution of exponents that the Main agents have at the end of Phase 3. If Reserve agents allowed split reactions in the same phase that they sample the `exponent` of Main agents, then the splits would disrupt the distribution of the Main agents before all Reserve agents have finished sampling. This would possibly give the Reserve agents a significantly different distribution among levels than the Main agents had at the start. While this may possibly work anyway, we find it is more straightforward to prove if the Reserve agents have a close distribution over `exponent` values to that of the Main agents.

[6]A naïve idea to reach a consensus at this phase is to allow cancel reactions $\frac{1}{2^i}, -\frac{1}{2^j} \to 0, 0$ between arbitrary pairs of exponents with opposite opinions. However, this has a positive probability of erroneously eliminating the majority. This is because the majority, while it necessarily has larger mass than the minority at this point, could have smaller *count*. For example, we could have 16 A's with `exponent` $= -2$ and 32 B's with `exponent` $= -5$, so A's have mass $16 \cdot 2^{-2} = 4$ and B's have smaller mass $32 \cdot 2^{-5} = 1$, but larger count than A.

Crucially, agents that have consumed another agent and set `full` = True may themselves then be consumed by a third agent (with `full` = False) at an even larger exponent. This is needed because a minority agent at exponent $i \leq -(l+3)$ may consume a (rare) majority agent at exponent $j < i$, but the minority agent itself can be consumed by another majority agent with exponent $k > i$.

Phase 9: (*Untimed*) This is identical to Phase 2: it detects whether both biased opinions A and B remain. If not (the likely case), the algorithm halts, otherwise we proceed to the next phase.

Phase 10: (*Untimed*) Agents execute a simple, slow stable majority protocol [31], similar to that of [54, 73] but also handling ties. This takes $\Theta(n \log n)$ time, but the probability that an earlier error forces us to this phase is $O(1/n^2)$, so it contributes negligibly to the total expected time.

3.4. Simulation results

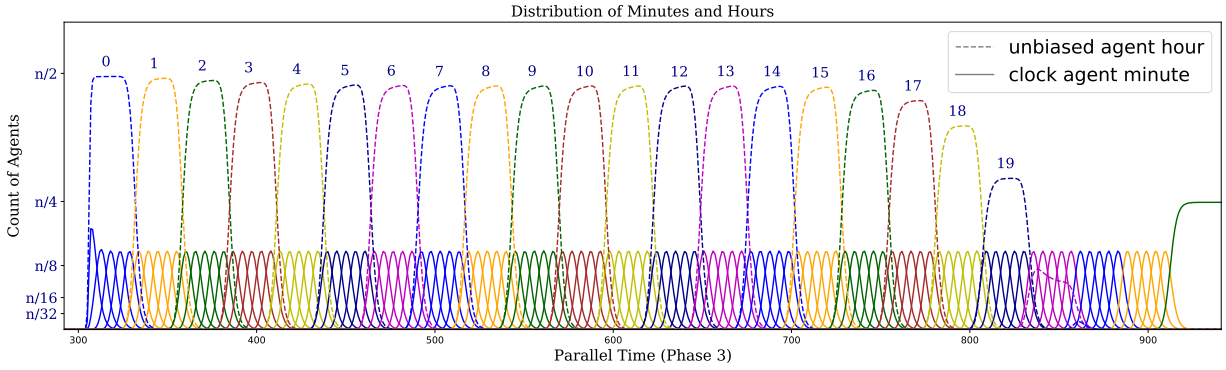
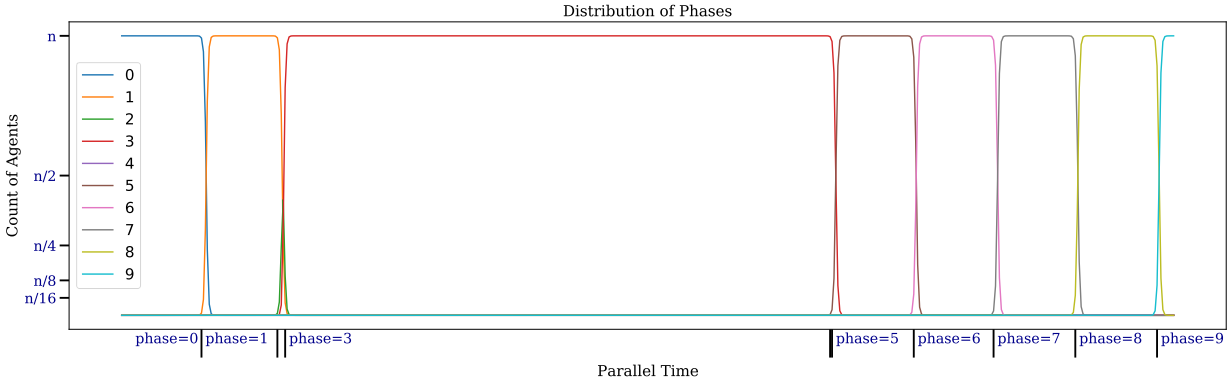
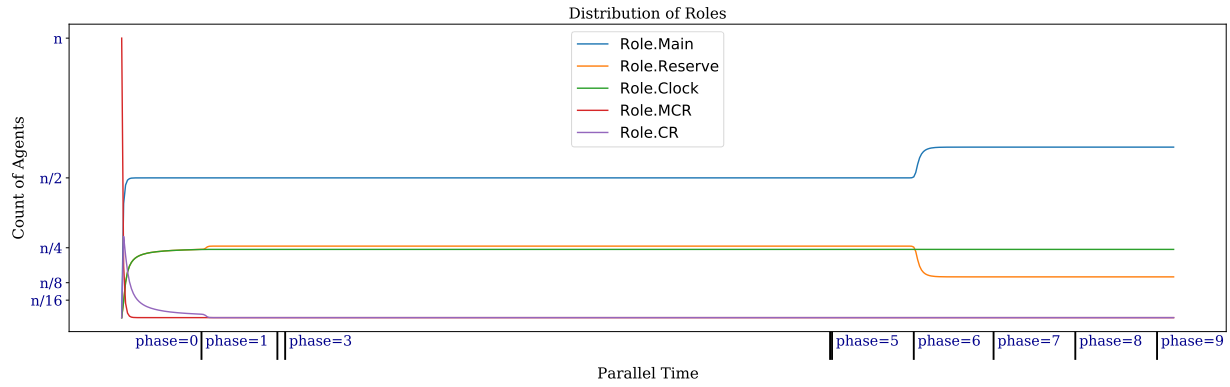


FIGURE 3.3. Fixed-resolution phase clock used in Phase 3, with $n \approx 2^{23}$. All Clock agents set `minute` = 0 and count up to $k \cdot L$ with special rules defined in Phase 3. The solid curves show the distributions of the counts at each value of `minute` in the Clock agents. Blocks of $k = 5$ consecutive minutes correspond to a `hour` in the unbiased $\mathcal{O}$ agents with `role` = Main and `bias` = 0. The dashed curves show the distribution of the counts of each value of `hour`, with the value written on top. We show every k minutes with one color equal to its hour color. Near the end of Phase 3, the $\mathcal{O}$ count falls to 0 as the majority count takes over. This plot used the value $k = 5$ to emphasize the clock behavior. Later plots will all use the weaker constant $k = 2$. In later plots we also omit the `minute` distribution and only show the `hour` from the $\mathcal{O}$ agents.

In this section we show simulation results, where the complete pseudocode of section 3.5 was translated into Java code available on GitHub [42]. In these simulations, we stop the protocol once all agents reach Phase 9. For the low probability case that agents switch to the Phase 10,



(a) The `phase` distribution, giving counts of the agents with each value of `phase`. We show these markers on the horizontal axis in later plots. We have set max `counter` $= 5\log_2(n)$, and removed the counting requirements for Clock agents in Phase 0. This makes all timed phases 5, 6, 10, 11, 12, 13 take around the same parallel time $2.5\log_2(n)$. The fixed-resolution clock in Phase 3 uses $O(\log n)$ time with a larger constant. Phase 2 and Phase 4 are untimed, so they end almost immediately and are not labeled above.

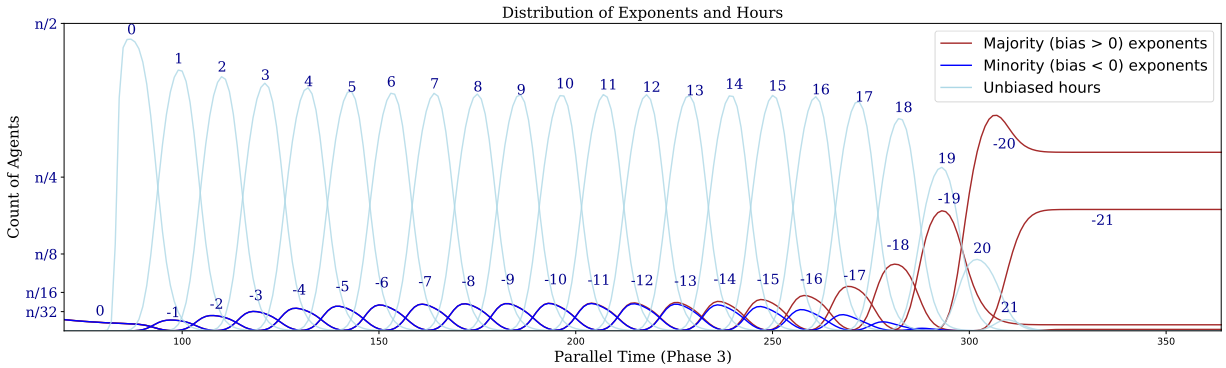


(b) The `role` distribution. All agents start in role $\mathsf{Role}_{\mathsf{MCR}}$. By the end of Phase 0 almost all agents decide on a role. They enter Phase 1 with $\approx \frac{n}{2}$, $\approx \frac{n}{4}$, and $\approx \frac{n}{4}$ agents in the respective roles Main, Clock, and Reserve. An agent's role remain the same in the following phases except Phase 6, where split reactions convert Reserve agents into Main agents.

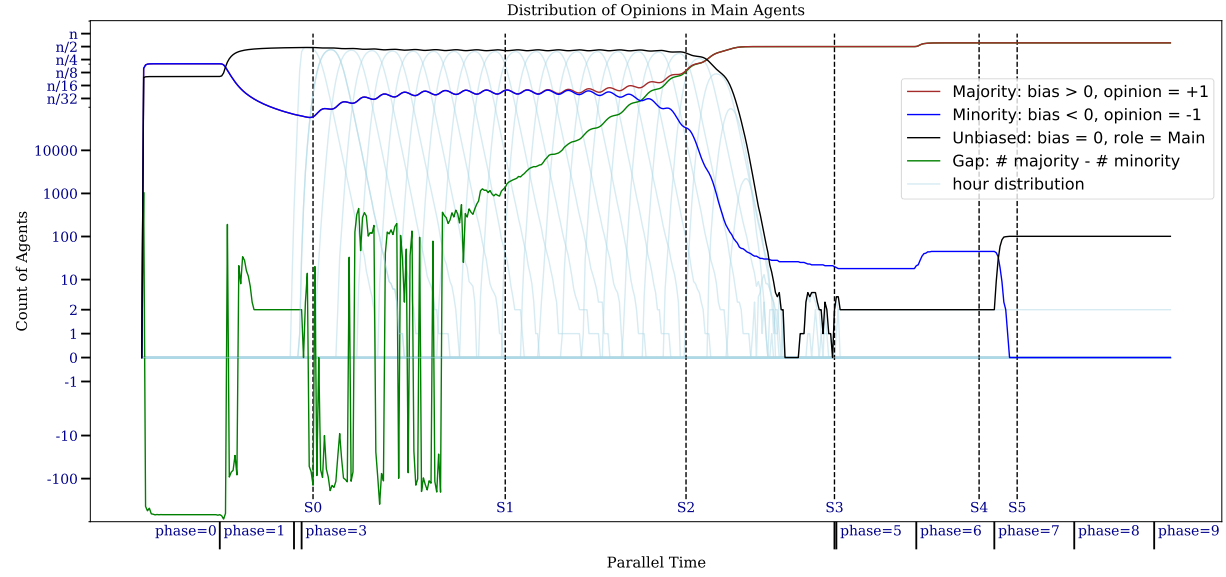
FIGURE 3.4. The `phase` and `role` distributions.

the simulator prints an error indicating the agents should switch to slow back up, but as expected this was not observed in our simulations. For all our plots, we collect data from simulations with $n \approx 2^{23}$, p (drip probability) $= 0.1$. The first simulation in fig. 3.3 shows the relationship between `minute` of Clock agents and `hour` of Main agents. Here we used $k = 5$ minutes per hour to show clearly the relationship and the discrete nature of the hours.

All remaining simulations used the even weaker value $k = 2$, to help see enough low probability behavior that the logic enforcing probability-1 correctness in later phases is necessary. We show 3 simulations corresponding to the 3 different types of initial gap. figs. 3.4 and 3.5 show constant

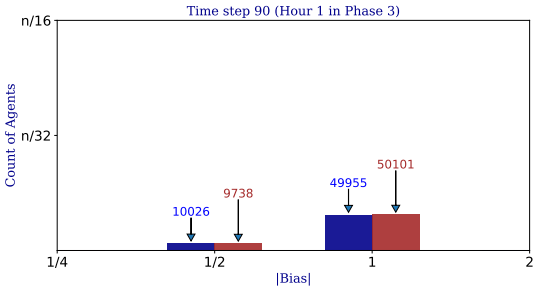


(c) `exponent` and `hour` distribution in biased and unbiased Main agents during Phase 3. The values for `exponent` can decrease from 0 to $-L$. As described in Phase 3, during hour h, only agents with `exponent` $> -h$ are allowed to split and decrease their `exponent` by one. Thus, the changes of `exponent` are synchronized with the the changes in the `hour` values. In this simulation, Phase 3 stops with majority of agents having 3 consecutive values $(-19, -20, -21)$ in their `exponent`. The `hour` values are shown behind the next plot in fig. 3.4d.

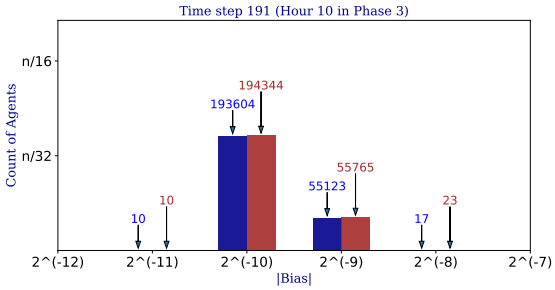


(d) The distribution of `opinion` over all Main agents, with count shown on a log scale. The green line shows the difference in count between majority and minority agents. Special snapshots at times marked S0, S1, S2, S3, S4, S5 are shown in fig. 3.5. All Main agents are assigned in Phase 0, with `bias` $\in \{0, \pm1, \pm2, \pm3\}$, and all initial biases represented held by this Main subpopulation. Here the gap depends randomly on the distribution of $|\texttt{bias}| \in \{1, 2, 3\}$. Then Phase 1 brings all `bias` $\in \{-1, 0, +1\}$, so the gap returns to exactly the initial gap of 2, and the biased counts decrease polynomially like $\frac{1}{t}$ from cancel reactions. In the first part of Phase 3 the gap oscillates randomly about 0, (S0 in fig. 3.5a). Once we reach a high enough hour / low enough exponent, the doubling trend takes over and the gap undergoes constant exponential growth (S1 in fig. 3.5b). Finally, this becomes visible as a separation between counts of majority and minority agents (S2 in fig. 3.5c). Phase 3 ends with a small but nonzero count of minority agents and the count of unbiased $\mathcal{O}$ agents brought near 0 (S3 in fig. 3.5d). Then during Phase 6, this minority count is amplified slightly by more split reactions bringing the minority exponents down (S4 in fig. 3.5e). During Phase 7, additional cancel reactions bring the minority count to 0 (S5 in fig. 3.5f). Since minority agents are gone, Phase 8 has no effect, and the protocol stabilizes to the correct majority output in Phase 9.

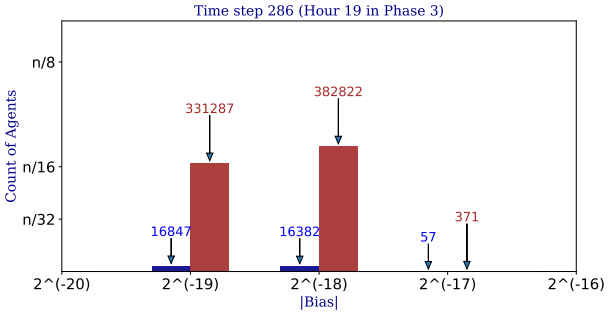
FIGURE 3.4. Simulation with $n = 5122666 \approx 2^{23}$, $p = 0.1$ (drip probability), $k = 2$ (number of minutes per hour), and initial gap $g = 2$. The horizontal axis is in units of parallel time, with the ranges corresponding to each phase marked. All agents converge to the correct majority output in Phase 9.



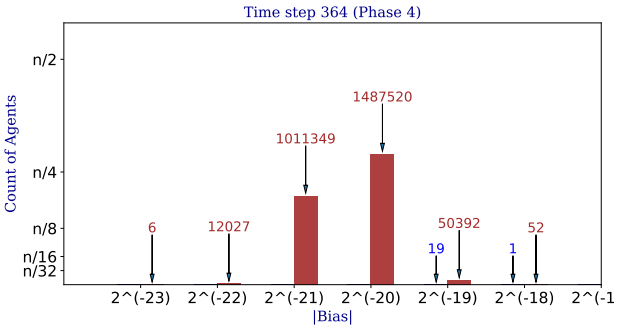
(a) Snapshot S0. At the start of Phase 3. The count of minority agents (blue) currently exceeds the count of majority agents because the minorities have done more split reactions. Summing the signed biases, however, gives $+50101 - 49955 + \frac{9738}{2} - \frac{10026}{2} = +146 - 144 = +2$, the invariant initial gap.



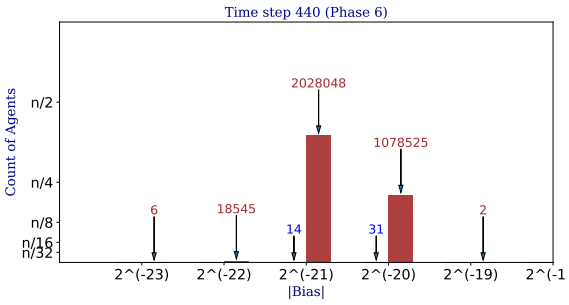
(b) Snapshot S1. A typical distribution at `hour` = 10, when split reactions have brought most agents to `exponent` $= -10$. Only a few $\mathcal{O}$ agents leaked ahead to `hour` = 11 and enabled splits down to `exponent` $= -11$, and no agents have leaked further ahead than this.



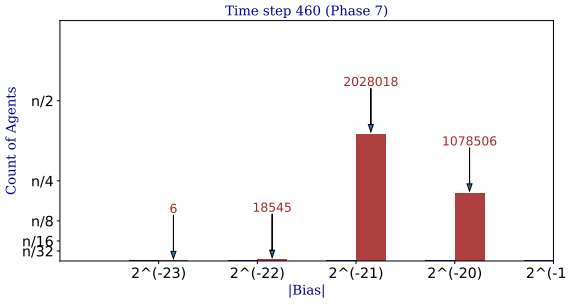
(c) Snapshot S2. We have reached the special exponent $-l = -19$. The count of minority (blue) agents has vastly decreased over the last few hours, and now there will be few more cancel reactions to produce more $\mathcal{O}$ agents.



(d) Snapshot S3. We end Phase 3 with most Main agents with the majority opinion, and `bias` $\in \{-19, -20, -21\}$ in a range of 3 consecutive values, as shown in theorem 3.6.8. Only a few minority agents are left.

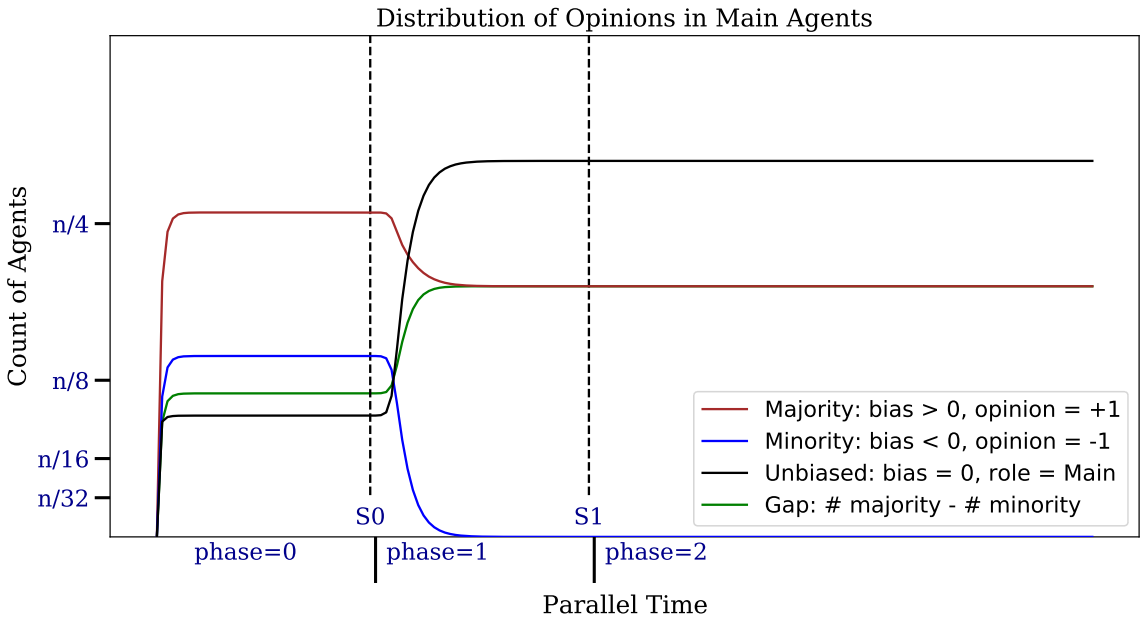


(e) Snapshot S4. After Phase 6, where Reserve agents with `sample` $\in \{-19, -20, -21\}$ enabled additional split reactions that brought all minority agents down.

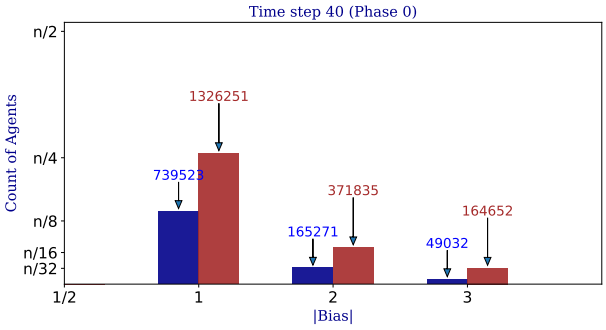


(f) Snapshot S5. After Phase 7, additional generalized cancel reactions eliminate all minority agents at the higher `exponent` values. At this point, there are no minority agents left, and we will stabilize to the correct output in Phase 9.

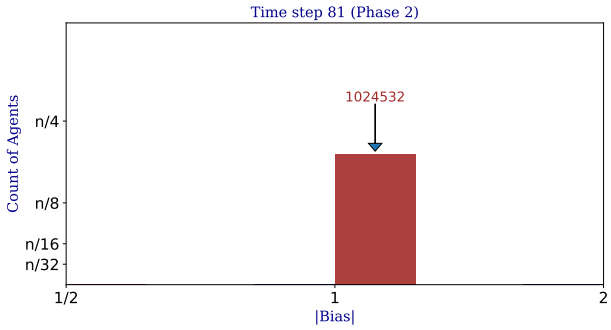
FIGURE 3.5. Snapshots S0, S1, S2, S3, S4, S5 from fig. 3.4d, showing the distribution of `bias` among agents with $|\texttt{bias}| > 0$. The red and blue bars give the count of A (majority, with `bias` > 0) and B (minority, with `bias` < 0) agents respectively. The exact counts are written above the bars, and everywhere not explicitly written the count of is 0. See the link to animations of these plots at the end of this section.



(a) The distribution of `opinion` in the Main agents in case of a linear size gap $g = n/10$. The red line gives the majority count (`opinion` $= +1$), the blue line the minority count (`opinion` $= -1$), and the green line their difference. The black line gives the unbiased $\mathcal{O}$ agents.



(b) Snapshot S0. At the end of Phase 0, the biased agents have $|\texttt{bias}| \in \{1, 2, 3\}$.



(c) Snapshot S1. The discrete averaging of Phase 1 first brings all `bias` $\in \{-1, 0, +1\}$. Then all `bias` $= -1$ cancel, leaving `bias` $= +1$ as the only nonzero bias. With the minority opinion eliminated, we converge in Phase 2.

FIGURE 3.6. The case of linear size gap $g = n/10$, again with $n = 5122666 \approx 2^{23}$, $p = 0.1$, $k = 2$. With large initial gap, the simulation converges in Phase 2. fig. 3.6a shows the counts of each opinion among the population of Main agents. Two special times S0 and S1, and the configurations of biased agents at these snapshots are shown in figs. 3.6b and 3.6c. See link to animations of these plots at the end of this section.

initial gap $g = +2$. This is our "typical" case, where the simulation eventually stabilizes to the correct output in Phase 9. fig. 3.6 shows linear initial gap $g \approx \frac{n}{10}$, which stabilizes in Phase 2 after

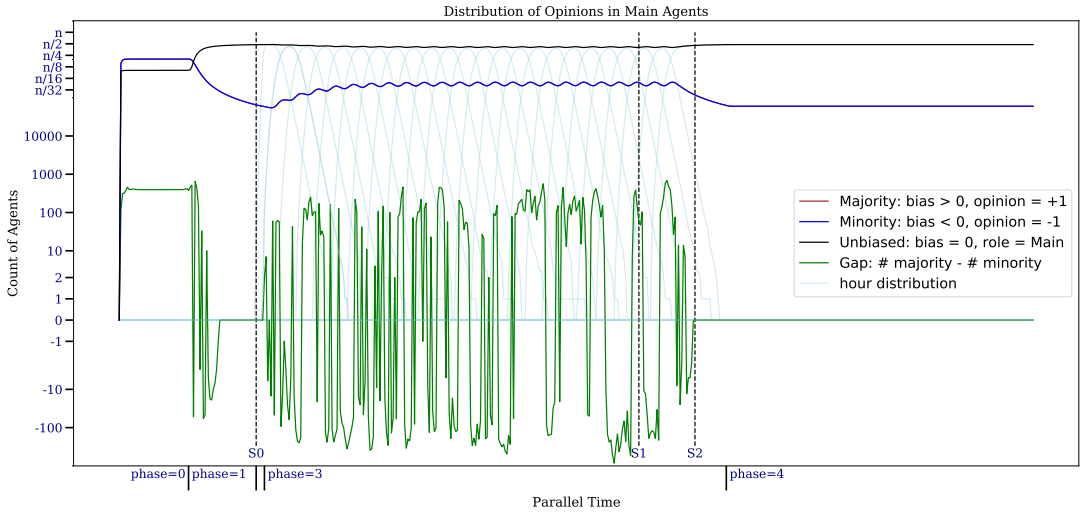


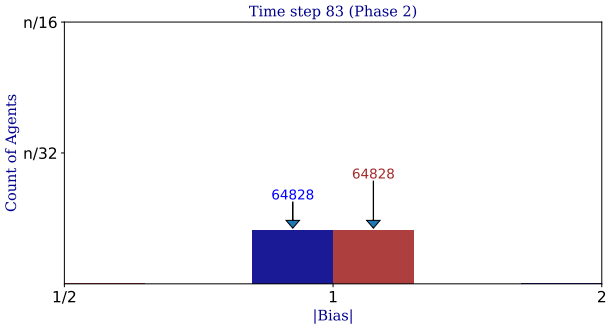
FIGURE 3.7. Showing a case of a tie. The simulation converges in Phase 2. fig. 3.6a shows the counts of each opinion among the population of Main agents. Three special times S0, S1, S2, and the configurations of biased agents at these snapshots are shown in Figures 3.8a, 3.8b, 3.8c. The distribution of `opinion` in the Main agents in case of a tie. The red line gives the majority count ($\texttt{opinion} = +1$), the blue line the minority count ($\texttt{opinion} = -1$), though they overlap everywhere, and the green line their difference. The black line gives the unbiased $\mathcal{O}$ agents. In Phase 3 we see the same qualitative behavior as the first part of Phase 3 with a constant initial gap in fig. 3.4d. Now this continues the whole phase, with the gap in counts oscillating about 0 until finally reaching 0 when all biased agents have $\texttt{exponent} = -23$ at time S2. With no $\texttt{exponent} > -23$, we stabilize to output T in Phase 4.

quickly cancelling all minority agents. fig. 3.8 shows initial tie $g = 0$, which stabilizes in Phase 4 after all biased agents reach the minimum $\texttt{exponent} = -L = -23$.

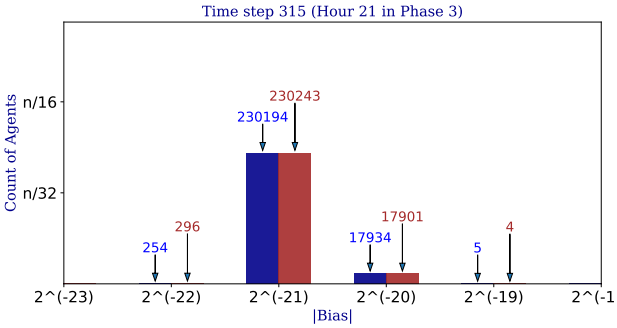
All three simulations show various snapshots of configurations of the biased agents. These particular snapshots are at special times marked in figs. 3.4d, 3.6a and 3.7. In all cases, an animation is available at GitHub [**43**], showing the full evolution of these distributions over all recorded time steps from the simulation.

3.5. Algorithm pseudocode

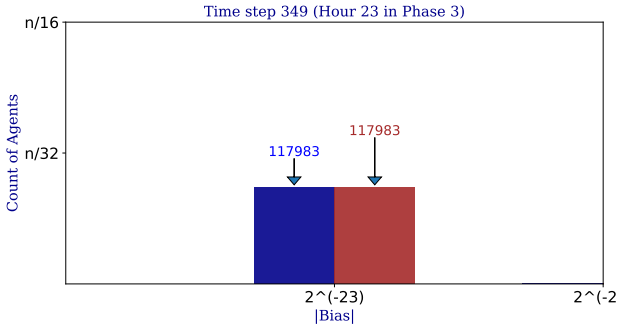
In this section we give a full formal description of the main algorithm. Every agent starts with a read-only field $\texttt{input} \in \{\mathsf{A}, \mathsf{B}\}$, a field $\texttt{output} \in \{\mathsf{A}, \mathsf{B}, \mathsf{T}\}$ corresponding to outputs that the majority is A, B, or a tie. The protocol is broken up into 11 consecutive phases, marked by the additional field $\texttt{phase} = 0 \in \{0, \ldots, 10\}$. The phase updates via the epidemic reaction $u.\texttt{phase}, v.\texttt{phase} \leftarrow \max(u.\texttt{phase}, v.\texttt{phase})$. Some fields are only used in particular phases, to



(a) Snapshot S0. We enter Phase 3 with an equal number of `bias` $= \pm 1$.



(b) Snapshot S1. We see similar distributions as in fig. 3.5b all the way until the end of Phase 3.



(c) Snapshot S2. After reaching synchronous hour 23, split reactions bring all remaining biased agents down to `exponent` $= -23$, which is only possible with an initial tie.

FIGURE 3.8. Snapshots from a case of a tie, with initial gap $g = 0$, again with $n = 5122666 \approx 2^{23}$, $p = 0.1$, $k = 2$.

ensure the total state space is $\Theta(\log n)$.[7] Such fields and the initial behavior of an agent upon entering a phase are described in the **Init** section above each phase. Whenever an agent increments their `phase`, they execute **Init** for the new phase (and sequentially for any phases in between if they happen to increment `phase` by more than 1). We refer to the "end of phase i" to mean the time when the first agent sets `phase` $\leftarrow$ `i` $+ 1$. Note that the agents actually enter each new phase by epidemic, so there is technically no well-defined "beginning of phase i". To simplify the analysis, we formally start our arguments for each phase assuming each agent is in the current phase, although technically $\Theta(\log n)$ time will pass between the time the first agent enters phase i and the last agent does.

[7]Note that using two fields, both with $O(\log n)$ possible values, requires $O(\log^2 n)$ states, not $O(\log n)$.

Each timed phase i each requires setting agents to count from $c_i \ln n$ down to 0, where the minimum required value of c_i depends on the phase. These constants can be derived from the technical analysis in section 3.6.2 and section 3.6.7 but for brevity we avoid giving them concrete values in the pseudocode. Our simulations (section 3.4) that used the same small constant $5 \log_2(n)$ for all counters seem to work, but the proofs require larger constants to ensure the necessary behavior within each phase can complete with high probability $1 - O(1/n^2)$. By increasing these constants c_i (along with changing the phase clock constants p, k; see Phase 3 and theorem 3.6.15), we could also push high probability bound to $1 - O(1/n^c)$ for any desired constant c. For concreteness, use $1 - O(1/n^2)$ for most high probability guarantees, since this is large enough to take appropriate union bounds and ensure the extra time from low probability failures does not contribute meaningfully to the total $O(\log n)$ time bound.

Nonuniform Majority$_L(u, v)$. Nonuniform majority algorithm for population sizes n with $L = \lceil \log n \rceil$.
Init: `phase` $\leftarrow 0 \in \{0, 1, \ldots, 10\}$ and execute **Init** for Phase 0.

1: **if** i.`phase` $<$ `j.phase` where $\{i, j\} = \{u, v\}$ **then**
2: **for** $p = \{i$.`phase` $+ 1, \ldots,$ `j.phase`$\}$ **do**
3: execute **Init** for **Phase** p on agent i
4: i.`phase` $\leftarrow$ `j.phase`
5: execute **Phase** u.`phase`(`u`, `v`)

Phase 0 is a timed phase that splits the population into three subpopulations: Main to compute majority, Clock to time the phases and the movement through exponents in Phase 3, and Reserve to aid in cleanup during Phase 6. An agent can only move into role Clock or Reserve by "donating" its opinion to a Main agent, who can collect up to two other opinions in addition to their own, leading to a *bias* of up to ± 3. After this phase, the populations of the three roles are near the expected one quarter Main, one quarter Clock, and one half Reserve. theorem 3.6.5 shows that all initial opinions have been given to assigned Main agents and these subpopulations are near their expected fractions, both with high probability.

It is likely that we could use a simpler population splitting scheme. For example, we could simply have each pair of agents with initially opposite opinions change to roles Clock and Reserve. However, this would mean the number of agents in the role Main after Phase 0 would depend on the initial gap. The current method of population splitting gives us stronger guarantees on the number of each agent in each role, simplifying subsequent analysis.

Phase 5 Initialize Roles. Agent u interacting with agent v.
Init $\texttt{role} \leftarrow \mathsf{Role}_{\mathsf{MCR}} \in \{\mathsf{Main}, \mathsf{Clock}, \mathsf{Reserve}, \mathsf{Role}_{\mathsf{MCR}}, \mathsf{Role}_{\mathsf{CR}}\}$
$\texttt{assigned} \leftarrow \mathsf{False} \in \{\mathsf{True}, \mathsf{False}\}$
if $\texttt{input} = \mathsf{A}$, $\texttt{bias} \leftarrow +1 \in \{-3, -2, -1, 0, +1, +2, +3\}$
if $\texttt{input} = \mathsf{B}$, $\texttt{bias} \leftarrow -1 \in \{-3, -2, -1, 0, +1, +2, +3\}$
we always maintain the invariant $\texttt{opinion} = \text{sign}(\texttt{bias}) \in \{-1, 0, +1\}$
if $\texttt{role} = \mathsf{Clock}$, $\texttt{counter} \leftarrow c_0 \ln n \in \{0, \ldots, c_0 \ln n\}$ only used in the current phase

1: **if** $u.\texttt{role} = v.\texttt{role} = \mathsf{Role}_{\mathsf{MCR}}$ **then** ▷ Allocate $\approx \frac{1}{2}\mathsf{Main}$ agents
2: $u.\texttt{role} \leftarrow \mathsf{Main}$; $u.\texttt{bias} \leftarrow u.\texttt{bias} + v.\texttt{bias}$
3: $v.\texttt{bias} \leftarrow 0$; $v.\texttt{role} \leftarrow \mathsf{Role}_{\mathsf{CR}}$ ▷ v won't use its bias subsequently
4: **if** $i.\texttt{role} = \mathsf{Role}_{\mathsf{MCR}}$, $j.\texttt{role} = \mathsf{Main}$, $j.\texttt{assigned} = \mathsf{False}$ where $\{i, j\} = \{u, v\}$ **then**
5: $j.\texttt{assigned} \leftarrow \mathsf{True}$; $j.\texttt{bias} \leftarrow j.\texttt{bias} + i.\texttt{bias}$ ▷ Main agents can assign 1 non-Main agent
6: $i.\texttt{bias} \leftarrow 0$; $i.\texttt{role} \leftarrow \mathsf{Role}_{\mathsf{CR}}$ ▷ i won't use its bias subsequently
7: **if** $i.\texttt{role} = \mathsf{Role}_{\mathsf{MCR}}$, $j.\texttt{role} \neq \mathsf{Main}, \mathsf{Role}_{\mathsf{MCR}}$, $j.\texttt{assigned} = \mathsf{False}$ where $\{i, j\} = \{u, v\}$ **then**
8: $j.\texttt{assigned} \leftarrow \mathsf{True}$ ▷ non-Main agents can assign 1 Main agent
9: $i.\texttt{role} \leftarrow \mathsf{Main}$
10: **if** $u.\texttt{role} = v.\texttt{role} = \mathsf{Role}_{\mathsf{CR}}$ **then** ▷ Allocate $\approx \frac{1}{4}\mathsf{Clock}$ agents, $\approx \frac{1}{4}\mathsf{Reserve}$ agents
11: $u.\texttt{role} \leftarrow \mathsf{Clock}$; $u.\texttt{counter} \leftarrow \Theta(\log n)$
12: $v.\texttt{role} \leftarrow \mathsf{Reserve}$
13: **if** $u.\texttt{role} = v.\texttt{role} = \mathsf{Clock}$ **then** ▷ time the phase once we have at least 2 Clock agents
14: execute **Standard Counter Subroutine**(u), execute **Standard Counter Subroutine**(v)

Standard Counter Subroutine(c). Agent c with field $\texttt{counter}$.

1: $c.\texttt{counter} \leftarrow c.\texttt{counter} - 1$
2: **if** $c.\texttt{counter} = 0$ **then**
3: $c.\texttt{phase} \leftarrow \texttt{c.phase} + 1$ ▷ move to next phase

Phase 1 is a timed phase that averages the biases in Main agents; with high probability at the end of the phase the $\texttt{bias}$ fields have three consecutive values, shown in theorem 3.6.6. Phase 1 expects no agent to remain in role $\mathsf{Role}_{\mathsf{MCR}}$ by this point, signaling an error otherwise.[8]

[8] $\texttt{role} = \mathsf{Role}_{\mathsf{MCR}}$ is an error because we need all agents not in role Main to have "donated" their bias to a Main agent. It is okay for $\mathsf{Role}_{\mathsf{CR}}$ agents to be undecided about Clock versus Reserve. All such agents because Reserve, which WHP leaves sufficiently many agents in role Clock to make the clock interactions sufficiently fast.

Phase 6 Discrete Averaging. Agent u interacting with agent v.
Init if $\texttt{role} = \mathsf{Role}_{\mathsf{MCR}}$, $\texttt{phase} \leftarrow 10$ (error, skip to stable backup)
if $\texttt{role} = \mathsf{Role}_{\mathsf{CR}}$, $\texttt{role} \leftarrow \mathsf{Reserve}$
if $\texttt{role} = \mathsf{Clock}$, $\texttt{counter} \leftarrow c_1 \ln n \in \{0, \ldots, c_1 \ln n\}$ only used in the current phase

1: **if** $u.\texttt{role} = v.\texttt{role} = \mathsf{Main}$ **then**
2: $\quad u.\texttt{bias} \leftarrow \lfloor \frac{u.\texttt{bias}+v.\texttt{bias}}{2} \rfloor$; $v.\texttt{bias} \leftarrow \lceil \frac{u.\texttt{bias}+v.\texttt{bias}}{2} \rceil$
3: **for** $c \in \{u, v\}$ with $c.\texttt{role} = \mathsf{Clock}$ **do**
4: $\quad$ execute **Standard Counter Subroutine**(c)

Phase 2 (an untimed phase) checks to see if the entire minority population was eliminated in Phase 1 by checking whether both positive and negative biases still exist. It assumes a starting condition where all $\texttt{bias} \in \{-1, 0, +1\}$ (like the initial condition, but allowing some cancelling to have already happened). So the **Init** checks if any $|\texttt{bias}| > 1$, which can only happen with low probability (since Phase 1 WHP reaches the three consecutive values $\{-1, 0, +1\}$), and we consider this an error and simply proceed immediately to Phase 10. If not, the minority opinion is gone, and the protocol will stabilize here to the correct output. Otherwise, we proceed to the next phase; note this phase is untimed and proceeds immediately upon detection of conflicting opinions. theorem 3.6.6 shows that starting from a large initial gap, we will stabilize here with high probability.

Phase 7 Output the Consensus. Agent u interacting with agent v.
Init if $|\texttt{bias}| > 1$, $\texttt{phase} \leftarrow 10$ (error, skip to stable backup)
$\texttt{opinions} \leftarrow \{\texttt{opinion}\} \subseteq \{-1, 0, +1\}$

1: $u.\texttt{opinions}, v.\texttt{opinions} \leftarrow u.\texttt{opinions} \cup v.\texttt{opinions}$ ▷ union opinions
2: **if** $\{-1, +1\} \subseteq \texttt{opinions}$ **then**
3: $\quad u.\texttt{phase}, \texttt{v.phase} \leftarrow \texttt{u.phase} + 1$ ▷ no consensus, move to next phase
4: **else if** $+1 \in \texttt{opinions}$ **then**
5: $\quad u.\texttt{output}, v.\texttt{output} \leftarrow \mathsf{A}$ ▷ current consensus is A
6: **else if** $-1 \in \texttt{opinions}$ **then**
7: $\quad u.\texttt{output}, v.\texttt{output} \leftarrow \mathsf{B}$ ▷ current consensus is B
8: **else if** $\texttt{opinions} = \{0\}$ **then**
9: $\quad u.\texttt{output}, v.\texttt{output} \leftarrow \mathsf{T}$ ▷ current consensus is T

Phase 3 is where the bulk of the work gets done. The biased agents (with non-zero `opinion`) have an additional field `exponent` $\in \{-L, \ldots, 0\}$, initially 0, where $L = \lceil \log_2(n) \rceil$, corresponding to holding $2^{\texttt{exponent}}$ units of mass. The unbiased $\mathcal{O}$ agents have an additional field `hour` $= L \in \{0, \ldots, L\}$, and will only participate in split reactions with $-\texttt{exponent} > \texttt{hour}$. The field `hour` is set by the Clock agents, who have a field `minute` $= 0 \in \{0, \ldots, kL\}$, which counts up as Phase 3 proceeds. Intuitively, there are k minutes in an hour, so `hour` $= \lceil \frac{\texttt{minute}}{k} \rceil$ ranges from 0 to L.[9] fig. 3.3 shows how nonconsecutive minutes in the Clock agents may overlap significantly, but nonconsecutive hours in the $\mathcal{O}$ agents have negligible overlap. Once a Clock agent has `minute` at its maximum value kL, they initialize a new field `counter` $= \Theta(\log n) \in \{0, \ldots, \Theta(\log n)\}$ to wait for the end of the phase (waiting for any remaining $\mathcal{O}$ agents to reach `hour` $= L$ and the distribution to settle).

See the overview in section 3.3.2 for an intuitive description of this phase. theorem 3.6.7 gives the main result of Phase 3 in the case of an initial tie, where all remaining biased agents are at the minimal `exponent` $= -L$. In the other case, theorem 3.6.8 gives the main result that most of the population settle on the majority output with `exponent` $\in \{-l, -(l+1), -(l+2)\}$.

[9] The drip reaction $C_i, C_i \to C_i, C_{i+1}$ implemented by line 5 is asymmetric, but could be made symmetric, i.e., $C_i, C_i \to C_{i+1}, C_{i+1}$ without affecting the analysis meaningfully.

Phase 8 Synchronized Rational Averaging. Agent u interacting with agent v.
Init if $\texttt{role} = \mathsf{Main}$ and $\texttt{opinion} \in \{-1, +1\}$, $\texttt{exponent} \leftarrow 0 \in \{-L, \ldots, -1, 0\}$, and we define $\texttt{bias} = \texttt{opinion} \cdot 2^{\texttt{exponent}}$
if $\texttt{role} = \mathsf{Main}$ and $\texttt{opinion} = 0$, $\texttt{hour} \leftarrow 0 \in \{0, \ldots, L\}$
if $\texttt{role} = \mathsf{Clock}$, $\texttt{minute} \leftarrow 0 \in \{0, \ldots, kL\}$ and $\texttt{counter} \leftarrow c_3 \ln n \in \{0, \ldots, c_3 \ln n\}$

1: **if** $u.\texttt{role} = v.\texttt{role} = \mathsf{Clock}$ **then**
2: **if** $u.\texttt{minute} \neq v.\texttt{minute}$ **then**
3: $u.\texttt{minute}, v.\texttt{minute} \leftarrow \max(u.\texttt{minute}, v.\texttt{minute})$ ▷ clock epidemic reaction
4: **else if** $u.\texttt{minute} < kL$ **then**
5: $u.\texttt{minute} \leftarrow u.\texttt{minute} + 1$ with probability p ▷ clock drip reaction, $p = 1$ in theorem 3.6.15
6: **else** ▷ count only when both clocks finished
7: execute **Standard Counter Subroutine**(u), **Standard Counter Subroutine**(v)
8: **if** $m.\texttt{role} = \mathsf{Main}, m.\texttt{opinion} = 0$ and $c.\texttt{role} = \mathsf{Clock}$ where $\{m, c\} = \{u, v\}$ **then**
9: $m.\texttt{hour} \leftarrow \max\left(m.\texttt{hour}, \lfloor \frac{c.\texttt{minute}}{k} \rfloor\right)$ ▷ clock update reaction
10: **else if** $u.\texttt{role} = v.\texttt{role} = \mathsf{Main}$ **then**
11: **if** $\{u.\texttt{opinion}, v.\texttt{opinion}\} = \{-1, +1\}$ and $u.\texttt{exponent} = v.\texttt{exponent} = -h$ **then**
12: $u.\texttt{opinion}, v.\texttt{opinion} \leftarrow 0$; $u.\texttt{hour}, v.\texttt{hour} \leftarrow h$ ▷ cancel reaction
13: **if** $t.\texttt{opinion} = 0$, $i.\texttt{opinion} \in \{-1, +1\}$ and $|t.\texttt{hour}| > |i.\texttt{exponent}|$, where $\{t, i\} = \{u, v\}$ **then**
14: $t.\texttt{opinion} \leftarrow i.\texttt{opinion}$ ▷ split reaction
15: $i.\texttt{exponent}, t.\texttt{exponent} \leftarrow i.\texttt{exponent} - 1$ ▷ sets $i.\texttt{bias}, t.\texttt{bias} \leftarrow i.\texttt{bias}/2$

In Phase 4 (an untimed phase), the population checks if all Main agents have reached the minimum $\texttt{exponent} = -L$, which only happens in the case of a tie. If so, the population will stabilize to the tie output. Otherwise, any Main agent with $\texttt{exponent}$ above L can trigger the move to the next phase.

Phase 9 Output Tie. Agent u interacting with agent v.
Init $\texttt{output} \leftarrow \mathsf{T}$

1: **if** $|m.\texttt{bias}| > 2^{-L}$ where $m \in \{u, v\}$ **then** ▷ stable if all $\texttt{bias} \in \left\{-\frac{1}{2^L}, 0, +\frac{1}{2^L}\right\}$
2: $u.\texttt{phase}, \texttt{v.phase} \leftarrow \texttt{u.phase} + 1$ ▷ end of this phase

If there was not a tie detected in Phase 4, then most agents should have the majority opinion, with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$ in a small range. The next goal is to bring all agents with $\texttt{exponent} > -l$ down to $\texttt{exponent} \leq -l$. This will be accomplished by the Reserve agents, whose goal is to let exponents above l do split reactions.

The Reserve agents do this across two consecutive phases. In Phase 5, the Reserve agents become active, and will set $\texttt{sample} = \perp \in \{\perp, 0, \ldots, L\}$ to the exponent of the first biased agent they meet, which is likely in $\{-l, -(l+1), -(l+2)\}$. The Clock agents now only hold a field $\texttt{counter} = \Theta(\log n) \in \{0, \ldots, \Theta(\log n)\}$ to act as a simple timer for how long to wait until moving to the next phase. This allows the Reserve agents to adopt a distribution of exponent values approximately equal to that of the Main agents. This behavior is proven in theorem 3.6.25 and theorem 3.6.26.

Phase 10 Reserves Sample exponent. Agent u interacting with agent v.
Init if $\texttt{role} = \text{Reserve}$, $\texttt{sample} \leftarrow \perp \in \{\perp, -L, \ldots, -1, 0\}$
if $\texttt{role} = \text{Clock}$, $\texttt{counter} \leftarrow c_5 \ln n \in \{0, \ldots, c_5 \ln n\}$

1: **if** $r.\texttt{role} = \text{Reserve}$ and $m.\texttt{role} = \text{Main}, m.\texttt{opinion} \in \{-1, +1\}$ where $\{r, m\} = \{u, v\}$ **then**
2: **if** $r.\texttt{sample} = \perp$ **then** ▷ sample exponent of biased agent m
3: $r.\texttt{sample} \leftarrow m.\texttt{exponent}$
4: **for** $c \in \{u, v\}$ with $c.\texttt{role} = \text{Clock}$ **do**
5: execute **Standard Counter Subroutine**(c)

In Phase 6, the Reserve agents can help facilitate more split reactions, with any agent at an exponent above their sampled exponent. Because they have approximately the same distribution across all exponents as Main agents, particular exponents $-l, -(l+1), -(l+2)$, this allows them to bring all agents above $\texttt{exponent} = -l$ down to $-l$ or below. Again, the Clock agents keep a $\texttt{counter} \in \{0, \ldots, c_6 \ln n\}$. theorem 3.6.26 proves that Phase 6 works as intended, bringing all agents down to $\texttt{exponent} \leq -l$ with high probability.

Phase 11 Reserve Splits. Agent u interacting with agent v.
Init if $\texttt{role} = \mathsf{Clock}$, $\texttt{counter} \leftarrow c_6 \ln n \in \{0, \ldots, c_6 \ln n\}$

if $r.\texttt{role} = \mathsf{Reserve}$ and $m.\texttt{role} = \mathsf{Main}, m.\texttt{opinion} \in \{-1, +1\}$ where $\{r, m\} = \{u, v\}$ **then**
  if $r.\texttt{sample} \neq \perp$ and $r.\texttt{sample} < m.\texttt{exponent}$ **then**
    $r.\texttt{role} \leftarrow \mathsf{Main}$; $r.\texttt{opinion} \leftarrow m.\texttt{opinion}$ ▷ split reaction
    $r.\texttt{exponent}, m.\texttt{exponent} \leftarrow m.\texttt{exponent} - 1$ ▷ sets $r.\texttt{bias}, m.\texttt{bias} \leftarrow m.\texttt{bias}/2$
for $c \in \{u, v\}$ with $c.\texttt{role} = \mathsf{Clock}$ **do**
  execute **Standard Counter Subroutine**(c)

Now that all agents are $\texttt{exponent} \leq -l$, the goal of Phase 7 is to eliminate any minority agents with exponents $-l, -(l+1), -(l+2)$. This is done by letting the biased agents do generalized cancel reactions that allow their difference in exponents to be up to 2, while still preserving the mass invariant. Again, the Clock agents keep a $\texttt{counter} \in \{0, \ldots, c_7 \ln n\}$. theorem 3.6.29 shows that by the end of this phase, any remaining minority agents must have $\texttt{exponent} < -(l+2)$, with high probability.

Phase 12 High-exponent Minority Elimination. Agent u interacting with agent v
Init if $\texttt{role} = \mathsf{Clock}$, $\texttt{counter} \leftarrow c_7 \ln n \in \{0, \ldots, c_7 \ln n\}$

if $u.\texttt{role} = v.\texttt{role} = \mathsf{Main}$ and $\{u.\texttt{opinion}, v.\texttt{opinion}\} = \{-1, +1\}$ **then**
  if $u.\texttt{exponent} = v.\texttt{exponent}$ **then**
    $u.\texttt{opinion}, v.\texttt{opinion} \leftarrow 0$ ▷ cancel reaction
  else if $i.\texttt{exponent} = j.\texttt{exponent} + 1$ where $\{i, j\} = \{u, v\}$ **then**
    $i.\texttt{exponent} \leftarrow i.\texttt{exponent} - 1$ ▷ gap-1 cancel reaction
    $j.\texttt{opinion} \leftarrow 0$ ▷ example $\texttt{bias}$ update: $+\frac{1}{4}, -\frac{1}{8} \to +\frac{1}{8}, 0$
  else if $i.\texttt{exponent} = j.\texttt{exponent} + 2$ where $\{i, j\} = \{u, v\}$ **then**
    $j.\texttt{opinion} \leftarrow i.\texttt{opinion}$ ▷ gap-2 cancel reaction
    $i.\texttt{exponent} \leftarrow i.\texttt{exponent} - 1$ ▷ example $\texttt{bias}$ update: $+\frac{1}{4}, -\frac{1}{16} \to +\frac{1}{8}, +\frac{1}{16}$
    $j.\texttt{exponent} \leftarrow i.\texttt{exponent} - 2$
for $c \in \{u, v\}$ with $c.\texttt{role} = \mathsf{Clock}$ **do**
  execute **Standard Counter Subroutine**(c)

Now that all minority agents occupy exponents below $-(l+2)$, yet a large number of majority agents remain at exponents $-l, -(l+1), -(l+2)$, in Phase 8, the algorithm eliminates the last remaining minority opinions at any exponent. It allows opposite-opinion agents of *any* two exponents to react and eliminate the smaller-exponent opinion. The larger exponent $\frac{1}{2^i}$ "absorbs" the smaller $-\frac{1}{2^j}$, setting the smaller to mass 0; the larger now represents mass $\frac{1}{2^i} - \frac{1}{2^j}$, which it lacks the memory to track exactly, so it cannot absorb any further agents (though it can itself be absorbed by $-\frac{1}{2^m}$ for $m > i$). theorem 3.6.30 shows that Phase 8 eliminates any remaining minority agents with high probability. A key property is that it cannot violate correctness since, although the biases held by some agents becomes unknown, the allowed transitions maintain the sign of the bias. Thus the only source of error is failing to eliminate all minority agents, detected in the next phase.

Phase 13 Low-exponent Minority Elimination. Agent u interacting with agent v
Init if `role` = Clock, `counter` $\leftarrow c_8 \ln n \in \{0, \ldots, c_8 \ln n\}$

1: **if** u.`role` = v.`role` = Main and $\{u$.`bias`$, v$.`bias`$\} = \{-1, +1\}$ **then**
2: **if** i.`exponent` > j.`exponent` and i.`full` = False where $\{i, j\} = \{u, v\}$ **then**
3: i.`full` $\leftarrow$ True ▷ consumption reaction
4: j.`opinion` $\leftarrow 0$
5: **for** $c \in \{u, v\}$ with c.`role` = Clock **do**
6: execute **Standard Counter Subroutine**(c)

Phase 9 (an untimed phase) acts exactly as Phase 2, to check that agents have reached consensus. Note that the initial check for |`bias`| > 1 is not required, since it is guaranteed to pass if we reach this point: it passed in Phase 2, and the population-wide maximum |`bias`| could only have decreased in subsequent phases.

Phase 14 Output the Consensus. Exact repeat of Phase 2.

In Phase 10, the agents give up on the fast algorithm, having determined in Phase 9 that it failed to reach consensus, or detected an earlier error with `role` or `bias` assignment. Instead they rely instead on a slow stable backup protocol. This is a 6-state protocol that stably decides between the three cases of majority A, B, and tie T. They only use the fields `output` = `input` $\in \{$A, B, T$\}$,

and the initial field `active` = True $\in$ {True, False}. theorem 3.6.31 proves this 6-state protocol stably computes majority in $O(n \log n)$ time.

Phase 15 Stable Backup. Agent u interacting with agent v. Similar to 6-state algorithm from [31], slight modification of 4-state algorithm from [54, 73].
Init `output` $\leftarrow$ `input`, `active` $\leftarrow$ True

1: **if** u.`active` = v.`active` = True **then**
2: **if** $\{u$.`output`$, v$.`output`$\} = \{$A, B$\}$ **then**
3: u.`output`$, v$.`output` $\leftarrow$ T ▷ cancel reaction
4: **else if** i.`output` $\in \{$A, B$\}$ and t.`output` = T where $\{i, t\} = \{u, v\}$ **then**
5: t.`output` $\leftarrow i$.`output`, t.`active` $\leftarrow$ False ▷ biased converts unbiased
6: **if** a.`active` = True and p.`active` = False where $\{a, p\} = \{u, v\}$ **then**
7: p.`output` $\leftarrow a$.`output` ▷ active converts passive

3.6. Analysis of exact majority protocol

3.6.1. Useful time bounds. This section introduces various probability bounds which will be used repeatedly in later analysis.

We use the standard Azuma inequality for supermartingales:

THEOREM 3.6.1. *Let $X_0, X_1, X_2, \ldots$ be a supermartingale such that, for all $i \in \mathbb{N}$, $|X_{i+1} - X_i| \leq c_i$. Then for all $n \in \mathbb{N}$ and $\epsilon > 0$, $\Pr\left[(X_n - X_0) - \mathbb{E}\left[X_n - X_0\right] \geq \epsilon\right] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=0}^{n} c_i^2}\right)$.*

In application we will consider potential functions ϕ that decay exponentially, with $\mathbb{E}\left[\phi_{j+1}\right] \leq (1-\epsilon)\phi_j$. In this case, we will take the logarithm $\Phi = \ln(\phi)$, which we will show is a supermartingale upon which we can apply Azuma's Inequality to conclude that ϕ achieves a requisite amount of exponential decay.

Next we consider the cancel reactions $a, b \to 0, 0$, which are key to the majority protocol.

LEMMA 3.6.2. *Consider two disjoint subpopulations A and B of initial sizes $|A| = a \cdot n$ and $|B| = b \cdot n$, where $0 < b < a < 1$. An interaction between an agent in A and an agent in B is a* cancel reaction *which removes both agents from their subpopulations. Thus after i cancel reactions we have $|A| = a \cdot n - i$ and $|B| = b \cdot n - i$.*

The expected parallel time t until $d \cdot n$ cancel reactions occur, where $d < b$ is

$$\mathbb{E}[t] = \frac{\ln(b) - \ln(a) - \ln\left(b - d + \frac{1}{n}\right) + \ln\left(a - d + \frac{1}{n}\right)}{2(a-b)}.$$

Let $c = (a - d + \frac{1}{n})(b - d + \frac{1}{n})$ and $0 < \epsilon < 1$. Then $(1-\epsilon)\mathbb{E}[t] < t < (1+\epsilon)\mathbb{E}[t]$ with probability at least $1 - \exp\left(-\Theta(\epsilon^2 \mathbb{E}[t]\, nc)\right)$.

The parallel time t until all $b \cdot n$ cancel reactions occur has $\mathbb{E}[t] \sim \frac{\ln n}{2(a-b)}$ and satisfies $t \leq \frac{5 \ln n}{2(a-b)}$ with high probability $1 - O(1/n^2)$.

Again, the first bound is with very high probability if all constants a, b, d are independent of n.

Proof. After i cancel reactions, the probability of the next cancel reaction is $p \sim \frac{2|A| \cdot |B|}{n^2} = 2(a - \frac{i}{n})(b - \frac{i}{n})$, and the number of interactions until this cancel reaction is a geometric random variable with mean p. The number of interactions T for $d \cdot n$ cancel reactions to occur is a sum of geometrics with mean

$$\begin{aligned}\mathbb{E}[T] &= \sum_{i=0}^{d \cdot n - 1} \frac{1}{2(a - \frac{i}{n})(b - \frac{i}{n})} \sim n \int_{x=0}^{d - \frac{1}{n}} \frac{dx}{2(a-x)(b-x)} = \frac{n}{2(a-b)} \int_{x=0}^{d - \frac{1}{n}} \left(\frac{1}{b-x} - \frac{1}{a-x}\right) dx \\ &= \frac{n}{2(a-b)} \Big[-\ln(b-x) + \ln(a-x) \Big]_0^{d - \frac{1}{n}} = n \cdot \frac{\ln(b) - \ln(a) - \ln\left(b - d + \frac{1}{n}\right) + \ln\left(a - d + \frac{1}{n}\right)}{2(a-b)}.\end{aligned}$$

Translating to parallel time and using theorem 1.3.3 gives the first result, where minimum probability $p^* = \Theta(c)$.

In the second case where $d = b$ and we are waiting for all cancel reactions to occur, then

$$\mathbb{E}[t] \sim \frac{\ln(b) - \ln(a) + \ln(n) + \ln(a-b)}{2(a-b)} = \frac{\ln n}{2(a-b)},$$

and $\mu = \mathbb{E}[T] \sim \frac{n \ln n}{2(a-b)}$. Now the minimum geometric probability is when $i = bn - 1$ and $p^* \sim \frac{2(a-b)}{n}$. Choosing $\lambda = 5$ so that $\lambda - 1 - \ln \lambda > 2$, by theorem 1.3.2 we have

$$\Pr\left[t \geq \frac{5 \ln n}{2(a-b)}\right] \leq \Pr[T \geq \lambda \mu] \leq e^{-p^* \mu(\lambda - 1 - \ln \lambda)} = \exp\left(-\frac{2(a-b)}{n} \cdot \frac{n \ln n}{2(a-b)} \cdot 2\right) = n^{-2}.$$

Thus $t \leq \frac{5 \ln n}{2a}$ with high probability.

Note we get the same result for this second case also by assuming a fixed minimal fraction $a - b$ in $|A|$ is always there to cancel the agents from $|B|$, and using the next theorem 3.6.3. □

Now we consider a "one-sided" cancel process, where reactions $a, b \to a, 0$ change only the b agents into a different state. This process happens for example when Clock agents change the `hour` of $\mathcal{O}$ agents.

LEMMA 3.6.3. *Let $0 < a, b_1, b_2, \epsilon < 1$ be constants with $b_1 > b_2$. Consider a subpopulation A maintaing its size above $a \cdot n$, and B initially of size $b_1 \cdot n$. Any interaction between an agent in A and in B is* meaningful*, and forces the agent in B to leave its subpopulation.*

The expected parallel time t until the subpopulation B reaches size $b_2 \cdot n$ is

$$\mathbb{E}[t] = \frac{\ln(b_1) - \ln(b_2)}{2a},$$

and satisfies $(1-\epsilon)\mathbb{E}[t] < t < (1+\epsilon)\mathbb{E}[t]$ with probability at least $1 - \exp\left(-\Theta(\epsilon^2 \mathbb{E}[t]\, nab_2)\right)$.

The parallel time t until the subpopulation B reaches size 0 has $\mathbb{E}[t] \sim \frac{\ln n}{2a}$ and satisfies $t \leq \frac{5 \ln n}{2a}$ with high probability $1 - O(1/n^2)$.

Again, the first bound is with very high probability if constants a, b_2 are independent of n.

PROOF. When $|B| = i$, then the probability of a meaningful interaction $p \sim \frac{2ia}{n}$. Then the number of interactions before the next meaningful interaction is a geometric with probability p, and the total number T of interaction is a sum of geometrics with mean

$$\mathbb{E}[T] = \sum_{i=b_2 n+1}^{b_1 n} \frac{n}{2ia} = \frac{n}{2a}\left(\sum_{i=1}^{b_1 n} \frac{1}{i} - \sum_{i=1}^{b_2 n+1} \frac{1}{i}\right) \sim \frac{n}{2a}(\ln(b_1 n) - \ln(b_2 n + 1)) \sim n\frac{\ln(b_1) - \ln\left(b_2 + \frac{1}{n}\right)}{2a}.$$

Translating to parallel time and using theorem 1.3.3 gives the first result, where minimum probability $p^* = \Theta(a \cdot b_2)$.

In the case where $b_2 = 0$, we have $\mu = \mathbb{E}[T] \sim \frac{n \ln n}{2a}$ and $\mathbb{E}[t] \sim \frac{\ln(n)}{2a}$. Now the minimum geometric probability $p^* = \frac{2a}{n}$. Choosing $\lambda = 5$ so that $\lambda - 1 - \ln \lambda > 2$, by theorem 1.3.2 we have

$$\Pr\left[t \geq \frac{5 \ln n}{2a}\right] \leq \Pr[T \geq \lambda\mu] \leq e^{-p^*\mu(\lambda - 1 - \ln\lambda)} = \exp\left(-\frac{2a}{n} \cdot \frac{n \ln n}{2a} \cdot 2\right) = n^{-2}.$$

Thus $t \leq \frac{5 \ln n}{2a}$ with high probability. □

3.6.2. Analysis of initial phases. Define $\mathcal{M}$, $\mathcal{C}$, and $\mathcal{R}$ to be the sub-populations of agents with roles Main, Clock, Reserve when they first set `phase` $= 1$. Let $|\mathcal{M}| = m \cdot n, |\mathcal{C}| = c \cdot n, |\mathcal{R}| = r \cdot n$, where $m + c + r = 1$.

The population splitting of Phase 0 will set the fractions $m \approx \frac{1}{2}$, $c \approx \frac{1}{4}$, and $r \approx \frac{1}{4}$. The rules also ensure deterministic bounds on these fractions once all $\mathsf{Role}_{\mathsf{MCR}}$ agents have been assigned. The probability 1 guarantees on subpoplation sizes using this method will be key for a later uniform adaptation of the protocol. We first prove the behavior of the rules used for this initial top level split in Phase 0 between the roles Main and $\mathsf{Role}_{\mathsf{CR}}$.

LEMMA 3.6.4. *Consider the reactions*

$$U, U \to S_f, M_f$$

$$S_f, U \to S_t, M_f$$

$$M_f, U \to M_t, S_f$$

starting with n U agents. Let $u = \#U$, $s = \#S_f + \#S_t$, and $m = \#M_f + \#M_t$. This converges to $u = 0$ in expected time at most $2.5 \ln n$ and in $12.5 \ln n$ time with high probability $1 - O(1/n^2)$. Once $u = 0$, $\frac{n}{3} \leq s, m \leq \frac{2n}{3}$ with probability 1, and for any $\epsilon > 0$, $\frac{n}{2}(1-\epsilon) \leq s, m \leq \frac{n}{2}(1+\epsilon)$ with very high probability.

PROOF. Let $s_f = \#S_f, s_t = \#S_t, m_f = \#M_f, m_t = \#M_t$.

First we consider the interactions that happen until $u = 2n/3$. Note that while $u \geq 2n/3$, the probability of the first reaction is $\sim \left(\frac{u}{n}\right)^2 \geq \frac{4}{9}$ and the probability of the other two reactions is $\sim 2\left(\frac{u}{n}\right)\left(\frac{m_f+s_f}{n}\right) \leq 2 \cdot \frac{2}{3} \cdot \frac{1}{3} = \frac{4}{9}$. Thus until $u = 2n/3$, each non-null interaction is the top reaction with at least probability $1/2$. Then by standard Chernoff Bounds, with very high probability, at least a fraction $\frac{1}{2} - \epsilon$ of these reactions are the top reaction, which create a count $s_f + m_f \geq \frac{2}{9}(1-\epsilon)n > \frac{n}{5}$.

Now notice that this count $s_f + m_f$ can never decrease, so we have $s_f + m_f > \frac{n}{5}$ for all future interactions. Now we can use the rate of the second and third reactions to bound the completion time. The probability of decreasing u is at least $2\left(\frac{u}{n}\right)\left(\frac{1}{5}\right)$, so the number of interactions it takes to decrement u is stochastically dominated by a geometric random variable with probability $p = \frac{2u}{5n}$. Then the number of interactions for u to decrease from $\frac{2n}{3}$ down to 0 is dominated by a sum T of geometric random variables with mean

$$\mathbb{E}[T] = \sum_{u=1}^{2n/3} \frac{5n}{2u} = \frac{5n}{2} \sum_{u=1}^{2n/3} \frac{1}{u} \sim \frac{5n}{2} \ln(2n/3) \sim \frac{5}{2} n \ln n.$$

Now we will apply the upper bound of theorem 1.3.2, using $\mu = \frac{5}{2}n \ln n$, $p^* = \frac{2}{5n}$ (when $u = 1$) and $\lambda = 5$, where $\lambda - 1 - \ln(\lambda) > 2$, so we get

$$\Pr[T \geq \lambda\mu] \leq \exp(-p^*\mu(\lambda - 1 - \ln\lambda)) \leq \exp\left(-\frac{2}{5n} \cdot \frac{5}{2} n \ln n \cdot 2\right) = n^{-2}.$$

Thus with high probability $1 - O(1/n^2)$, this process converges in at most $\frac{\lambda\mu}{n} = 12.5 \ln n$ parallel time.

Observe that the reactions all preserve the following invariant: $s_f + 2s_t = m_f + 2m_t$. The probability-1 count bounds follow by observing that when $u = 0$ (i.e., we have converged) we have $s_f + s_t + m_f + m_t = n$. Maximizing $s = s_f + s_t$ and minimizing $m = m_f + m + t$ is achieved by setting $s_f = 2n/3, s_t = 0, m_f = 0, m_t = n/3$, and symmetrically for maximizing m.

Assume we have $s > m$ at some point in the execution, so $s_f + s_t > m_f + m_t$. Subtracting the invariant equation gives $-s_t > -m_t$, which implies $s_t < m_t$. Together with the first inequality this implies that $s_f > m_f$. Thus the rate of second reaction is higher than the rate of the third reaction, so it is more likely that the next reaction changing the value $s - m$ decreases it.

By symmetry the opposite happens when $m > s$, so we conclude that the absolute value $|m - s|$ is more likely to decrease than increase. Thus we can stochastically dominate $|m - s|$ by a sum of independent coin flips. The high probability count bounds follow by standard Chernoff bounds. □

Now we can prove bounds on the sizes $|\mathcal{M}| = mn, |\mathcal{C}| = cn, |\mathcal{R}| = rn$ of Main, Clock, and Reserve agents from the population splitting of Phase 0.

Lemma 3.6.5. *For any $\epsilon > 0$, with high probability $1 - O(1/n^2)$, by the end of Phase 0, $|\mathsf{Role}_{\mathsf{MCR}}| = 0$, $\frac{n}{2}(1 - \epsilon) \leq |\mathcal{M}| \leq \frac{n}{2}(1 + \epsilon)$ and $|\mathcal{C}|, |\mathcal{R}| \geq \frac{n}{4}(1 - \epsilon)$. If $|\mathsf{Role}_{\mathsf{MCR}}| = 0$ and Phase 1 initializes without error, then with probability 1, $\frac{1}{3}n \leq |\mathcal{M}| \leq \frac{2}{3}n$, $\frac{1}{6}n \leq |\mathcal{R}| \leq \frac{2}{3}n$, and $2 \leq |\mathcal{C}| \leq \frac{1}{3}n$.*

Proof. The top level of splitting of $\mathsf{Role}_{\mathsf{MCR}}$ into $\mathsf{Role}_{\mathsf{CR}}$ and Main is equivalent to the reactions of theorem 3.6.4, with $U = \mathsf{Role}_{\mathsf{MCR}}$, $M = \mathsf{Main}$, $S = \mathsf{Role}_{\mathsf{CR}}$, and the f and t subscripts representing the Boolean value of the field `assigned`. theorem 3.6.4 gives the stated bounds on Main, if there were no further splitting of $\mathsf{Role}_{\mathsf{CR}}$.

theorem 3.6.4 gives that with high probability, all $\mathsf{Role}_{\mathsf{MCR}}$ are converted to $\mathsf{Role}_{\mathsf{CR}}$ and Main in $12.5 \ln n$ time; we begin the analysis at that point, letting s be the number of $\mathsf{Role}_{\mathsf{CR}}$ agents

produced, noting $n/3 \le s \le 2n/3$ with probability 1, and for any ϵ' $\frac{n}{2}(1-\epsilon') \le s \le \frac{n}{2}(1+\epsilon')$ with high probability.

The splitting of $\mathsf{Role}_{\mathsf{CR}}$ into Clock and Reserve follows a simpler process that we analyze here. Let $r = |\mathcal{R}|$ and $c = |\mathcal{C}|$ at the end of Phase 0. To see the high probability bounds on r and c, we model the splitting of $\mathsf{Role}_{\mathsf{CR}}$ by the reaction $U, U \to R, C$ during Phase 0 and $U \to R$ at the end of Phase 0, since all un-split $\mathsf{Role}_{\mathsf{CR}}$ agents become Reserve agents upon leaving Phase 0.

The reaction $U, U \to R, C$ reduces the count of U from its initial value s ($= n/2 \pm \epsilon' n/2$ WHP) to $\epsilon' s$, with the number of interactions between each reaction when $\#U = l$ governed by a geometric random variable with success probability $O(l^2/n^2)$. Applying theorem 1.3.3 with $k = s - \epsilon' s$, $p_i = O((i + \epsilon' s)^2/n^2)$ for $i \in \{1, \ldots, k\}$, the reaction $U, U \to R, C$ takes $O(1)$ time to reduce the count of U from its initial value m ($= n/2 \pm \epsilon' n/2$ WHP) to $\epsilon' m$ with very high probability. This implies that after $O(1)$ time, r and c are both at least

$$\begin{aligned} s/2 - \epsilon' s = s(1/2 - \epsilon') &\ge (n/2 - \epsilon' n/2)(1/2 - \epsilon') \\ &= n/4 - \epsilon' n/2 - \epsilon' n/4 + (\epsilon')^2 n/2 \\ &> n/4 - \epsilon' n = n/4(1 - 4\epsilon') \end{aligned}$$

with very high probability. Choosing $\epsilon = \epsilon'/4$ gives the high probability bounds on r and c. Thus we require $12.5 \ln n + O(1) \le 13 \ln n$ time, and for appropriate choice of `counter` constant c_0, this happens before the first Clock agent advances to the next phase with high probability.

We now argue the probability-1 bounds. The bound $r \le 2n/3$ follows from $|\mathcal{M}| \ge n/3$. The bound $r \ge n/6$ follows from $|\mathcal{M}| \le 2n/3$, so $\mathsf{Role}_{\mathsf{CR}} \ge n/3$ if no $U, U \to R, C$ splits happen, and the fact that at least half of $\mathsf{Role}_{\mathsf{CR}}$ get converted to Reserve: exactly half by $U, U \to R, C$ and the rest by $U \to R$.

Although the reactions $U, U \to R, C$ and $U \to R$ can produce only a single C, there must be at least two Clock agents for Standard Counter Subroutine to count at all and end Phase 0, so if Phase 1 initializes, $c \ge 2$. The bound $c \le n/3$ follows from the fact that c is maximized when $|\mathcal{M}| = n/3$ and no $U \to R$ reactions happen, i.e., all $2n/3$ $\mathsf{Role}_{\mathsf{CR}}$ agents are converted via $U, U \to R, C$, leading to $c = n/3$. □

We now reason about Phase 1, which has different behavior based on the initial gap g.

LEMMA 3.6.6. *If the initial gap $|g| \geq 0.025|\mathcal{M}|$, then we stabilize to the correct output in Phase 2. If $|g| < 0.025|\mathcal{M}|$, then at the end of Phase 1, all agents have* `bias` $\in \{-1, 0, +1\}$*, and the total count of biased agents is at most $0.03|\mathcal{M}|$. Both happen with high probability $1 - O(1/n^2)$.*

PROOF. In the case where all agents enter Phase 1, none are still in $\mathsf{Role}_{\mathsf{MCR}}$, so every non-Main agent has given their `bias` to a Main agent via line 2 or line 5 of Phase 0. Thus the initial gap $g = \sum_{m.\texttt{role}=\mathsf{Main}} m.\texttt{bias}$.

Let $\mu = \lfloor \frac{g}{|\mathcal{M}|} \rceil$ be the average `bias` among all Main agents, rounded to the nearest integer. By [83], we will converge to have all `bias` $\in \{\mu - 1, \mu, \mu + 1\}$, in $O(\log n)$ time with high probability $1 - O(1/n^2)$. We use Corollary 1 of [83], where the constant $K = O(\sqrt{n})$ and $\delta = \frac{1}{n^2}$. This gives that with probability $1 - \delta$, all `bias` $\in \{\mu - 1, \mu, \mu + 1\}$ after a number of interactions

$$t \geq (n-1)(2\ln\big(K + \sqrt{n}\big) - \ln(\delta) - \ln(2)) \sim n(2\ln\big(\sqrt{n}\big) + \ln\big(n^2\big)) = 3n \ln n.$$

Thus after time $3 \ln n$, all `bias` $\in \{\mu - 1, \mu, \mu + 1\}$ with high probability $1 - 1/n^2$.

If $|g| > 0.5|\mathcal{M}|$, then $|\mu| \geq 1$, so all remaining biased agents have the majority opinion, and we will stabilize in Phase 2 to the correct majority output.

If $|g| \leq 0.5|\mathcal{M}|$, then $\mu = 0$, so now all `bias` $\in \{-1, 0, +1\}$. We will use theorem 3.6.2, with the sets of biased agents $A = \{a : a.\texttt{bias} = +1\}$ and $B = \{b : b.\texttt{bias} = -1\}$, which have initial sizes $|A| = a \cdot n$ and $|B| = b \cdot n$.

In the first case where $0.025|\mathcal{M}| \leq |g| \leq 0.5|\mathcal{M}|$, we have $a - b \geq 0.025m$ (assuming WLOG that A is the majority). Then by theorem 3.6.2, with high probability $1 - O(1/n^2)$, the count of B becomes 0 in at most time $\frac{5 \ln n}{2(a-b)} = \ln n \frac{5}{2 \cdot 0.025m} = \frac{100}{m} \ln n \leq 201 \ln n$. With all minority agents eliminated, we will again stabilize in Phase 2 with the correct output.

In the second case where $|g| < 0.025|\mathcal{M}|$, we can use theorem 3.6.2 with constant $d = b - 0.0025m$. Then even with maximal gap $a - b = 0.025m$, with very high probability in constant time we bring the counts down to $b = 0.0025m$ and $a = 0.0275m$. Thus the total count of biased agents is at most $(0.0025m + 0.0275m)n = 0.03|\mathcal{M}|$.

Since all the above arguments take at most $O(\log n)$ time, for appropriate choice of counter constant c_1, the given behavior happens before the first Clock agent advances to the next phase with high probability. □

3.6.3. Analysis of main averaging Phase 3. The longest part in the proof is analyzing the behavior of the main averaging Phase 3. The results of this section culminate in the following two theorems, one for the case of an initial tie, and the other for an initial biased distribution.

In the case of an initial tie, we will show that all biased agents have minimal $\texttt{exponent} = -L$ by the end of the phase.

THEOREM 3.6.7. *If the initial configuration was a tie with gap 0, then by the end of Phase 3, all biased agents have* $\texttt{exponent} = -L$*, with high probability* $1 - O(1/n^2)$*.*

Note that the where all biased agents have $\texttt{exponent} = -L$ gives a stable configuration in the next Phase 4, with all agents having $\texttt{output} = \mathsf{T}$. Thus from theorem 3.6.7, we conclude that from an intial tie, the protocol will stabilize in Phase 4 with high probability. Conversely, we have already observed that this configuration can only be reached in the case of a tie, since the sum of all biased agents would bound the magnitude of the initial gap $|g| < 1$. Thus in the other case of a majority initial distribution, the agents will proceed through to Phase 5 with probability 1.

In this majority case, we will show that a large majority of the Main agents have $\texttt{opinion}$ set to the majority opinion and $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$ is in a consecutive range of 3 possible exponents, where the value $-l$ depends on the initial distribution. In addition, we show the upper tail above this exponent $-l$ is very small.

THEOREM 3.6.8. *Assume the initial gap* $|g| < 0.025|\mathcal{M}|$*. Let the exponent* $-l = \lfloor \log_2(\frac{g}{0.4|\mathcal{M}|}) \rfloor$*. Let* $i = sign(g)$ *be the majority opinion and* M *be the set of all agents with* $\texttt{role} = \mathsf{Main}$*,* $\texttt{opinion} = i$*,* $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$*. Then at the end of Phase 3,* $|M| \geq 0.92|\mathcal{M}|$ *with high probability* $1 - O(1/n^2)$*.*

In addition, the total mass above exponent $-l$ *is* $\mu_{(>-l)} = \sum_{a.\texttt{exponent} > -l} |a.\texttt{bias}| \leq 0.002|\mathcal{M}|2^{-l}$*, and the total minority mass is* $\beta_{-} = \sum_{a.\texttt{opinion} = -i} |a.\texttt{bias}| \leq 0.004|\mathcal{M}|2^{-l}$*.*

Note the assumption of small initial gap g that we get from Phase 1 is just for convenience in the proof, to reason uniformly about the base case behavior at hour $h = 0$ for our inductive argument. The rules of Phase 3 would also work as intended with even larger initial gaps, just requiring a variant of the later analysis to acknowledge that the initial $\texttt{exponent} = 0$ is quite close to the final range $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$.

3.6.4. Clock synchronization theorems. We will first consider the behavior of the Clock agents in Phase 3. The goal is to show their `minute` fields remain tightly concentrated while moving from 0 up to kL, summarized in theorem 3.6.15. See fig. 3.9 for simulations of the clock distribution. The back tail behind the peak of the `minute` distribution decays exponentially, since each agent is brought ahead by epidemic at a constant rate and thus their counts each decay exponentially. The front part of the distribution decays much more rapidly. With a fraction f of agents at $\texttt{minute} = i$, the rate of the drip reaction is proportional to pf^2. This repeated squaring leads to the concentrations at the leading minutes decaying doubly exponentially. The rapid decay is key to showing that very few Clock agents can get very far ahead of the rest.

While our algorithm runs for only $O(\log n)$ minutes, the clock behavior we require can be made to continue for $O(n^c)$ minutes for arbitrary c. Our proofs rely on induction on minutes, and all results in this section hold with very high probability. Thus, we can take a union bound over polynomially many minutes and still keep the very high probability guarantees. If the clock runs for a superpolynomial number of minutes, the results of this section no longer hold.

We start by consider an entire population running the clock transitions (lines 1-5 of Phase 3), so $|\mathcal{C}| = n$. The following lemmas describe the behavior of just this clock protocol. In our actual protocol, the clock agents are a subpopulation $|\mathcal{C}| = c \cdot n$, and these clock transitions only happen when two clock agents meet with probability $\binom{|\mathcal{C}|}{2}/\binom{n}{2} \sim \frac{1}{c^2}$. This more general situation is handled in later theorems applying to our exact protocol.

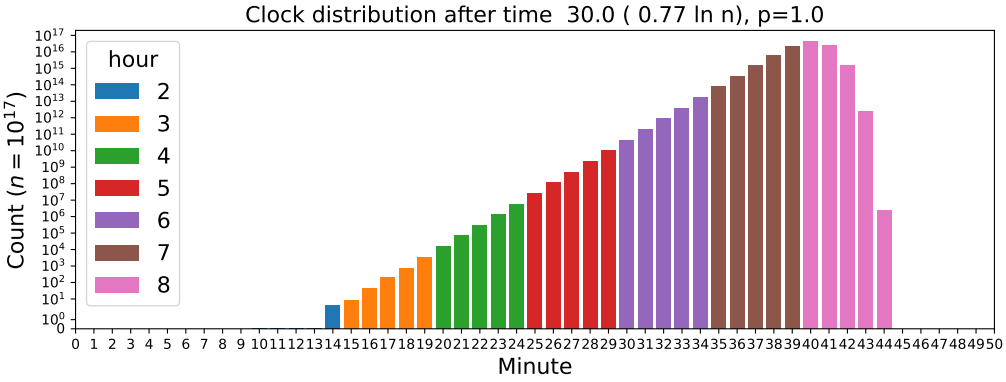
Definitions of values used in subsequent lemma statements. Throughout this section, we reference the following quantities. For each minute i and parallel time t, define $c_{\geq i}(t) = |\{c : c.\texttt{minute} \geq i\}|/|\mathcal{C}|$ to be the fraction of clock agents at minute i or beyond at time t. Then define $t^{+}_{\geq i} = \min\{t : c_{\geq i}(t) > 0\}$, $t^{0.1}_{\geq i} = \min\{t : c_{\geq i}(t) \geq 0.1\}$ and $t^{0.9}_{\geq i} = \min\{t : c_{\geq i}(t) \geq 0.9\}$ to be the first times where this fraction becomes positive, hits 0.1 and hits 0.9.

We first show that the $c_{\geq i+1}$ is significantly smaller than $c_{\geq i}$ while both are still increasing, so the front of the clock distribution decays very rapidly. In our argument, we consider three types of reactions that change the counts at minutes i or above:

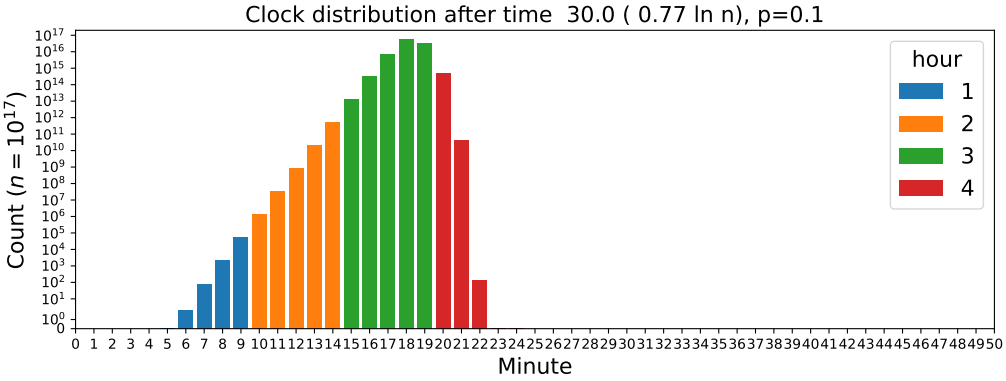
(i) Drip reactions $i, i \to i, (i+1)$ (line 5 in Phase 3)

(ii) Epidemic reactions $j, k \to j, j$, for any minutes j, k with $k \leq i < j$ (line 3 in Phase 3)

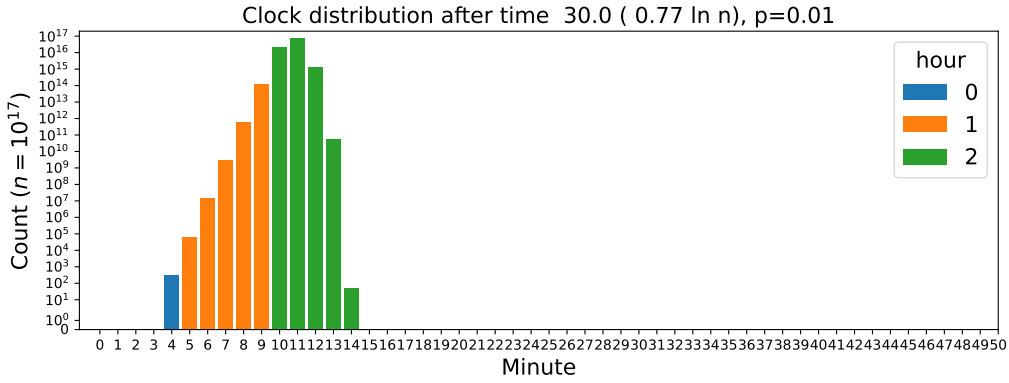
(iii) Epidemic reactions $j, k \to j, j$, for any minutes j, k with $k < i \leq j$ (line 3 in Phase 3)



(a) Simulating the clock rules with $p = 1$.



(b) Simulating the clock rules with $p = \frac{1}{10}$.



(c) Simulating the clock rules with $p = \frac{1}{100}$.

FIGURE 3.9. Simulating the clock rules on a large population of size $n = 10^{17}$, with $k = 5$ minutes per hour and multiple values of p. The `minute` distribution gets tighter for smaller values of p. To make the `hour` distribution tighter, we can also simply make k larger. The full evolution of these distributions can be seen in the example notebook [41], along with the code that ran the large simulation to generate the data.

Note we ignore the drip reactions $(i-1),(i-1) \to (i-1), i$, which would only help the argument, but we do not have guarantees on the count at minute $i-1$.

We will try to show the relationship pictured in fig. 3.9, that $c_{\geq(i+1)}(t) \approx c_{\geq i}(t)^2$. One challenge here is that this relationship will no longer hold with high probability for small values of $c_{\geq i}$. For example, if $c_{\geq i}(t) = n^{-1/2-\epsilon}$, the desired relationship would require $c_{\geq i+1}(t) < \frac{1}{n}$, meaning the count above minute i is 0. A drip reaction at minute i could happen with non-negligible probability $\approx n^{-2\epsilon}$.

To handle this difficulty, we define the set of *early drip agents* $D_{\geq i+1}(t)$, the set of agents that moved above minute i via a drip reaction at a time $\leq t$ when $c_{\geq i}(t) < n^{-0.45}$, or that were brought above minute i via an epidemic reaction with another early drip agent in $d_{\geq i+1}$. Note that the latter group of agents can move to minute $\geq i+1$ *after* time $t^+_{\geq i+1}$. Thus, this set represents the effect of any drip reactions that happen before $c_{\geq i}(t)$ grows large enough for our large deviation bounds to work. We then define $d_{\geq i+1}(t) = |D_{\geq i+1}(t)|/|\mathcal{C}|$ to be the fraction of early drip agents (including effects that persist beyond this time due to epidemic reactions). The set of agents above minute i that comprise the fraction $c_{\geq(i+1)}$ is then partitioned into the early drip agents that comprise $D_{\geq i+1}$, and the rest. By first ignoring these early drip agents $D_{\geq i+1}$, we can show that the rest of $c_{\geq(i+1)}$ stays small compared to $c_{\geq i}$.

LEMMA 3.6.9. *With very high probability, if* $n^{-0.45} \leq c_{\geq i}(t) \leq 0.1$, *then* $c_{\geq i+1}(t) \leq 0.9pc_{\geq i}(t)^2 + d_{\geq i+1}(t)$.

PROOF. The proof will proceed by induction on time t. As a base case, for all t such that $c_{\geq i}(t) < n^{-0.45}$, the statement holds simply by definition of $d_{\geq i+1}(t)$, which is equal to $c_{\geq i+1}(t)$.

For the inductive step, to show the relationship $c_{\geq(i+1)}(t) < 0.9pc_{\geq i}(t)^2 + d_{\geq i+1}(t)$ holds at time t, we will use the inductive hypothesis that $c_{\geq(i+1)}(t-0.1) < 0.9pc_{\geq i}(t-0.1)^2 + d_{\geq i+1}(t-0.1)$. Let $x(t) = c_{\geq i}(t)$ and $y(t) = c_{\geq i+1}(t) - d_{\geq i+1}(t)$, so we need to show $y(t) < 0.9px(t)^2$.

We first lower bound how much $x(t)$ grows by epidemic reactions, in order to show $x(t-0.1) < 0.84x(t)$. Using theorem 1.3.5, the expected amount of time for an epidemic to grow from fraction $0.84x$ to x is

$$\frac{1}{2}\left[\ln(x) - \ln(0.84x) + \ln\left(\frac{1-0.84x}{1-x}\right)\right] \leq \frac{1}{2}\left[-\ln(0.84) + \ln\left(\frac{1-0.84\cdot 0.1}{1-0.1}\right)\right] < 0.096,$$

where we used the fact that $\frac{1-0.84x}{1-x}$ is nondecreasing and $x(t) \leq 0.1$.

The minimum probability p^* is $\Theta(x(t)) = \Omega(n^{-0.45})$, and the expected number of interactions μ is $\Theta(n)$. So the application of theorem 1.3.2 will give probability $1 - e^{-\Omega(n^{0.55})}$ for the epidemic to have grown enough within time 0.1. Thus $x(t-0.1) < 0.84x(t)$ with very high probability.

Next we bound how much $y(t)$ grows, by both epidemic reactions and drip reactions. The probability of a drip reaction is at most $px(t)^2$, so the expected number of drip reactions in time 0.1 is $0.1px(t)^2$. A standard Chernoff bound then gives that there are at most $0.11px(t)^2$ drip reactions with very high probability. We assume in the worst case all these drip reactions happen at time $t - 0.1$, and then y grows by epidemic starting from $z = y(t-0.1) + 0.11px(t)^2$.

By theorem 1.3.5, the expected amount of time for an epidemic to grow from fraction z to $1.23z$ is

$$\frac{1}{2}\left[\ln(1.23z) - \ln(z) + \ln\left(\frac{1-z}{1-1.23z}\right)\right] \geq \frac{1}{2}\ln(1.23) > 0.103.$$

The minimum probability $p^* = \Omega(x(t)^2) = \Omega(n^{-0.9})$, and the expected number of interactions $\mu = \Theta(n)$. So the application of theorem 1.3.2 will give probability $1 - e^{-\Omega(n^{0.1})}$. Thus with very high probability

$$\begin{aligned} y(t) &\leq 1.23[y(t-0.1) + 0.11px(t)^2] \\ &\leq 1.23[0.9px(t-0.1)^2 + 0.11px(t)^2] \\ &\leq 1.23[0.9p(0.84x(t))^2 + 0.11px(t)^2] < 0.9px(t)^2. \end{aligned}$$ □

In order to bound $c_{\geq(i+1)}(t)$ as a function of $c_{\geq i}(t)$ only (theorem 3.6.11), we need to bound how large the set $D_{\geq i+1}$ of early drip agents can be. The strategy will be to show there is not enough time for the set $D_{\geq i+1}$ to grow very large starting from $t^+_{\geq i+1}$, the first time any agent appears at minute $\geq i+1$ (i.e., the first drip reaction into minute $i+1$), because there are only $O(\log\log n)$ minutes in the front tail (see fig. 3.9), so the time between $t^+_{\geq i+1}$ and $t^{0.1}_{\geq i+1}$ is $O(\log\log n)$.

We will first need an upper bound on how long it takes the clock to move from one minute to the next.

LEMMA 3.6.10. $t^{0.1}_{\geq i+1} - t^{0.1}_{\geq i} \leq 2.11 + \frac{1}{2}\ln\left(\frac{1}{p}\right)$ *with very high probability.*

PROOF. First we argue that $c_{\geq i+1}(t_i^{0.1} + \frac{1}{2}) > 0.0045p$. If not, the count at minute i, $c_{\geq i}(t) - c_{\geq i+1}(t) > 0.1 - 0.0045 = 0.0955$ for all $t_i^{0.1} < t < t_i^{0.1} + 0.5$. Then, the probability of a drip reaction is at least $0.0955^2 p > 0.0091p$. By standard Chernoff bounds, we then have that in time $\frac{1}{2}$, there are at least $0.0045p$ drip reactions with very high probability. Thus $c_{\geq i+1}(t_i^{0.1} + \frac{1}{2}) > 0.0045p$ from just those drip reactions alone.

Now we argue that the amount of time it takes for epidemic reactions to bring $c_{\geq(i+1)}$ up to 0.1. By theorem 1.3.5, the expected amount of time for an epidemic to grow from fraction $0.0045p$ to 0.1 is

$$\begin{aligned}\frac{1}{2}[\ln(0.1) - \ln(0.0045p) + \ln(1 - 0.0045p) - \ln(0.9)] &< \frac{\ln(0.1) - \ln(0.0045) - \ln(0.9)}{2} - \frac{\ln p}{2} \\ &< 1.603 - \frac{\ln p}{2}.\end{aligned}$$

As long as $p = \Theta(1)$, then the minimum probability $p^* = \Theta(p)$ is constant, and by theorem 1.3.5, the epidemic takes at most time $1.61 - \frac{\ln p}{2}$ with very high probability.

In total, we then get that $t_{i+1}^{0.1} - t_i^{0.1} \leq 0.5 + 1.61 - \frac{\ln p}{2} = 2.11 + \frac{1}{2}\ln\left(\frac{1}{p}\right)$ with very high probability. □

Proving that the set $D_{\geq i+1}$ remains small will let us prove the main theorem about the front tail of the clock distribution:

THEOREM 3.6.11. *With very high probability, if* $n^{-0.4} \leq c_{\geq i}(t) \leq 0.1$, *then* $c_{\geq(i+1)}(t) < pc_{\geq i}(t)^2$.

PROOF. The proof will proceed by induction on the minute i, where the base case is vacuous because $c_{\geq 0}(0) = 1$, so $c_{\geq 0}(t) > 0.1$ for all times t.

The inductive hypothesis will use two claims. The first is that the time $t_{\geq i}^{0.1} - t_{\geq i}^{+} = O(\log\log n)$ for minute i. The second is that $d_{\geq i+1}(t_{\geq i}^{0.1}) = O(n^{-0.85})$. Note that using this second claim along with theorem 3.6.9 proves the Theorem statement at minute i: when $n^{-0.4} \leq c_{\geq i}(t) \leq 0.1$, by theorem 3.6.9 we have

$$c_{\geq(i+1)}(t) < 0.9pc_{\geq i}(t)^2 + d_{\geq i+1}(t) \leq pc_{\geq i}(t)^2,$$

because $c_{\geq i}(t)^2 \geq n^{-0.8}$, so the $d_{\geq i+1}$ term is negligible for sufficiently large n.

We will prove the first claim in two parts. First we argue that $t_{\geq i}^{0.1} - t_{\geq i}^{+} = O(\log\log n)$, because the width of the front tail is at most $2\log\log n$. We will show that at $t_{\geq i}^{+}$, when the first agent

arrives at $\texttt{minute} = i$, we already have $c_{\geq j}(t^+_{\geq i}) \geq 0.1$, where $j = i - 2\log\log n$ (for $i < 2\log\log n$, we just have $j = 0$ and there is nothing to show because the width of the front tail can be at most i). First we move $\log\log n$ levels back to $k = i - \log\log n$, to show $c_{\geq k}(t^+_{\geq i}) \geq n^{-0.4}$.

Assume, for the sake of contradiction that $c_{\geq k}(t^+_{\geq i}) < n^{-0.4}$. Between $t^+_{\geq k}$ and $t^{n^{-0.4}}_{\geq k}$, consider the number of drips that happen from levels $k+1$ and above. By the inductive hypothesis, we have $c_{\geq k+1}(t) \leq pc_{\geq k}(t)^2 < n^{-0.8}$ during this whole time. Thus in any interaction of this period the probability of a drip above level $k+1$ is at most $p \cdot (n^{-0.8})^2 \leq n^{-1.6}$. The interval length is $t^{n^{-0.4}}_{\geq k} - t^+_{\geq k} = O(\log\log n)$ by the inductive hypothesis, so the probability of having at least $\log\log n$ drips during this interval is at most

$$\binom{O(n\log\log n)}{\log\log n}(n^{-1.6})^{\log\log n} \leq \left(\frac{O(n\log\log n)}{n^{1.6}}\right)^{\log\log n} = n^{-\omega(1)}.$$

This implies $c_{\geq i}(t^{n^{-0.4}}_{\geq k}) = 0$ with very high probability.

Thus with very high probability, we already have $c_{\geq k}(t^+_{\geq i}) \geq n^{-0.4}$. Then we can iterate the inductive hypothesis $c_l(t) \leq p(c_{l-1}(t))^2$ for the $\log\log n$ minutes $l = k, k-1, \ldots, j$, which implies $c_{\geq j}(t^+_{\geq i}) \geq 0.1$. Now that we have shown the width of the front tail is at most $2\log\log n$, we use theorem 3.6.10, which shows that each minute takes $O(1)$ time, so it takes $O(\log\log n)$ time between $t^+_{\geq i}$ when the first agent gets to $\texttt{minute} = i$ and $t^{0.1}_{\geq i}$, when the fraction at $\texttt{minute} \geq i$ reaches 0.1.

Now we prove the second claim, arguing that in this $O(\log\log n)$ time, $d_{\geq i+1}$ can grow to at most $O(n^{-0.85})$. By definition of $d_{\geq i+1}$, the drip reactions that increase $d_{\geq i+1}$ happen with probability at most $p(n^{-0.45})^2 = pn^{-0.9}$. By a standard Chernoff bound, the number of drip reactions in $O(\log\log n)$ time is $O(\log\log n \cdot n^{-0.9}) = O(n^{-0.89})$. Then we assume in the worst case this maximal number of drip reactions happen, and then $d_{\geq i+1}$ can grow by epidemic. By theorem 1.3.5, the time for an epidemic to grow from fraction $O(n^{-0.89})$ to $\Omega(n^{-0.85})$ is $\Omega(\log n) > O(\log\log n)$ with very high probability. Thus, with very high probability, we still have $d_{\geq i+1} = O(n^{-0.85})$ within $O(\log\log n)$ time. □

Now the relationship proven in theorem 3.6.11 implies that $c_{\geq (i+1)}(t^{0.1}_{\geq i}) \leq 0.01p$. We can now use this bound to get lower bounds on the time required to move from one minute to the next.

LEMMA 3.6.12. *With very high probability,* $t^{0.1}_{i+1} - t^{0.1}_i \geq \frac{1}{2}\ln\left(1 + \frac{2}{9p}\right) - 0.01$.

PROOF. We start at time $t^{0.1}_{\geq i}$, where by theorem 3.6.11 we have $c_{\geq i+1}(t^{0.1}_{\geq i}) \leq 0.01p$ with very high probability.

Define $x = x(t) = c_{\geq i}(t)$ and $y = y(t) = c_{\geq i+1}(t)$. The number of interactions for $Y = yn$ to increase by 1 is a geometric random variable with mean $\frac{1}{\Pr[\text{(i)}]+\Pr[\text{(ii)}]}$, where the drip reaction (i) has probability $\Pr[\text{(i)}] \sim p(x-y)^2 \leq p(1-y)^2$ and the epidemic reaction (ii) has probability $\Pr[\text{(ii)}] \sim 2y(1-y)$. Assuming in the worst case that $y(t^{0.1}_i) = 0.01p$, then the number of interactions $T = (t^{0.1}_{i+1} - t^{0.1}_i)n$ for Y to increase from $0.1pn$ to $0.1n$ is a sum of independent geometric random variables with mean

$$\begin{aligned}
\mathbb{E}[T] &\geq \sum_{Y=0.01pn}^{0.1n-1} \frac{1}{p(1-Y/n)^2 + 2(Y/n)(1-Y/n)} \sim n\int_{0.01p}^{0.1} \frac{dy}{p(1-y)^2 + 2y(1-y)} \\
&= n\int_{0.01p}^{0.1} \frac{dy}{(1-y)(p+(2-p)y)} = n\int_{0.01p}^{0.1} \frac{1/2}{1-y} + \frac{1-p/2}{p+(2-p)y} dy \\
&= n\left[-\frac{1}{2}\ln(1-y) + \frac{1}{2}\ln(p+(2-p)y)\right]_{0.01p}^{0.1} \\
&= \frac{n}{2}[-\ln(0.9) + \ln(1-0.01p) + \ln(p+0.1(2-p)) - \ln(p+0.01p(2-p))] \\
&= \frac{n}{2}\left[-\ln(0.9) + \ln\left(\frac{1-0.01p}{p-0.01p^2+0.02p}\right) + \ln(0.9p+0.2)\right] \\
&\geq \frac{n}{2}\left[-\ln(0.9) + \ln\left(\frac{0.9p+0.2}{1.02p}\right)\right] \geq \frac{n}{2}\left[-0.02 + \ln\left(1+\frac{2.04}{9p}\right)\right]
\end{aligned}$$

Note that the probability in the geometric includes the term $p(1-y)^2 \geq 0.81p$ since $y \leq 0.1$, thus the minimum geometric probability $p^* \geq 0.81p$ is bounded by a constant. Also the mean $\mu = \Theta(n)$ so by theorem 1.3.3, $\Pr\left[T \leq n(-0.01 + \frac{1}{2}\ln\left(1+\frac{2}{9p}\right)\right] \leq \exp(-\Theta(n))$, so the time $t^{0.1}_{i+1} - t^{0.1}_i = T/n \geq \frac{1}{2}\ln\left(1+\frac{2}{9p}\right) - 0.01$ with very high probability. □

The worst case upper bound for the dripping probability, $p(1-y)^2$, used in the above Lemma, is weakest when $p = 1$. We now give a special case lower bound that is stronger in the deterministic case with $p = 1$.

LEMMA 3.6.13. *With very high probability,* $t^{0.1}_{i+1} - t^{0.1}_i \geq 0.45$.

PROOF. We assume in the worst case that $p = 1$. Then by theorem 3.6.11, we have $c_{\geq i+1}(t^{0.1}_i) \leq 0.01$. This initial fraction will grow by epidemic to at most $0.01 \cdot e^{2\cdot 0.45}(1+\epsilon)$ with very high

probability by theorem 1.3.5. We must also consider all agents that later make it to minute $i+1$ by a drip reaction, and how much they grow by epidemic.

By time $t_i^{0.1} + s$, the fraction $c_{\geq i}$ could have increased to at most $0.1 + s$, since at most 1 agent can increase its minute in each interaction. Then the probability of a drip reaction at this time is at most $(0.1+s)^2$. By standard Chernoff bounds, there will be at most $(1+\epsilon)(0.1+s)^2 n^{0.5}$ drip reactions in the next $n^{0.5}$ interactions with very high probability. Then by theorem 1.3.5, these agents can grow by epidemic by at most a factor $(1+\epsilon)e^{2\cdot(0.45-s)}$ by time $t_i^{0.1} + 0.45$, with very high probability. Summing over consecutive groups of $n^{0.5}$ interactions at time $i \cdot \frac{n^{0.5}}{n}$ and using the union bound, we get a total bound

$$c_{\geq i+1}(t_i^{0.1} + 0.45) \leq 0.01 \cdot e^{2\cdot 0.45}(1+\epsilon) + \sum_{i=0}^{0.45n^{0.5}} (1+\epsilon)(0.1 + n^{-0.5}i)^2 n^{0.5} \cdot e^{2\cdot(0.45 - n^{-0.5}i)}$$
$$\sim (1+\epsilon)\cdot e^{0.9}\left[0.01 + \int_0^{0.45} (1+s)^2 e^{-2s} ds\right] \leq 0.45.$$

□

We now summarize all bounds on the length of a clock minute in a single theorem:

THEOREM 3.6.14. *Let $t^{0.1}_{\geq i+1} - t^{0.1}_{\geq i}$ be the time between when a fraction 0.1 of agents have* `minute` $\geq i$ *and when a fraction 0.1 of agents have* `minute` $\geq i+1$. *Then with very high probability,*

$$\max\left(0.45, \frac{1}{2}\ln\left(1 + \frac{2}{9p}\right) - 0.01\right) \leq t^{0.1}_{\geq i+1} - t^{0.1}_{\geq i} \leq 2.11 + \frac{1}{2}\ln\left(\frac{1}{p}\right)$$

These bounds are shown in fig. 3.10, along with sampled minute times from simulation. This suggests the actual time per minute is roughly $0.75 + \frac{1}{2}\ln\left(\frac{1}{p}\right)$.

We can now build from these theorems to get bounds on the values of `hour`. For a clock agent a, define $a.\texttt{hour} = \lfloor \frac{a.\texttt{minute}}{k} \rfloor$. Define $\text{start}_h = \min\left(t : |\{a : a(t).\texttt{hour} \geq h\}| \geq 0.9|\mathcal{C}|\right)$ be the first time when the fraction of clock agents at hour h or beyond reaches 0.9 and $\text{end}_h = \min\left(t : |\{a : a(t).\texttt{hour} > h\}| \leq 0.001|\mathcal{C}|\right)$ be the first time when the fraction of clock agents beyond hour h reaches 0.001. Define the *synchronous hour* h to be the parallel time interval $[\text{start}_h, \text{end}_h]$, i.e. when fewer than 0.1% are in any hour beyond h and at least (90 - 0.1) = 89.9% are in hour h. Note that if $\text{end}_h < \text{start}_h$ then this interval is empty, but we show this happens with low probability. We choose the threshold $0.001|\mathcal{C}|$ to be a sufficiently small constant for later proofs.

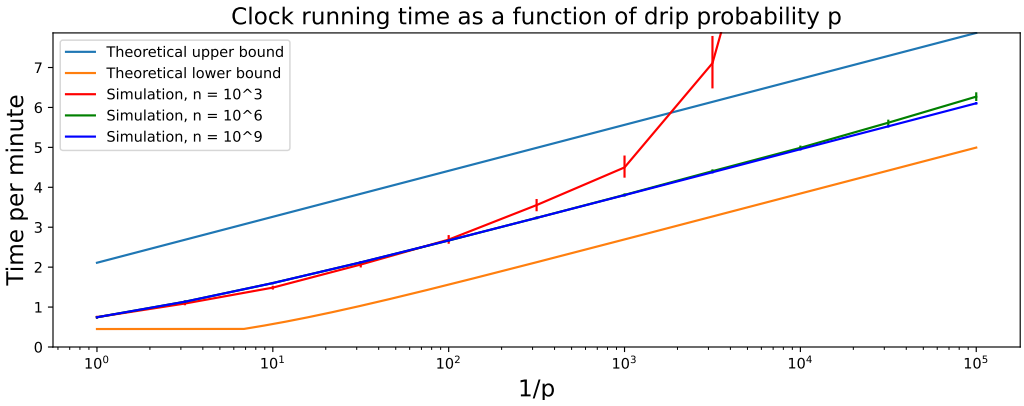


FIGURE 3.10. The upper and lower bounds from theorem 3.6.14 for the time of one clock minute, along with samples from simulation. For each value of n, 100 minute times were sampled, taking $t_{i+1}^{0.1} - t_i^{0.1}$ for $i = 9, \ldots, 18$ over 10 independent trials. All our proofs assume p is constant, and for any fixed value of p, will only hold for sufficiently large n. The case $n = 10^3$ shows that when $p = O(1/n)$, the bounds no longer hold. This is to be expected because the expected number of drips becomes too small for large deviation bounds to still hold.

Recall $c = |\mathcal{C}|/n$ is the fraction of clock agents and k is the number of minutes per hour.

THEOREM 3.6.15. *Consider a fraction c of agents running the clock protocol, with $p = 1$. Then for every synchronous hour h, its length* $\text{end}_h - \text{start}_h \geq \frac{1}{c^2}[0.45k - 3.1]$, *with very high probability. The time between consecutive synchronous hours* $\text{start}_{h+1} - \text{start}_h \leq \frac{1}{c^2}[2.11k + 2.2]$ *with very high probability.*

PROOF. Note that the previous lemmas assumed $|\mathcal{C}| = n$, so the entire population was running the clock algorithm. In reality, we have a fraction $c = |\mathcal{C}|/n$ of clock agents. The reactions we considered only happen between two clock agents, which interact with probability $\sim c^2$. Thus we can simply multiply the bounds from our lemmas by the factor $\frac{1}{c^2}$ to account for the number of regular interactions for the requisite number of clock interactions to happen, which is very close to its mean with very high probability by standard Chernoff bounds.

Because our definition references time $t_{\geq i}^{0.9}$ when the fraction reaches 0.9, we will first bound the time it takes an epidemic to grow from 0.1 to 0.9. By theorem 1.3.5, this takes parallel time t, where

$$\mathbb{E}[t] \sim \frac{\ln(0.9) - \ln(1 - 0.9) - \ln(0.1) + \ln(1 - 0.1)}{2} = \ln(9) < 2.2.$$

Since $\ln(9)(1 + \epsilon) < 2.2$, this completes within parallel time 2.2 with very high probability.

Since $c.\texttt{hour} = h \iff hk \le c.\texttt{minute} < (h+1)k$, the times $\text{start}_h = t^{0.9}_{hk}$ and $\text{end}_h = t^{0.001}_{hk}$. For the upper bound, by theorem 3.6.10 we have $t^{0.1}_{i+1} - t^{0.1}_i \le 2.11/c^2$ with very high probability. We start at $t^{0.9}_{hk} \ge t^{0.1}_{hk}$, and sum over the k minutes in hour h, then add at most time $\frac{2.2}{c^2}$ between $t^{0.1}_{(h+1)k}$ and $t^{0.1}_{(h+1)k}$. This gives that $\text{start}_{h+1} - \text{start}_h \le \frac{1}{c^2}[2.11k + 2.2]$ with very high probability.

For the lower bound, by theorem 3.6.11, at time $t^{0.1}_{(h+1)k-2}$, we have $c_{\ge(h+1)k} \le (0.1^2)^2 = 10^{-4} < 0.001$, so $t^{0.1}_{(h+1)k-2} < \text{end}_h$. Then $\text{start}_h \le t^{0.1}_{hk} + \frac{2.2}{c^2}$. Using theorem 3.6.13, we have $t^{0.1}_{i+1} - t^{0.1}_i \le 0.45/c^2$ with very high probability, for each $i = hk, \dots, hk + k - 3$. All together, this gives $\text{end}_h - \text{start}_h \ge \frac{1}{c^2}[0.45(k-2) - 2.2] = \frac{1}{c^2}[0.45k - 3.1]$. □

We will set $k = 45$ to give us sufficiently long synchronous hours for later proofs. Since every hour takes constant time with very high probability, all L hours will finish within $O(\log n)$ time.

We finally show one more lemma concerning how the Clock agents affect the Main agents. The Clock agents will change the `hour` of the $\mathcal{O}$ agents via line 9 of Phase 3. There are a small fraction $0.001|\mathcal{C}|$ of Clock agents that might be running too fast and thus have a larger `hour` than the synchronized hour. We must now show these agents are only able to affect a small fraction of the Main agents. Intuitively, the Clock agents with $\texttt{hour} > h$ bring up the hour of both $\mathcal{O}$ agents and other Clock agents. The following lemma will show they don't affect too many Main agents before also bringing in a large number of Clock agents.

We will redefine $c_{>h} = c_{>h}(t) = |\{c : c.\texttt{hour} > h\}|/|\mathcal{C}|$ to be the fraction of Clock agents beyond hour h, and similarly $m_{>h} = m_{>h}(t) = |\{m : m.\texttt{hour} > h\}|/|\mathcal{M}|$ to be the fraction of Main agents beyond hour h.

Lemma 3.6.16. *For all times $t \le \text{end}_h$, we have $m_{>h}(t) \le 1.2c_{>h}(\text{end}_h) = 0.0012$ with very high probability.*

Proof. We have $c_{>h}(\text{end}_h) = 0.001$ by the definition of synchronous hour h. Thus it suffices to show that $m_{>h}(t) \le 1.2c_{>h}(\text{end}_h)$.

Recall $c = |\mathcal{C}|/n$ and $m = |\mathcal{M}|/n$, so $c \cdot c_{>h}$ and $m \cdot m_{>h}$ are rescaled to be fractions of the whole population.

We will assume in the worst case that every Main agent can participate in the clock update reaction

$$C_h, M_j \to C_h, M_h \text{ where } h > j,$$

so the probability of clock update reaction is $2c \cdot c_{>h} \cdot m(1 - m_{>h})$. Among the Clock agents, we will now only consider the epidemic reactions $C_h, C_j \to C_h, C_h$ between agents in different hours $h > j$, so the probability of the clock epidemic reaction is $2c^2 \cdot c_{>h}(1 - c_{>h})$.

We use the potential $\Phi(t) = m_{>h}(t) - 1.1 \cdot c_{>h}(t)$. Note that initially $\Phi(0) = 0$ since both terms are 0. The desired inequality is $\Phi(\text{end}_h) \leq 0.1 c_{>h}(\text{end}_h) = 0.0001$, and we will show this holds with very high probability by Azuma's Inequality because Φ is a supermartingale. The clock update reaction increases Φ by $\frac{1}{|\mathcal{M}|} = \frac{1}{mn}$. The clock epidemic reaction decreases Φ by $\frac{1.1}{|\mathcal{C}|} = \frac{1.1}{cn}$. This gives an expected change

$$\begin{aligned}
\mathbb{E}\left[\Phi(t + 1/n) - \Phi(t)\right] &= \frac{1}{mn}[2c \cdot c_{>h} \cdot m(1 - m_{>h})] - \frac{1.1}{cn}\left[2c^2 \cdot c_{>h}(1 - c_{>h})\right] \\
&= \frac{2c \cdot c_{>h}}{n}[(1 - m_{>h}) - 1.1(1 - c_{>h})] \\
&\leq \frac{2c \cdot c_{\geq i}}{n}[1 - 1.1(0.999)] < 0.
\end{aligned}$$

Thus the sequence $\left(\Phi_j = \Phi(\frac{j}{n})\right)_{j=0}^{n \cdot \text{end}_h}$ is a supermartingale, with bounded differences $|\Phi_{j+1} - \Phi_j| \leq \max\left(\frac{1}{mn}, \frac{r}{cn}\right) = O(\frac{1}{n})$. So we can apply Azuma's Inequality (theorem 3.6.1) to conclude

$$\Pr\left[\Phi_{n \cdot \text{end}_h} \geq \delta\right] \leq \exp\left(-\frac{\delta^2}{2\sum_{j=0}^{n \cdot \text{end}_h} O(\frac{1}{n})^2}\right) = \exp\left(-O(n)\delta^2\right),$$

since by theorem 3.6.15, we have time $\text{end}_h = O(1)$ with very high probability. Thus we can choose $\delta = 0.1 c_{>h}(\text{end}_h) = 0.0001$ to conclude that $m_{>h}(\text{end}_h) \leq 1.2 c_{>h}(\text{end}_h)$ with very high probability $1 - \exp(-\Omega(n)) = 1 - O(n^{-\omega(1)})$.

The lemma statement for times $t \leq \text{end}_h$ simply follows from the monotonicity of the value $m_{>h}$, since agents only decrease their `hour` field. □

3.6.5. Phase 3 exponent dynamics. We now analyze the behavior of the Main agents in Phase 3. We will show their `exponent` fields stay roughly synchronized with the `hour` of the Clock agents, decreasing from 0 toward $-L$. We first use the above results on the clock to conclude that during synchronous hour h, the tail of either $\mathcal{O}$ agents with $\texttt{hour} > h$ or biased agents with $\texttt{exponent} < -h$ is sufficiently small.

LEMMA 3.6.17. *Until time* end_h*, the count* $|\{\mathcal{O} : \mathcal{O}.\texttt{hour} > h\} \cup \{b : b.\texttt{exponent} < -h\}| \leq 0.0024|\mathcal{M}|$ *with very high probability.*

PROOF. By theorem 3.6.16, the count of $\mathcal{O}$ agents that have been pulled up by a Clock agent to $\texttt{hour} > h$ is at most $0.0012|\mathcal{M}|$ with very high probability. Each of these agents could participate in a split reaction that results in two biased agents with $\texttt{exponent} < -h$, increasing the total count of $|\{\mathcal{O} : \mathcal{O}.\texttt{hour} > h\} \cup \{b : b.\texttt{exponent} < -h\}|$ by 1. Thus this total count can be at most twice as large: $\leq 0.0024|\mathcal{M}|$. □

Our main argument will proceed by induction on synchronous hours. During each synchronous hour h, we will first show that at least a constant fraction $0.77|\mathcal{M}|$ of agents have $\texttt{bias} = \mathsf{T}$ with $\texttt{hour} = h$. Then we will show that split reactions bring most biased agents down to $\texttt{exponent} = -h$. Finally we will show that cancel reactions will reduce the count of biased agents, leaving sufficiently many $\mathcal{O}$ agents for the next step of the induction.

Recall that the initial gap

$$g = (|\{a : a.\texttt{input} = \mathsf{A}\}| - |\{b : b.\texttt{input} = \mathsf{B}\}|) = \sum_{m.\texttt{role}=\mathsf{Main}} m.\texttt{bias}$$

is both the difference between the counts of A and B agents in the initial configuration and the invariant value of the total bias. We define $\beta_+ = \beta_+(t) = \sum_{a.\texttt{opinion}=+1} |a.\texttt{bias}|$ and $\beta_- = \beta_-(t) = \sum_{b.\texttt{opinion}=-1} |b.\texttt{bias}|$ to be the total unsigned bias of the positive A agents at time t and negative B agents at time t. Then the net bias $g = \beta_+ - \beta_-$ is the invariant.

We also define $\mu = \mu(t) = \beta_+ + \beta_-$ to be the total unsigned bias, which we interpret as "mass". Note that split reactions preserve μ, while cancel reactions strictly decrease μ. The initial configuration has mass $\mu(0) = n$, and if we eliminate all of the minority opinion then we will get $\beta_- = 0$ and $\mu = g$. Thus decreasing the mass shows progress toward reaching consensus. Also, if every biased agent decreased their exponent by 1, this would exactly cut the μ in half. We will show an upper bound on μ that cuts in half after each synchronous hour, which implies that on average all biased agents are moving down one exponent.

We also define $\mu_{(>-i)}(t) = \sum_{m.\texttt{exponent}>-i} |m.\texttt{bias}|$ as the total mass of all biased agents above exponent $-i$. Note that a bound $\mu_{(>-i)} \leq x2^{-i+1}$ gives a bound on total count $|\{a : a.\texttt{exponent} > -i\}| \leq x$, since even if all agents above exponent $-i+1$ split down to exponent $-i+1$, they would have count at most x. Also note that $\mu(t)$ and $\mu_{(>-i)}(t)$ are nonincreasing in t, since the mass above a given exponent can never increase.

This inductive argument on synchronous hours will stop working once we reach a low enough exponent that the gap has been sufficiently amplified. We define $g_i = g \cdot 2^i$, which we call the relative gap to hour i / exponent $-i$, because if all agents had $\texttt{exponent} = -i$, then a gap $g_i = |\{a : a.\texttt{opinion} = +1\}| - |\{b : b.\texttt{opinion} = -1\}|$ would maintain the invariant $g = \sum_v v.\texttt{bias} = \sum_{a.\texttt{opinion}=+1} \frac{1}{2^i} - \sum_{b.\texttt{opinion}=-1} \frac{1}{2^i} = \frac{g_i}{2^i}$. Note that $g_0 = g$, and the relative gap doubles as we increment the hour and decrement the exponent. So if the Main agents have roughly synchronous exponents, there will be some minimal exponent where the relative gap has grown to exceed the number of Main agents $|\mathcal{M}|$, and there are not enough agents available to maintain the invariant using lower exponents.

We now formalize this idea of a minimal exponent. In the case where there is an initial majority, $g_i \neq 0$, and because we assume the majority is A, we have $g_i > 0$. Define the minimal exponent $-l = \lfloor \log_2(\frac{g}{0.4|\mathcal{M}|}) \rfloor$ to be the unique exponent corresponding to relative gap $0.4|\mathcal{M}| \leq g_l < 0.8|\mathcal{M}|$. For larger exponents $-i \geq -l + 5 = -(l-5)$, we have $g_i \leq g_{l-5} < 0.025|\mathcal{M}|$. Thus for hours $0, 1, \ldots, l-5$ and exponents $0, -1, \ldots, -l+5$, the gap is still very small. The small gap makes the inductive argument stronger, and we will use this small gap to show a high rate of cancelling keeps shrinking the mass μ in half each hour and keeps the count of $\mathcal{O}$ agents large, above a constant fraction $0.77|\mathcal{M}|$.

For the last few hours $l-4, \ldots, l$ and exponents $-l+4, \ldots, -l$, the doubling gap weakens the argument. Thus we have separate bounds for each of these hours. These weaker bounds acknowledge the fact that the majority count is starting to increase while the count of $\mathcal{O}$ agents is starting to decrease.

THEOREM 3.6.18. *With very high probability the following holds. For all synchronous hours $h = 0, \ldots, l$, during times $[\text{start}_h + \frac{2}{c}, \text{end}_h]$, the count $\mathcal{O}_h$ of $\mathcal{O}$ agents at $\texttt{hour} = h$, obeys $|\mathcal{O}_h| \geq \tau_h|\mathcal{M}|$. Then by time $\text{start}_h + \frac{2}{c} + \frac{41}{m}$, the mass $\mu_{(>-h)}$ above exponent $-h$ satisfies $\mu_{(>-h)} \leq 0.001 \cdot 2^{-h+1}$. Then by time $\text{start}_h + \frac{2}{c} + \frac{47}{m} \leq \text{end}_h$, the total mass $\mu(\text{end}_h) \leq \rho_h|\mathcal{M}|2^{-h}$.*

The constant $\rho_h = 0.1$ for $h \leq l-5$. Then we have $\rho_{l-4} = 0.104$, $\rho_{l-3} = 0.13$, $\rho_{l-2} = 0.212$, $\rho_{l-1} = 0.408$, and $\rho_l = 0.808$.

The constant $\tau_h \geq 0.97 - 2\rho_{(h-1)}$, so $\tau_h \geq 0.77$ for $h \leq l-4$, and the minimum value $\tau_l \geq 0.15$.

Note that in the case of a tie, l is undefined since we always have gap $g_i = 0$. Here the stronger inductive argument will hold for all hours and exponents. In the tie case, we will technically define $l = L + 5$ so the stronger $h \le l - 5$ bounds apply to all hours.

We will prove the three sequential claims via three separate lemmas. The first argument of theorem 3.6.19, where the clock brings a large fraction of $\mathcal{O}$ agents up to hour h, will need parallel time $\frac{2}{c}$. The second argument of theorem 3.6.21, where the $\mathcal{O}$ agents reduce the mass above exponent $-h$ by split reactions, will need parallel time $\frac{41}{m}$. The third argument of theorem 3.6.22, where cancel reactions at exponent $-h$ reduce the total mass, will need parallel time $\frac{6}{m}$. Thus the total time we need in a synchronous hour is

$$\frac{2}{c} + \frac{47}{m} \le \frac{49}{c} \le \frac{17}{c^2} \le \frac{0.45 \cdot 45 - 3.1}{c^2},$$

where we use that $c < \frac{1}{3} < m$ by theorem 3.6.5. Thus by theorem 3.6.15, since we have constant $k = 45$ minutes per hour, each synchronous hour is long enough with very high probability.

The argument proceeds by induction, with each lemma using the previous lemmas and the inductive hypotheses at previous hours. The base case comes from theorem 3.6.6, where we have that the initial gap $|g| < 0.025m$, so $l - 5 \ge 0$, and the starting mass is at most $0.03|\mathcal{M}|$. This mass bound gives the base case for theorem 3.6.22, whereas since $h = 0$ is the minimum possible hour, the base cases for theorem 3.6.19 and theorem 3.6.21 are trivial.

LEMMA 3.6.19. *With very high probability the following holds. For each hour $h = 0, \dots, l$, during the whole synchronous hour $[\text{start}_h, \text{end}_h]$, the count of $\mathcal{O}$ agents $|\mathcal{O}| \ge (0.9976 - 2\rho_{(h-1)})|\mathcal{M}|$. Then during times $[\text{start}_h + \frac{2}{c}, \text{end}_h]$, the count of $\mathcal{O}$ agents at* `hour` $= h$, $|\mathcal{O}_h| \ge (0.97 - 2\rho_{(h-1)})|\mathcal{M}|$.

PROOF. We show the first claim, that during synchronous hour h, the count $|\mathcal{O}| \ge (0.9976 - 2\rho_{(h-1)})|\mathcal{M}|$ by process of elimination. By theorem 3.6.22, by time $\text{end}_{h-1} < \text{start}_h$, the total mass $\mu \le \rho_{(h-1)}|\mathcal{M}|2^{-h+1}$, which implies the total count $\{a : a.\texttt{exponent} \ge -h\} \le 2\rho_{(h-1)}|\mathcal{M}|$ in all future configurations, since total mass is nonincreasing. Then by theorem 3.6.17, the count $|\{\mathcal{O} : \mathcal{O}.\texttt{hour} > h\} \cup \{b : b.\texttt{exponent} < -h\}| \le 0.0024|\mathcal{M}|$ until time end_h. This leaves $|\mathcal{M}| - 0.0024|\mathcal{M}| - 2\rho_{(h-1)}|\mathcal{M}| = (0.9976 - 2\rho_{(h-1)})|\mathcal{M}|$ agents who must be $\mathcal{O}$ agents with `hour` $\ge h$ until time end_h.

By definition of time start_h, there are at least $0.9|\mathcal{C}|$ Clock agents with $\texttt{hour} \geq h$ that will bring these $\mathcal{O}$ agents up to $\texttt{hour} = h$. We need to bring all but $0.0276|\mathcal{M}|$ of these agents up to $\texttt{hour} = h$ to achieve the desired bound $|\mathcal{O}_h| \geq (0.97 - 2\rho_{(h-1)})|\mathcal{M}|$. We can apply theorem 3.6.3, with $a = 0.9c$, $b_1 = (0.9976 - 2\rho_{(h-1)})m < (0.9976 - 2 \cdot 0.1)m$, and $b_2 = 0.0276m$ to conclude this happens after parallel time t, where $t < (1+\epsilon)\mathbb{E}[t]$ with very high probability. The expected time

$$\mathbb{E}[t] = \frac{\ln(b_1) - \ln(b_2)}{2a} \leq \frac{\ln(0.7976m) - \ln(0.0276m)}{2 \cdot 0.9c} \leq \frac{1.89}{c}.$$

Thus for small constant $\epsilon > 0$, we have $t < (1+\epsilon)\frac{1.89}{c} < \frac{2}{c}$ with very high probability.

□

In order to reason about the total mass $\mu_{(>-h)}$ above exponent $-h$, we will define the potential function $\phi_{(>-h)}$, where for a biased agent a with $a.\texttt{exponent} = -i \geq -h+1$ at time t, $\phi_{(>-h)}(a,t) = 4^{-i+h-1}$. The global potential

$$\phi_{(>-h)}(t) = \sum_{a.\texttt{exponent} > -h} \phi(a,t) \geq \sum_{a.\texttt{exponent} > -h} 4^{-h+1+h-1} = |\{a : a.\texttt{exponent} > -h\}|.$$

Since $\phi_{(>-h)}$ upper bounds the count above exponent h, we can bound the mass $\mu_{(>-h)}(t) \leq 2^{-h+1}\phi_{(>-h)}(t)$. Also note that unlike the mass, $\phi_{(>-h)}(t)$ strictly decreases via split reactions since $4^{-i} > 4^{-i-1} + 4^{-i-1}$. This will let us show that $\phi_{(>-h)}$ exponentially decays to 0 when there are a constant fraction of $\mathcal{O}$ agents to do these splits.

We first show that by hour h exponents significantly far above $-h$ are empty. Letting $q = \lfloor \frac{\ln n}{3} \rfloor$, we will show the potential $\phi_{(>-h+q)}$ hits 0 by hour h.

LEMMA 3.6.20. *For each hour $h = 0, \ldots, l$, by time $\text{start}_h + \frac{2}{c}$, the maximum level among all biased agents is at most $-h+q$ with high probability $1 - O(1/n^{12})$.*

PROOF. The statement is vacuous for hours $h < q$, so we must only consider hours $h \geq q$. We use the potential $\phi_{(>-h+q)}$, and start the argument at time $t_{\text{start}} = \text{start}_{(h-q)} + \frac{2}{c} + \frac{41}{m}$ where inductively by theorem 3.6.21 $\phi_{(>-h+q)}(t_{\text{start}}) \leq 0.001|\mathcal{M}|$. We must show that by time $t_{\text{end}} = \text{start}_h + \frac{2}{c}$, we have $\phi_{(>-h+q)}(t_{\text{end}}) = 0$.

By theorem 3.6.19, the count of $\mathcal{O}$ agents with $\texttt{hour} \geq h - q$, $|\mathcal{O}_{(\geq h-q)}| \geq 0.77|\mathcal{M}|$ during all synchronous hours after $\text{start}_{(h-q)} + \frac{2}{c}$ and up through synchronous hour $l - 5 \geq h - 5$. Thus the interval we consider consists of at least $q - 5$ synchronous hours. By theorem 3.6.15, each

synchronous hour takes at least time $[1.48(45-2)-2.2]/c^2 \geq 17/c^2 \geq 51/m$, since $c < \frac{1}{3} < m$ by theorem 3.6.5. Thus the total time for this argument is at least $t_{\text{end}} - t_{\text{start}} \geq \frac{51(q-5)}{m}$ time.

For each of the $\frac{51(q-5)}{m}n$ interactions in this interval, we consider the expected change to $\phi_{(>-h+q)}$. For each biased agent a with $a.\texttt{exponent} = -i > -h+q$, a split reaction will change the potential by $\Delta\phi_a \leq 4^{h-1}(2 \cdot 4^{-i-1} - 4^{-i}) = 4^{h-1}(-\frac{1}{2}4^{-i}) = -\frac{1}{2}\phi_a$. Then in each interaction at parallel time t, the expected change in the potential

$$\begin{aligned}\mathbb{E}\left[\phi_{(>-h+q)}(t+1/n) - \phi_{(>-h+q)}(t)\right] &\leq \sum_{a.\texttt{exponent}>(-h+q)} \Pr\left[a \text{ splits}\right] \cdot \Delta\phi_a \\ &\leq \sum_{a.\texttt{exponent}>(-h+q)} \frac{2 \cdot 0.77m}{n} \cdot -\frac{1}{2}\phi_a = -\frac{0.77m}{n}\phi_{(>-h+q)}(t).\end{aligned}$$

Then we have $\mathbb{E}\left[\phi_{(>-h+q)}(t+1/n) | \phi_{(>-h+q)}(t)\right] \leq (1 - \frac{0.77m}{n})\phi_{(>-h+q)}(t)$.

We now recursively apply this bound to all $\frac{51(q-5)}{m}n$ interactions beginning at time t_{start}:

$$\begin{aligned}\mathbb{E}\left[\phi_{(>-h+q)}(t_{\text{end}}) | \phi_{(>-h+q)}(t_{\text{start}})\right] &\leq \left(1 - \frac{0.77m}{n}\right)^{\frac{51(q-5)}{m}n} \phi_{(>-h+q)}(t_{\text{start}}) \\ \mathbb{E}\left[\phi_{(>-h+q)}(t_{\text{end}})\right] &\leq \exp\left(-0.77 \cdot 51 \ln n/3\right) \cdot \exp\left(51 \cdot 5 \cdot 0.77\right) \cdot 0.001|\mathcal{M}| \\ &\leq n^{-13}\exp\left(197\right) \cdot 0.001mn \\ &= O\big(n^{1-13}\big) = O\big(n^{-12}\big).\end{aligned}$$

Finally, since $\phi_{(>-h+q)}$ takes nonnegative integer values, we can apply Markov's Inequality to conclude $\Pr\left[\phi_{(>-h+q)}(t_{\text{end}}) > 0\right] = O(1/n^{12})$. □

Now we can reason about the potential $\phi_{(>-h)}$ during hour h, which will decrease by a large constant factor. The upper bound on $\phi_{(>-h)}$ gives the a bound on the mass of the upper tail $\mu_{(>-h)}$.

LEMMA 3.6.21. *For each hour $h = 0, \ldots, l$, by time $\text{start}_h + \frac{2}{c} + \frac{41}{m}$, the potential $\phi_{(>-h)} \leq 0.001|\mathcal{M}|$ with very high probability. This implies the mass above exponent $-h$ is $\mu_{(>-h)} \leq 0.001|\mathcal{M}|2^{-h+1}$.*

PROOF. By theorem 3.6.19, after time $\text{start}_h + \frac{2}{c}$, we have a count $|\mathcal{O}_h| \geq \tau_h|\mathcal{M}|$, where the weakest bound is at hour l, where $\tau_l = 0.15$.

Inductively, we have $\phi_{(>-h+1)}(\text{start}_h) \le 0.001|\mathcal{M}|$ by time $\text{start}_{(h-1)}+\frac{2}{c}+\frac{41}{m}$. By theorem 3.6.22, by time $\text{end}_{(h-1)}$, the total mass $\mu \le \rho_{(h-1)}|\mathcal{M}|2^{-h+1} \le \rho_{(l-1)}|\mathcal{M}|2^{-h+1}$. Thus there are at most $\rho_{(l-1)}|\mathcal{M}| = 0.408|\mathcal{M}|$ biased agents with exponent $-h+1$, which lets us bound the potential $\phi_{(>-h)}(\text{start}_h + \frac{2}{c}) \le (0.408 + 4 \cdot 0.001)|\mathcal{M}| = 0.412|\mathcal{M}|$. Thus we must drop the potential by the constant factor 412.

To show that ϕ drops by a constant factor, we will use $\Phi(t) = \ln\big(\phi_{(>-h)}(t)\big)$, which will be a supermartingale. If agent a at with exponent $-i$ splits, with $\phi_a = 4^{-i+h-1}$, this changes the potential ϕ by $\Delta\phi_a = -\frac{1}{2}\phi_a$. The potential Φ then changes by

$$\Delta\Phi_a = \ln\big(\phi_{(>-h)} + \Delta\phi_a\big) - \ln\big(\phi_{(>-h)}\big) = \ln\left(1 + \frac{-\frac{1}{2}\phi_a}{\phi_{(>-h)}}\right) \le -\frac{\phi_a}{2\phi_{(>-h)}}.$$

The expected change in Φ is then

$$\mathbb{E}\left[\Delta\Phi\right] \le \sum_{a.\texttt{exponent}>-h} \Pr\left[a \text{ splits}\right] \cdot \Delta\Phi_a \le \sum_{a.\texttt{exponent}>-h} \frac{2\tau_h m}{n} \cdot -\frac{\phi_a}{2\phi_{(>-h)}} = -\frac{0.15m}{n}.$$

We define the supermartingale $(\Phi_j)_{j=0}^{\frac{41}{m}n}$, where $\Phi_j = \Phi(\text{start}_h + \frac{2}{c} + \frac{j}{n})$. Then the desired inequality is $\phi_{>(h+1)}(\text{start}_h + \frac{2}{c} + \frac{41}{m}) \le 0.001|\mathcal{M}| \le \phi_{>(h+1)}(\text{start}_h + \frac{2}{c})/412$, so we need to show $\Phi_{\frac{41}{m}n} - \Phi_0 \le \ln\big(\frac{1}{412}\big)$. The expected value

$$\mathbb{E}\left[\Phi_{\frac{41}{m}n} - \Phi_0\right] \le -\frac{0.15m}{n} \cdot \frac{41}{m}n = -6.15 \le \ln\left(\frac{1}{412}\right) - 0.12.$$

To apply Azuma's Inequality, we will need a bound on the difference $|\Phi_{j+1} - \Phi_j| \le \max\left|\frac{\phi_a}{2\phi_{(>-h)}}\right|$. During the interval we consider, $\phi_{(>-h)} \ge 0.001|\mathcal{M}|$, since after that the desired inequality will hold. By theorem 3.6.20, by time $\text{start}_h + \frac{2}{c}$, the maximum exponent in the population is at most $-h+q$, where $q = \lfloor\frac{\ln n}{3}\rfloor$, so $\phi_a \le 4^{(-h+q)+h-1} = 4^{q-1}$. Using the fact that $4^q = e^{\ln 4 \ln n/3} = O(n^{0.47})$, we can bound the largest change in Φ as

$$|\Phi_{j+1} - \Phi_j| \le \frac{4^{q-1}}{0.002|\mathcal{M}|} = O\left(\frac{4^q}{n}\right) = O(n^{0.47}/n) = O(1/n^{0.53}).$$

Now by Azuma's Inequality (theorem 3.6.1) we have

$$\Pr\left[(\Phi_{\frac{41}{m}n} - \Phi_0) - \mathbb{E}\left[\Phi_{\frac{41}{m}n} - \Phi_0\right] \ge 0.12\right] \le \exp\left(-\frac{0.12^2}{2\sum_{j=0}^{\frac{41}{m}n}(O(n^{-0.53}))^2}\right) = \exp\left(-\Theta(n^{0.06})\right).$$

Thus $\phi_{(>-h)}$ will drop by at least the constant factor 412 with very high probability, giving $\phi_{(>-h)}(\text{start}_h + \frac{2}{c} + \frac{41}{m}) \le 0.001|\mathcal{M}|$. □

LEMMA 3.6.22. *For each hour* $h = 0, \dots, l$, *by time* $\text{start}_h + \frac{2}{c} + \frac{47}{m} \le \text{end}_h$, *the total mass* $\mu \le \rho_h|\mathcal{M}|2^{-h}$ *with very high probability. The constant* $\rho_h = 0.1$ *for all* $h \le l+5$, *then the last few constants* $\rho_{l-4} = 0.104$, $\rho_{l-3} = 0.13$, $\rho_{l-2} = 0.212$, $\rho_{l-1} = 0.408$, *and* $\rho_l = 0.808$.

PROOF. We start the argument at time $\text{start}_h + \frac{2}{c} + \frac{41}{m}$. Let $A = \{a : a.\texttt{opinion} = +1, a.\texttt{exponent} = -h\}$ and $B = \{b : b.\texttt{opinion} = -1, b.\texttt{exponent} = -h\}$. We will show that by time end_h, large number of cancel reactions happen between the agents in A and B to reduce the total mass. Inductively, by end_{h-1} we have total mass $\mu \le \rho_{(h-1)}|\mathcal{M}|2^{-h+1}$. We assume in the worst case (since we need the total mass to be small) that we start with the maximum possible amount of mass $\mu = \rho_{(h-1)}|\mathcal{M}|2^{-h+1}$. We use the upper bound on the gap $g_h \le \alpha|\mathcal{M}|$, where $\alpha = 0.8 \cdot 2^{h-l}$ and assume in the worst case (since larger gaps reduce the rate of cancelling reactions) the largest possible gap $\beta_+ - \beta_- = g_h 2^{-h} = \alpha|\mathcal{M}|2^{-h}$. This gives majority mass $\beta_+ = (\rho_{(h-1)} + \frac{\alpha}{2})|\mathcal{M}|2^{-h}$ and minority mass $\beta_- = (\rho_{(h-1)} - \frac{\alpha}{2})|\mathcal{M}|2^{-h}$.

Since the minority is the limiting reactant in the cancelling reactions, we assume in the worst case the smallest possible minority count at exponent $-h$, where all mass outside of exponent $-h$ is the minority. Then the majority count at exponent $-h$ is $|A| = (\rho_{(h-1)} + \frac{\alpha}{2})|\mathcal{M}|$. By theorem 3.6.21, the mass above exponent $-h$ is $\mu_{(>-h)}(\text{start}_h + t_2) \le 0.001|\mathcal{M}|2^{-h+1}$. By theorem 3.6.17, the maximum count below exponent $-h$ is $0.0024|\mathcal{M}|$, so the mass $\mu_{(<-h)} \le 0.0024|\mathcal{M}|2^{-h-1}$. This leaves a minority count at exponent $-h$ of $|B| = (\rho_{(h-1)} - \frac{\alpha}{2} - 0.002 - 0.0012)|\mathcal{M}|$.

Now we will apply theorem 3.6.2 with $a = (\rho_{(h-1)} + \frac{\alpha}{2})m$ and $b = (\rho_{(h-1)} - \frac{\alpha}{2} - 0.0032)m$. First we consider the cases where $h \le l+5$, where $\rho_h = \rho_{(h-1)} = 0.1$ There the bound on the gap $\alpha = 0.8 \cdot 2^{(l-5)-l} = 0.025$. This gives $a = 0.1125m$ and $b = 0.0843m$. By theorem 3.6.2, the time t to cancel a fraction $d = 0.05m$ from both a and b has mean

$$\begin{aligned}\mathbb{E}[t] &\sim \frac{\ln(b) - \ln(a) - \ln(b-d) + \ln(a-d)}{2(a-b)} \\ &= \frac{\ln(0.0843) - \ln(0.1125) - \ln(0.0343) + \ln(0.0625)}{2(0.0282m)} < \frac{5.53}{m},\end{aligned}$$

so $\mathbb{E}[t](1+\epsilon) < \frac{6}{m}$ and $t < \frac{6}{m}$ with very high probability. Cancelling this fraction d reduces the total mass by $2dn2^{-h} = 0.1|\mathcal{M}| \cdot 2^{-h} = \rho_{(h-1)}|\mathcal{M}|2^{-h+1} - \rho_h|\mathcal{M}|2^{-h}$. Then the total mass is at most $\rho_h|\mathcal{M}|2^{-h}$ by time $\text{start}_h + \frac{2}{c} + \frac{47}{m}$.

For the remaining levels $h = l-4, l-3, l-2, l-1, l$, we will repeat the above argument and calculation, but now the bound will change so $\rho_h \neq \rho_{(h-1)}$. First our bound on the gap α will double as we move down each level. This causes the worst case fractions a and b to be spread further apart. Then d, the largest fraction which will cancel from both sides within $\frac{6}{m}$ time with high probability, will be smaller. This gives the new mass bound ρ_h, since from the equation $\rho_{(h-1)}|\mathcal{M}|2^{-h+1} - \rho_h|\mathcal{M}|2^{-h} = 2dn2^{-h}$, we have $d = (\rho_{(h-1)} - \frac{\rho_h}{2})m$ and $\rho_h = 2(\rho_{(h-1)} - \frac{d}{m})$. This relative total mass bound ρ_h will be growing larger as h decreases. This is to be expected, since we do in fact see the count of biased agents growing as they reach the final exponents before the count of $\mathcal{O}$ agents runs out (see fig. 3.4c and fig. 3.4d, where the value $-l = -19$).

The following table shows the constants used in the computations at each of the steps $h = l-5, l-4, \ldots, l$. The top row $h = l-5$ corresponds to the exact calculations above, then the following rows are the constants used in the same argument for lower levels.

$h = l-5$	$\alpha = 0.025$	$a = 0.1125m$	$b = 0.0843m$	$d = 0.05m$	$\mathbb{E}[t] < 5.53/m$	$\rho_{(l-5)} = 0.1$
$h = l-4$	$\alpha = 0.05$	$a = 0.125m$	$b = 0.0718m$	$d = 0.048m$	$\mathbb{E}[t] < 5.83/m$	$\rho_{(l-4)} = 0.104$
$h = l-3$	$\alpha = 0.1$	$a = 0.154m$	$b = 0.0508m$	$d = 0.039m$	$\mathbb{E}[t] < 5.66/m$	$\rho_{(l-3)} = 0.13$
$h = l-2$	$\alpha = 0.2$	$a = 0.23m$	$b = 0.0268m$	$d = 0.024m$	$\mathbb{E}[t] < 5.29/m$	$\rho_{(l-2)} = 0.212$
$h = l-1$	$\alpha = 0.4$	$a = 0.412m$	$b = 0.0088m$	$d = 0.008m$	$\mathbb{E}[t] < 2.95/m$	$\rho_{(l-1)} = 0.408$
$h = l$	$\alpha = 0.8$	$a = 0.808m$	$b = 0.0048m$	$d = 0.004m$	$\mathbb{E}[t] < 1.11/m$	$\rho_l = 0.808$

Note that for the last couple hours, the worst case for b is very small, so the amount of cancelling that happens is negligible, and the relative mass bound essentially doubles. We will see later that the minority count does in fact sharply decrease, so it is accurate that the rate of cancellation drops to zero, and the count of biased majority agents is roughly doubling for these last couple rounds. □

3.6.6. End of Phase 3. Now that we have proven theorem 3.6.18, we will finish analyzing separately the cases of an initial tie and an initial majority opinion.

In the case of a tie, the stronger bounds from theorem 3.6.18 hold all through the final hour $h = L$. Thus at hour L we still have a constant fraction $0.77|\mathcal{M}|$ of $\mathcal{O}$ agents. We now show that in the remaining $O(\log n)$ time in Phase 3, while the Clock agents with $\texttt{hour} = h$ decrement their counter in line 6, these $\mathcal{O}$ agents can keep doing split reactions to bring all remaining biased agents down to exponent $-L$. This lets us now prove theorem 3.6.7, the main result of the section for the case of a tie.

THEOREM 3.6.7. *If the initial configuration was a tie with gap 0, then by the end of Phase 3, all biased agents have* $\texttt{exponent} = -L$*, with high probability* $1 - O(1/n^2)$*.*

PROOF. We start by using theorem 3.6.18, where at hour L, the total mass $\mu(\text{end}_L) \leq 0.1|\mathcal{M}|2^{-L}$. Thus the count of biased agents is at most $0.1|\mathcal{M}|$. We also have at least $0.77|\mathcal{M}|$ $\mathcal{O}$ agents with $\texttt{hour} = L$ (this count is actually slightly higher, as we could show almost all of the $0.9|\mathcal{M}|$ $\mathcal{O}$ agents reach $\texttt{hour} = L$ quickly by the same argument as theorem 3.6.19, but this bound will suffice).

We use the potential $\phi_{(>-L)}$, where by theorem 3.6.21, we have $\phi_{(>-L)}(\text{end}_0) \leq 0.001|\mathcal{M}|$. The goal is to now show that $\phi_{(>-L)} = 0$ within the $O(\log n)$ time it takes for any $\texttt{counter}$ to hit 0 and trigger the move to the next phase. This proof proceeds identically to the proof of theorem 3.6.20, where we had shown that $\mathbb{E}\left[\phi_{(>-L)}(t + 1/n)|\phi_{(>-L)}(t)\right] \leq (1 - \frac{0.77m}{n})\phi_{(>-L)}(t)$. We again recursively bound the expected value of $\phi_{(>-L)}$ after an additional $16 \ln n$ time.

$$\mathbb{E}\left[\phi_{(>-L)}(\text{end}_L + 16\ln n)|\phi_{(>-L)}(\text{end}_0)\right] \leq \left(1 - \frac{0.77m}{n}\right)^{16n\ln n} \phi_{(>-L)}(\text{end}_0)$$

$$\mathbb{E}\left[\phi_{(>-L)}(\text{end}_L + 16\ln n)\right] \leq \exp\left(-0.77 \cdot 0.25 \cdot 16\ln n\right) \cdot 0.001|\mathcal{M}|$$

$$\leq n^{-3.08} 0.001mn = O(1/n^2).$$

Again we conclude by Markov's Inequality that $\Pr\left[\phi_{(>-L)}(\text{end}_L + 16\ln n) > 0\right] \leq O(1/n^2)$.

Finally we must argue that Phase 3 lasts at least until time $\text{end}_L + 16\ln n$. We will bound the probability that a Clock agent can decrement its $\texttt{counter}$ down to 0 before time $\text{end}_L + 16\ln n$. Note the if statement on line 6 will only decrement the counter when both Clock agents have reached the final minute. Even if in the worst case an agent was at the final minute at the beginning of the phase, until time end_{L-1} the count of other Clock agents with $\texttt{minute} = kL$ is at most $0.1p|\mathcal{C}| = 0.001|\mathcal{C}|$. By theorem 3.6.15, the time between consecutive hours is at most $\frac{3.86k}{c^2} < \frac{290}{c}$ using $k = 45$ and the bound $c > 0.2$ with very high probability from theorem 3.6.5. Then the

number of interactions until end_{L-1} is at most $\frac{290}{c}Ln$. In each interaction, the probability of the Clock agent decrementing its counter is at most $2 \cdot 0.001c \cdot \frac{1}{n}$, so the expected number of decrements $\mathbb{E}[X] \leq 290L \cdot 0.002 \leq 0.58(\log_2(n)) \leq 0.84 \ln n$. Then by Chernoff bounds in theorem 1.3.1, with $\mu = 0.84 \ln n, \delta = 5$, we have

$$\Pr[X \geq 5.04 \ln n] \leq \Pr[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2+\delta}\right) = \exp(-3 \ln n) = n^{-3}.$$

Then after time end_{L-1}, we assume that all Clock agents have $\texttt{minute} = kL$. In this case, the probability a given Clock agent decrements its counter is at most $\frac{2c}{n}$. Thus in $16n \ln n$ interactions, the expected number of decrements $\mathbb{E}[X] \leq 32c \ln n \leq 11 \ln n$, using $c < \frac{1}{3}$ from theorem 3.6.5. We again use theorem 1.3.1, with $\mu = 11 \ln n, \delta = 0.9$, to conclude

$$\Pr[X \geq 20.9 \ln n] \leq \Pr[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2+\delta}\right) \leq \exp(-3.07 \ln n) \leq n^{-3}.$$

Then by union bound, Clock agents decrements their counter at most $5.04 \ln n + 20.9 \ln n < 26 \ln n$ times with high probability $1 - O(1/n^2)$. In Phase 3, we initialize $\texttt{counter} \leftarrow c_3 \ln n$, so setting this counter constant $c_3 = 26$, we conclude that no agent moves to the next phase before time $\text{end}_L + 16 \ln n$ with high probability. This gives enough time to conclude that all biased agents have $\texttt{exponent} = -L$ by the end of Phase 3. □

Now we consider the case where there is an initial majority, which we have assumed without loss of generality was A. Here the argument of theorem 3.6.18 stops working at exponent $-l$, because of the substantial relative gap $0.4|\mathcal{M}| \leq g_l < 0.8|\mathcal{M}|$. Next we will need to show that the count of $\mathcal{O}$ gets brought to almost 0 by split reactions with the majority. In order to show that the count of $\mathcal{O}$ stays small however, we will need some way to bound future cancel reactions. To do this, we will now show that during the last few hours $l-4, \ldots, l$, the minority mass β_- drops to a negligible amount. We will then use this tight bound to show the majority consumes most remaining $\mathcal{O}$ agents, and finally that constraints on the majority mass β_+ will force the majority count to remain above 99% at exponents $-l, -(l+1), -(l+2)$ for all reachable configurations in the rest of Phase 3.

We first show inductively that the minority mass becomes negligible during the last few hours $l-5, \ldots, l$. We already showed a bound on the total mass in theorem 3.6.22, which used an upper bound on the gap. If we want the minority mass to be small, now the worst case is the smallest

possible gap. Notice that just using the final mass bound $\mu < \rho_l|\mathcal{M}|2^{-l} = 0.808|\mathcal{M}|2^{-l}$ in the worst case with the smallest possible gap $g_l = 0.4$ at exponent $-l$, would imply $\beta^+ \approx 0.6|\mathcal{M}|2^{-l}$ and $\beta^- \approx 0.2|\mathcal{M}|2^{-l}$. To get a much tighter upper bound on minority mass, we will make a similar inductive argument to theorem 3.6.22, now using the lower bound for each gap g_h, to show enough minority mass cancels at exponent $-h$ during each hour h.

LEMMA 3.6.23. *For each hour $h = l-5, \ldots, l$, by time end_h, the minority mass $\beta_- \leq \xi_h|\mathcal{M}|2^{-h}$ with very high probability. The constants $\xi_{(l-5)} = 0.04375$, $\xi_{(l-4)} = 0.0375$, $\xi_{(l-3)} = 0.0267$, $\xi_{(l-2)} = 0.0145$, $\xi_{(l-1)} = 0.0056$, and $\xi_l = 0.004$.*

PROOF. For the base case $h = l-5$, by theorem 3.6.22, the total mass $\mu(\text{end}_{(l-5)}) \leq 0.1|\mathcal{M}|2^{-l+5}$. The relative gap $0.0125|\mathcal{M}| \leq g_{(l-5)} < 0.025|\mathcal{M}|$, so now in the worst case (for minority mass being small), we assume the smallest gap $g_{(l-5)} = 0.0125|\mathcal{M}|$ and largest total mass $\mu(\text{end}_{(l-5)}) = \beta_+ + \beta_- = 0.1|\mathcal{M}|2^{-l+5}$. Since the invariant gap $g = \beta_+ - \beta_- = g_{(l-5)}2^{-l+5}$, we have $\beta_+ = 0.05625|\mathcal{M}|2^{-l+5}$ and $\beta_- = 0.04375|\mathcal{M}|2^{-l+5}$, giving an upper bound constant $\xi_{(l-5)} = 0.04375$.

Now we outline the inductive step, for each level $h = l-4, \ldots, l$. We will show the constants for the first step $h = l-4$ here, and then list constants for the remaining steps in a table below. Let $A = \{a : a.\texttt{opinion} = +1, a.\texttt{exponent} = -h\}$ and $B = \{b : b.\texttt{opinion} = -1, b.\texttt{exponent} = -h\}$.

We start the argument at time $\text{start}_h + \frac{2}{c} + \frac{41}{m}$, and with the previous bound $\beta_-(\text{end}_{(h-1)}) = \xi_{(h-1)}|\mathcal{M}|2^{-h+1} = 0.0875|\mathcal{M}|2^{-h}$. We will then assume in the worst case the smallest possible gap $g_h = \alpha|\mathcal{M}| = 0.4 \cdot 2^{h-l}|\mathcal{M}| = 0.025|\mathcal{M}|$. This gives $\beta_+ = (2\xi_{(h-1)} + \alpha)|\mathcal{M}|2^{-h}$. We then assume in the worst case that all mass allowed outside exponent $-h$ belongs to the minority (so doesn't reduce by cancelling), giving majority count $|A| = (2\xi_{(h-1)} + \alpha)|\mathcal{M}|$. We also assume the maximum amount of mass outside exponent $-h$. By theorem 3.6.17, before time end_h, the count below exponent $-h$ is at most $0.0024|\mathcal{M}|$, so the mass is at most $0.0024|\mathcal{M}|2^{-h-1}$. By theorem 3.6.21, after time $\text{start}_h + \frac{2}{c} + \frac{41}{m}$, the mass above exponent $-h$ is $\mu_{(>-h)} \leq 0.001 \cdot 2^{-h+1}$. This leftover remaining mass at exponent $-h$ gives minority count $|B| = (2\xi_{(h-1)} - 0.0012 - 0.002)|\mathcal{M}|$.

Now we will apply theorem 3.6.2 with fractions $a = (2\xi_{(h-1)} + \alpha)m = 0.1125m$ and $b = (2\xi_{(h-1)} - 0.0036)m = 0.0843m$. These match the first fractions a, b used in the proof of theorem 3.6.22, and

again we use that the time t to cancel a fraction $d = 0.05m$ from both a and b has mean

$$\begin{aligned}\mathbb{E}[t] &\sim \frac{\ln(b) - \ln(a) - \ln(b-d) + \ln(a-d)}{2(a-b)} \\ &= \frac{\ln(0.0843) - \ln(0.1125) - \ln(0.0343) + \ln(0.0625)}{2(0.0282m)} < \frac{5.53}{m},\end{aligned}$$

so $\mathbb{E}[t](1+\epsilon) < \frac{6}{m}$ and $t < \frac{6}{m}$ with very high probability. Cancelling this fraction d reduces the minority mass by $dn2^{-h} = 0.05|\mathcal{M}| \cdot 2^{-h}$, giving a new minority mass $\beta_- = (2\xi_{(h-1)} - \frac{d}{m})|\mathcal{M}|2^{-h} = 0.0375|\mathcal{M}|2^{-h}$. Thus for $\xi_h = 0.0375$, the minority mass is at most $\xi_h|\mathcal{M}|2^{-h}$ by time $\text{start}_h + \frac{2}{c} + \frac{47}{m}$.

For the remaining levels $h = l-4, l-3, l-2, l-1, l$, we will repeat the above argument and calculation. The following table shows the constants used in the computations at each of the steps $h = l-4, l-3, \ldots, l$. The top row $h = l-4$ corresponds to the exact calculations above, then the following rows are the constants used in the same argument for lower levels.

$h = l-4$	$\alpha = 0.025$	$a = 0.1125m$	$b = 0.0843m$	$d = 0.05m$	$\mathbb{E}[t] < 5.53/m$	$\xi_{(l-4)} = 0.0375$
$h = l-3$	$\alpha = 0.05$	$a = 0.125m$	$b = 0.0718m$	$d = 0.048m$	$\mathbb{E}[t] < 5.83/m$	$\xi_{(l-3)} = 0.0267$
$h = l-2$	$\alpha = 0.1$	$a = 0.154m$	$b = 0.0508m$	$d = 0.039m$	$\mathbb{E}[t] < 5.66/m$	$\xi_{(l-2)} = 0.0145$
$h = l-1$	$\alpha = 0.2$	$a = 0.23m$	$b = 0.0268m$	$d = 0.024m$	$\mathbb{E}[t] < 5.29/m$	$\xi_{(l-1)} = 0.0056$
$h = l$	$\alpha = 0.4$	$a = 0.412m$	$b = 0.0088m$	$d = 0.008m$	$\mathbb{E}[t] < 2.95/m$	$\xi_l = 0.004$

Note that the structure of the proof yields the same sequence of values a, b as the proof of theorem 3.6.22, so we have repeated the exact same applications of theorem 3.6.2. This time, however, we are using the cancelled fraction d to show the minority mass becomes very small. □

Now theorem 3.6.23 gives that minority mass $\beta_- \leq 0.004|\mathcal{M}|2^{-l}$ by time end_l. The rest of the argument will not try to reason about probabilistic guarantees on what the distribution looks like. Instead, we will make claims purely about reachability, and show that we get to a configuration where invariants force the count of majority agents to remain high in any reachable configuration using the transitions of Phase 3.

LEMMA 3.6.24. *By the time* $\text{end}_{(l+2)}$*, the count of majority agents with* `exponent` $\in \{-l, -(l+1), -(l+2)\}$ *is at least* $0.96|\mathcal{M}|$*, with very high probability. Then in all reachable configurations*

where all agents are still in Phase 3, the count of majority agents with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$ *is at least* $0.92|\mathcal{M}|$.

Proof. We argue by process of elimination. First we apply theorem 3.6.17 to conclude that until time $\text{end}_{(l+2)}$, the count $|\{\mathcal{O} : \mathcal{O}.\texttt{hour} > l+2\} \cup \{b : b.\texttt{exponent} < -(l+2)\}| \leq 0.0024|\mathcal{M}|$ with very high probability.

Now theorem 3.6.23 gives that minority mass $\beta_- \leq 0.004|\mathcal{M}|2^{-l}$ after time end_l. Then we can bound the maximum minority count at exponent $-(l+2)$ and above by $0.016|\mathcal{M}|$. In addition, these minority agents could eliminate more majority agents by cancel reactions. The count of agents they could cancel with is at most $0.016|\mathcal{M}|$.

This leaves at least $|\mathcal{M}| - 0.0024|\mathcal{M}| - 0.016|\mathcal{M} - |0.016|\mathcal{M}| = 0.9656|\mathcal{M}|$ agents that are either majority or $\mathcal{O}$ agents at time end_l. We will next show that in hour $l+2$, most of any remaining $\mathcal{O}$ agents set $\texttt{hour} = l+2$ and then are consumed in split reactions with majority agents with $\texttt{exponent} > -(l+2)$.

The majority mass $\beta_+ \geq g = g_l 2^{-l} \geq 0.4|\mathcal{M}|2^{-l}$. By theorem 3.6.21, at most $0.001|\mathcal{M}|2^{-l+1} = 0.002|\mathcal{M}|2^{-l}$ of β_+ can come from exponents above $-l$. Thus we need mass $0.398|\mathcal{M}|2^{-l}$ from majority agents at exponent $-l$ and below. Note that if the count $|\{a : a.\texttt{opinion} = +1, a.\texttt{exponent} \in \{-l, -(l+1)\}\}| \leq 0.19|\mathcal{M}|$, then the maximum majority mass could achieve would be $0.81|\mathcal{M}|$ with $\texttt{exponent} = -l-2$ and $0.19|\mathcal{M}|$ with $\texttt{exponent} = -l$, which gives $(\frac{0.81}{4} + 0.19)|\mathcal{M}|2^{-l} < 0.398|\mathcal{M}|2^{-l}$. This implies we must have a count of at least $0.19|\mathcal{M}|$ of majority agents at exponents $-(l+1)$ and $-l$ in all reachable configurations.

Next we show that all but at most $0.0036|\mathcal{M}|$ of these $\mathcal{O}$ agents gets brought up to $\texttt{hour} = l+2$. We start at time start_{l+2}, when the count $|\mathcal{O}| \leq (0.9656 - 0.398)\mathcal{M} = 0.5676|\mathcal{M}|$, and the count of clock agents at $\texttt{hour} = l+2$ is at least $0.9|\mathcal{C}|$. Thus we can apply theorem 3.6.3, with $a = 0.9c$, $b_1 = 0.5676m$, $b_2 = 0.0056m$ to conclude that at most $0.0056|\mathcal{M}|$ of $\mathcal{O}$ agents are left at the wrong hour after parallel time $t_1 < (1+\epsilon)\mathbb{E}[t_1]$, where the expected time

$$\mathbb{E}[t_1] = \frac{\ln(b_1) - \ln(b_2)}{2a} = \frac{\ln(0.5676m) - \ln(0.0036m)}{1.8c} \leq \frac{2.82}{c}.$$

Thus with very high probability $t_1 < \frac{3}{c}$.

Next the at least $0.19|\mathcal{M}|$ majority agents at exponents $-(l+1)$ and $-l$ will eliminate these $\mathcal{O}$ agents by split reactions. We show that at most $0.001|\mathcal{M}|$ of $\mathcal{O}_h$ agents are left. We again apply

theorem 3.6.3, with $a = 0.19m$, $b_1 = 0.5676m$, $b_2 = 0.01m$ to conclude this takes at most parallel time $t_2 < (1+\epsilon)\mathbb{E}[t_2]$, where

$$\mathbb{E}[t_2] = \frac{\ln(b_1) - \ln(b_2)}{2a} = \frac{\ln(0.5676m) - \ln(0.001m)}{0.38m} \leq \frac{16.69}{m}.$$

Thus with very high probability $t_2 < \frac{17}{m}$.

Account for these counts $0.0032|\mathcal{M}|$ and $0.001|\mathcal{M}|$ of leftover $\mathcal{O}$ agents, along with the maximal count $0.001|\mathcal{M}|$ of biased agents above exponent $-l$, we are left with $0.9656|\mathcal{M}| - 0.0032|\mathcal{M}| - 0.001|\mathcal{M}| - 0.001|\mathcal{M}| = 0.96|\mathcal{M}|$ agents that must be majority agents with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$.

Now we argue that no reachable configurations can bring this count below $0.92|\mathcal{M}|$. We have already accounted for the maximum number of minority agents that can do cancel reactions at exponents $-l, -(l+1), -(l+2)$. Thus we only have to argue that no amount of split reactions can bring this count down by $0.04|\mathcal{M}|$. Note that reducing the count by $0.04|\mathcal{M}|$ will reduce the mass at these exponents by at least $0.04|\mathcal{M}|2^{-(l+2)}$. It will take at least $0.08|\mathcal{M}|$ agents below exponent $-l-2$ to account for the same mass, for a total of $0.96|\mathcal{M}| - 0.04|\mathcal{M}| + 0.08|\mathcal{M}| = |\mathcal{M}|$. Thus it is not possible to move more than $0.04|\mathcal{M}|$ majority agents below exponent $-(l+2)$ by split reactions, because there are not enough Main agents to account for the mass. □

We can now use the results of theorem 3.6.24, theorem 3.6.21, and theorem 3.6.23, including the high probability requirements from all previous lemmas, to prove theorem 3.6.8, the main result for Phase 3 in the non-tie case:

THEOREM 3.6.8. *Assume the initial gap $|g| < 0.025|\mathcal{M}|$. Let the exponent $-l = \lfloor \log_2(\frac{g}{0.4|\mathcal{M}|}) \rfloor$. Let $i = sign(g)$ be the majority opinion and M be the set of all agents with* $\texttt{role} = \mathsf{Main}$, $\texttt{opinion} = i$, $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$. *Then at the end of Phase 3, $|M| \geq 0.92|\mathcal{M}|$ with high probability $1 - O(1/n^2)$.*

In addition, the total mass above exponent $-l$ is $\mu_{(>-l)} = \sum_{a.\texttt{exponent} > -l} |a.\texttt{bias}| \leq 0.002|\mathcal{M}|2^{-l}$, and the total minority mass is $\beta_- = \sum_{a.\texttt{opinion} = -i} |a.\texttt{bias}| \leq 0.004|\mathcal{M}|2^{-l}$.

3.6.7. Analysis of final phases.

3.6.8. Reserve agent Phases Protocol 10 and Protocol 11. We now consider the behavior of the Reserve agents in Phase 5 and Phase 6. We first show that with high probability they are also able to set their `sample` field during Phase 5.

LEMMA 3.6.25. *By the end of Phase 5, all* Reserve *agents have* $\texttt{sample} \neq \perp$*, with high probability* $1 - O(1/n^2)$.

PROOF. By theorem 3.6.8, we have at least $0.92|\mathcal{M}|$ majority agents with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$ by the end of Phase 3. Now we can analyze the process of Reserve agents sampling the `exponent` of the first biased agent they encounter in Phase 5 with theorem 3.6.3, where subpopulations $A = \{a : a.\texttt{role} = \mathsf{Main}, a.\texttt{bias} = \pm 1\}$ with and $B = \mathcal{R}$, comprising fractions $a \geq 0.92m$ and $b_1 = r$. Then by theorem 3.6.3, all Reserve agents set their `sample` within time t, where $t \leq \frac{5 \ln n}{2 \cdot 0.92m}$ with high probability $1 - O(1/n^2)$. Thus for appropriate choice of counter constant c_5, this will happen before the first Clock agent advances to the next phase. □

We next show that the Reserve agents are able to bring the exponents of all biased agents down to at most $-l$ during Phase 6.

LEMMA 3.6.26. *By the end of Phase 6, all biased agents have* $\texttt{exponent} \leq -l$*, with high probability* $1 - O(1/n^2)$.

PROOF. theorem 3.6.8, the mass above exponent $-l$ is $\mu_{(>-l)} \leq 0.001|\mathcal{M}|2^{-l+1}$. We must show that all of this mass moves down to at least exponent $-l$ via split reactions with the Reserve agents (line 3 in Phase 6). We consider the following two cases based on whether the size of $A_{-l} = \{a : a.\texttt{opinion} = +1, a.\texttt{exponent} = -l\}$.

In the first case, $|A| > 0.19|\mathcal{M}|$ and we will show all agents get brought down to $\texttt{exponent} \leq -l$. Because of the sampling process in Phase 5, the count of $R_{-l} = \{r : r.\texttt{role} = \mathsf{Reserve}, r.\texttt{sample} = -l\}$ has size at least $|R_{-l}| \geq 0.18|\mathcal{R}|$ with very high probability, by standard Chernoff bounds. If all the agents above exponent $-l$ split down to exponent $-l$, they would have count at most $0.002|\mathcal{M}|$, potentially all coming out of the initial count of R_{-l}. Thus for the entirety of Phase 6, we have

$$|R_{-l}| \geq 0.18|\mathcal{R}| - 0.002|\mathcal{M}| = n(0.18r - 0.002m) \geq n(0.18 \cdot 0.24 - 0.002) \geq 0.04n,$$

using the very high probability bound $r > 0.24$ from theorem 3.6.5.

In the second case $|A| \le 0.19|\mathcal{M}|$. Now we cannot make guarantees on the size $|R_{-l}|$, so we will instead reason about $A_{-(l+1)} = \{a : a.\texttt{opinion} = +1, a.\texttt{exponent} = -(l+1)\}$ and $R_{-(l+1)} = \{r : r.\texttt{role} = \mathsf{Reserve}, r.\texttt{sample} = -(l+1)\}$.

We first show that $|A_{-(l+1)}| > 0.59|\mathcal{M}|$. Recall by definition of l and g_l that the majority mass $\beta_+ \ge g_l 2^{-l} \ge 0.4|\mathcal{M}|2^{-l}$. At most $0.001|\mathcal{M}|2^{-l+1}$ can come from exponents above $-l$ and at most $0.19|\mathcal{M}|2^{-l}$ comes from exponent $-l$. Thus there must be at least mass $(0.398 - 0.19)|\mathcal{M}|2^{-l}$ from $A_{-(l+1)}$ and the exponents below. If $|A_{-(l+1)}| = 0.59|\mathcal{M}|$, then even putting all $0.41|\mathcal{M}|$ remaining Main agents at $\texttt{exponent} = -(l+2)$ would only give mass $0.59|\mathcal{M}|2^{-(l+1)} + 0.41|\mathcal{M}|2^{-(l+2)} = 0.3975|\mathcal{M}|2^{-l}$. Thus we require $|A_{-(l+1)}| > 0.59|\mathcal{M}|$. Again by standard Chernoff bounds from the sampling process of Phase 5, this implies the initial size of $|R_{-(l+1)}| \ge 0.58|\mathcal{R}|$.

If all the agents above exponent $-l$ split down to exponent $-(l+1)$, they would have count at most $0.004|\mathcal{M}|$, potentially all coming out of the initial count of R_{-l}. In addition, we can lose count $|A| \le 0.19|\mathcal{M}|$ from R_{-l} from additional split reactions. This implies that for the entirety of Phase 6, we have count at least

$$\begin{aligned}|R_{-(l+1)}| &\ge 0.58|\mathcal{R}| - 0.004|\mathcal{M}| - 0.19|\mathcal{M}| = n(0.58r - 0.004m - 0.19m)\\ &\ge n(0.58 \cdot 0.24 - (0.004 + 0.19) \cdot 0.51) \ge 0.04n,\end{aligned}$$

using the very high probability bounds $r > 0.24$ and $m < 0.51$ from theorem 3.6.5.

Thus in both cases we maintain a count of at least $0.04n$ $\mathsf{Reserve}$ agents at level $-l$ or $-(l+1)$. For an agent a with $a.\texttt{exponent} > -l$, the probability that in a given interaction agent a does a split reaction (line 3 in Phase 6) with an agent in $R_{-l} \cup R_{-(l+1)}$ is at least $\frac{2 \cdot 0.04n \cdot 1}{n^2} = \frac{0.08}{n}$. Now we proceed as in the proofs of theorem 3.6.20 and theorem 3.6.7, analyzing the potential $\phi_{(>-l)}$ to show it hits 0 in $O(\log n)$ time.

Recall that for each biased agent a with $a.\texttt{exponent} = -i > -l$ and local potential $\phi_a = 4^{-i+l-1}$, a split reaction changes the potential by $\Delta\phi_a \le 4^{l-1}(2 \cdot 4^{-i-1} - 4^{-i}) = 4^{l-1}(-\frac{1}{2}4^{-i}) = -\frac{1}{2}\phi_a$. Then in each interaction at parallel time t, the expected change in the potential

$$\begin{aligned}\mathbb{E}\left[\phi_{(>-l)}(t + 1/n) - \phi_{(>-l)}(t)\right] &\le \sum_{a.\texttt{exponent} > -l} \Pr\left[a \text{ splits}\right] \cdot \Delta\phi_a\\ &\le \sum_{a.\texttt{exponent} > -l} \frac{0.08}{n} \cdot -\frac{1}{2}\phi_a = -\frac{0.04}{n}\phi_{(>-l)}(t).\end{aligned}$$

Thus we have $\mathbb{E}\left[\phi_{(>-l)}(t+1/n)|\phi_{(>-l)}(t)\right] \le (1-\frac{0.04}{n})\phi_{(>-l)}(t)$.

When we begin the argument at time t_{start} at the beginning of Phase 6, we start with $\phi_{(>-l)}(t_{\text{start}}) \le 0.001|\mathcal{M}|$ from the final iteration of theorem 3.6.21. We will wait until time $t_{\text{end}} = t_{\text{start}} + 75\ln n$, and now recursively bound the potential after these $75n\ln n$ interactions:

$$\begin{aligned}\mathbb{E}\left[\phi_{(>-l)}(t_{\text{end}})|\phi_{(>-l)}(t_{\text{start}})\right] &\le \left(1-\frac{0.04}{n}\right)^{75n\ln n}\phi_{(>-l)}(t_{\text{start}})\\ \mathbb{E}\left[\phi_{(>-h+q)}(t_{\text{end}})\right] &\le \exp\left(-0.04\cdot 75\ln n\right)\cdot 0.001|\mathcal{M}|\\ &\le n^{-3}\cdot 0.001mn = O\big(n^{-2}\big)\end{aligned}$$

Finally, since $\phi_{(>-l)}$ takes nonnegative integer values, we can apply Markov's Inequality to conclude $\Pr\left[\phi_{(>-l)}(t_{\text{end}})>0\right] = O(1/n^2)$. So after $75\ln n$ time in Phase 6, all biased agents have $\texttt{exponent} \le -l$, with high probability. For appropriate choice of counter constant c_6, this happens before any Clock agent advances to the next phase. □

Recall M is the set of all majority agents with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$, where $|M| \ge 0.92|\mathcal{M}|$ at the end of Phase 3 by theorem 3.6.8. We finally observe that after Phase 6, M is still large.

Lemma 3.6.27. *At the end of Phase 6, $|M| \ge 0.87|\mathcal{M}|$ with very high probability.*

Proof. By theorem 3.6.8, $|M| \ge 0.92|\mathcal{M}|$ at the end of Phase 3, so the count of all other main agents is at most $0.08|\mathcal{M}|$. By standard Chernoff bounds on the sampling process in Phase 5, the count

$$|\{r : r.\texttt{role} = \mathsf{Reserve}, r.\texttt{sample} \notin \{-l, -(l+1), -(l+2)\}\}| \le 0.09|\mathcal{R}| \le 0.05|\mathcal{M}|.$$

Split reactions that consume these Reserve agents are the only way to bring agents out of the set M. Thus at the end of Phase 6, the count is still at least

$$|M| \ge 0.92|\mathcal{M}| - 0.09|\mathcal{R}| \ge 0.87|\mathcal{M}|,$$

using $|\mathcal{R}| < \frac{5}{9}|\mathcal{M}|$ with very high probability by theorem 3.6.5. □

3.6.9. Minority elimination Phases Protocol 12 and Protocol 13. Next we argue that in Phase 7, the agents in M are able to eliminate all minority agents with $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$. Note that these minority agents are able to do a cancel reaction with any agent in M, since by design Phase 7 allows reactions between agents with an exponent gap of at most 2. We first argue that $|M|$ must stay large through the entirety of Phase 7:

LEMMA 3.6.28. *At the end of Phase 7,* $|M| \geq 0.8|\mathcal{M}|$.

PROOF. We use the bound on minority mass $\beta_- \leq 0.004|\mathcal{M}|2^{-l}$ from theorem 3.6.8. This implies the minority count at exponent $-(l+4)$ and above is at most $0.064|\mathcal{M}|$, since a $0.064|\mathcal{M}| \cdot 2^{-(l+4)} \geq \beta$. Note that cancelling with these minority agents is the only way for a majority agent in M to change its state in Phase 7. Using the previous bound from theorem 3.6.27, the count of M can decrease at most to $|M| = 0.87|\mathcal{M}| - 0.064|\mathcal{M}| \geq 0.8|\mathcal{M}|$. □

Now we argue that these agents in M eliminate all high exponent minority agents.

LEMMA 3.6.29. *At the end of Phase 7, all minority agents have* $\texttt{exponent} < -(l+2)$*, with high probability* $1 - O(1/n^2)$.

PROOF. From theorem 3.6.26, all minority agents already have $\texttt{exponent} \leq -l$.

Let $B_{-l}, B_{-(l+1)}, B_{-(l+2)}$ be the sets of minority agents with $\texttt{exponent} = -l, -(l+1), -(l+2)$, respectively. We argue successively that $|B_{-l}| = 0$, then $|B_{-(l+1)}|$, then $|B_{-(l+2)}| = 0$.

The initial bound on minority mass $\beta_- \leq 0.004|\mathcal{M}|2^{-l}$ from theorem 3.6.8 implies that initially $|B_{-l}| \leq 0.004|\mathcal{M}|$. Note that an interaction with any agent in M will bring remove an agent from B_{-l} via one of the Phase 7 cancel reactions. Using the bound $|M| \geq 0.8|\mathcal{M}|$ during the entire phase from theorem 3.6.28, we can use theorem 3.6.3 to bound the time for $|B_{-l}| = 0$. We have constants $a = 0.8m$ and $b_1 = 0.004m$. Then by theorem 3.6.28, with high probability $1 - O(1/n^2)$, the time t_1 to eliminate the count of B_{-l} is at most $t_1 \leq \frac{5 \ln n}{2(0.8m - 0.004m)} \leq 6.41 \ln n$, using the bound $m \geq 0.49$ from theorem 3.6.5.

Next we wait to eliminate the count of $B_{-(l+1)}$, by a similar argument. Initially the count is at most $|B_{-(l+2)}| \leq 0.008|\mathcal{M}|$, and this will all cancel in at most time t_2. By theorem 3.6.28, with high probability, $t_2 \leq \frac{5 \ln n}{2(0.8m - 0.008m)} \leq 6.45 \ln n$. The same argument for $B_{-(l+2)}$, initially $|B_{-(l+2)}| \leq 0.016|\mathcal{M}|$, gives $t_3 \leq \frac{5 \ln n}{2(0.8m - 0.016m)} \leq 6.51 \ln n$.

Thus the entire process requires time at most $t_1 + t_2 + t_3 < 20 \ln n$. So for appropriate choice of counter constant c_7, this will happen before the first Clock agent advances to the next phase. □

Now we will prove that the remaining $0.8|\mathcal{M}|$ majority agents in M are able to eliminate all remaining minority agents in Phase 8.

LEMMA 3.6.30. *At the end of Phase 8, there are no more minority agents, with high probability* $1 - O(1/n^2)$.

PROOF. From theorem 3.6.28 and theorem 3.6.29, by the end of Phase 7, we have $|M| \geq 0.8|\mathcal{M}|$ and all minority agents have `exponent` $< -(l+2)$. We assume in the worst case all $0.2|\mathcal{M}|$ other Main agents have the minority opinion. Note by the consumption reaction in Phase 8, every minority agent will be eliminated by an agent in M that still has `full` = False. Thus we can apply theorem 3.6.2, with $a = 0.8m$ and $b = 0.2m$, to conclude that all remaining minority agents are eliminated in time t, where $t \leq \frac{5 \ln n}{2(0.8m - 0.2m)} \leq 8.5 \ln n$, using $m \geq 0.495$ with very high probability from theorem 3.6.5. So for appropriate choice of counter constant c_8, this will happen before the first Clock agent advances to the next phase. □

3.6.10. Fast Stabilization. We next give a time bound for the stable backup:

LEMMA 3.6.31. *The 6-state protocol in Phase 10 stably computes majority in $O(n \log n)$ stabilization time, both in expectation and with high probability.*

PROOF. Note that agents in the initial state with `active` = True and `output` $\in \{\mathsf{A}, \mathsf{B}\}$ can only change their state by a cancel reaction in line 3 with another active agent with the opposite opinion.

First we consider the case where one opinion, without loss of generality A, is the majority. We first wait for all active B agents to meet an active A agent and changed their state to active T via line 3. This is modelled by the "cancel reaction" process described in theorem 3.6.2, taking expected $O(n)$ time and $O(n \log n)$ time with high probability. At this point, there are no active B agents and at least one active A agent remaining. We next wait for this active A agent to interact with all remaining active T agents, which will then become passive via line 5. This takes $O(n \log n)$ time in expectation and with high probability by a standard coupon-collector argument, after which point there are only active A agents and passive agents. Finally, we wait for these active A agents

to convert all passive agents to $\texttt{output} = \mathsf{A}$ via line 7, taking another $O(n \log n)$ time by the same coupon-collector argument.

Next we consider the case where the input distribution is a tie. We first wait for all $n/2$ pairs of active A and B agents to cancel via line 3, taking $O(n)$ time in expectation and $O(n \log n)$ time with probability. Consider the last such interaction. At this point, those two agents have become active T agents, and there are no active A or B agents left. Thus after interacting with one of the T agents, all passive agents will output T via line 7. This takes $O(n \log n)$ time in expectation and with high probability by the same coupon-collector argument. □

We are now able to combine all the results from previous sections to prove theorem 3.3.1, giving guarantees on the behavior of the protocol:

THEOREM 3.3.1. *There is a nonuniform population protocol Nonuniform Majority, using $O(\log n)$ states, that stably computes majority in $O(\log n)$ stabilization time, both in expectation and with high probability.*

PROOF. We first argue that with high probability $1 - O(1/n^2)$, the protocol Nonuniform Majority stabilizes to the correct output in $O(\log n)$ time. We consider three cases based on the size of the initial gap $|g|$. For the majority cases where $|g| > 0$, we assume without loss of generality $g > 0$, so A is the majority.

If $|g| \geq 0.025|\mathcal{M}|$, then by theorem 3.6.6, we stabilize in Phase 2 to the correct output with high probability $1 - O(1/n^2)$.

If $0 < |g| < 0.025|\mathcal{M}|$, then by theorem 3.6.8, with high probability $1 - O(1/n^2)$, we end Phase 3 with at least 92% of Main agents in the set M, with the majority opinion and $\texttt{exponent} \in \{-l, -(l+1), -(l+2)\}$. Then after Phase 5 and Phase 6, all biased agents have $\texttt{exponent} \leq -l$ with high probability $1 - O(1/n^2)$ by theorem 3.6.26. Then after Phase 7 and Phase 7, there are no more minority agents with high probability $1 - O(1/n^2)$ by theorem 3.6.30. Thus in Phase 9, the majority opinion $+1$ will spread to all agents by epidemic, and we reach a stable correct configuration where all agents have $\texttt{opinions} = \{0, +1\}$ and $\texttt{output} = \mathsf{A}$.

If $g = 0$, then by theorem 3.6.7, with high probability $1 - O(1/n^2)$, we end Phase 3 with all biased agents at $\texttt{exponent} = -L$. Thus in Phase 4, we reach a stable correct configuration where all agents have $\texttt{output} = \mathsf{T}$, and there are no agents with $\texttt{exponent} > -L$ to increment the phase.

Next we justify that Nonuniform Majority stably computes majority, since it is always possible to reach a stable correct configuration. By theorem 3.6.5, we must produce at least 2 Clock agents by the end of Phase 0. Thus we must eventually advance through all timed phases 5, 6, 8, 10, 11, 12, 13 using the `counter` field. Also all agents have left the initial role $\mathsf{Role}_{\mathsf{MCR}}$, otherwise the **Init** of Phase 1 will trigger all agents to move to Phase 10 by epidemic. Thus the sum of `bias` across all Main agents is the initial gap g, and this key invariant is maintained through all phases.

Using this invariant, if the agents stabilize in Phase 2, they have a consensus on their `output`, the sign of their `bias`, and this consensus must be the sign of the sum of the biases, giving the sign of the initial gap. If the agents stabilize in Phase 4, all $|\texttt{bias}| \leq \frac{1}{2^L} \leq \frac{1}{n}$. This implies the gap $|g| \leq |\mathcal{M}| \cdot \frac{1}{n} < 1$, so the gap $g = 0$ and the agents are correctly outputting T. Finally, note that in Phase 8, when the agents set the field `full` = True, they can no longer actually store in memory the true bias they are holding. For example an agent with `opinion` $= +1$, `exponent` $= -i$, `full` = True, is actually representing the interval $\frac{1}{2^{i+1}} \leq \texttt{bias} < \frac{1}{2^i}$. Then we still have the guarantee that if we reach stable consensus in Phase 9, then this consensus is the sign of the sum of the biases and is thus correct. The final case is that we do not stabilize here, and then move to Phase 10, where they agents eventually stabilize to the correct output.

We finally justify that the expected stabilization time is $O(n \log n)$. By theorem 3.6.31, the stable backup Phase 10 will stabilize in expected $O(n \log n)$ time. Note that the high probability guarantees are all at least $1 - O(1/n^2)$, so the time for the stable backup contributes at most $O\left(\frac{n \log n}{n^2}\right) = o(1)$ to the total expected time. Now by theorem 3.6.5, with very high probability we have at least $|\mathcal{C}| \geq 0.24n$ Clock agents. In this case, the time upper bounds of theorem 3.6.15 and upper bounds on the `counter` times using standard Chernoff bound, imply that every timed phase lasts expected $O(\log n)$ time. If we do not stabilize in the untimed phases, then we also pass through by epidemic expected $O(\log n)$ time. Thus we either stabilize or reach the backup Phase 10 in expected $O(\log n)$ time. There is a final very low probability case that $|\mathsf{Clock}| = o(n)$, but we must at least have $|\mathsf{Clock}| = 2$. Even in this worst case, the time upper bounds of theorem 3.6.15 and all counter rounds are at most polynomial in n, whereas the low probability of such a small Clock population is smaller than any polynomial. Thus this event adds a negligible contribution, and we conclude the total expected stabilization time is $O(\log n)$. □

3.7. Uniform, stable protocols for majority using more states

The algorithm described in section 3.3 is *nonuniform*: the set of transitions used for a population of size n depends on the value $\lceil \log n \rceil$. A uniform protocol [37, 47, 51] is a single set of transitions that can be used in any population size. Since it is "uniformly specified", the transition function is formally defined by a linear-space[10] Turing machine, where the space bound is the maximum space needed to read and write the input and output states. The original model [10] used $O(1)$ states and transitions for all n and so was automatically uniform, but many recent $\omega(1)$ state protocols are nonuniform. With the exception of the uniform variant in [25], all $\omega(1)$ state stable majority protocols are nonuniform [2, 3, 7, 20, 24, 30]. The uniform variant in [25] has a tradeoff parameter s that, when set to $O(1)$ to minimize the states, uses $O(\log n \log\log n)$ states and $O(\log^2 n)$ time.

In this section we show that there is a way to make Nonuniform Majority in section 3.3 uniform, retaining the $O(\log n)$ time bound, but the expected number of states increases to $\Theta(\log n \log\log n)$.[11]

3.7.1. Main idea of $O(\log n \log\log n)$ state uniform majority (not handling ties). Since Nonuniform Majority uses the hard-coded value $L = \lceil \log n \rceil$, to make the algorithm uniform, we require a way to estimate $\log n$ and store it in a field L (called `logn` below) of each agent. For correctness and speed, it is only required that `logn` be within a constant multiplicative factor of $\log n$.

GKasieniec and Stachowiak [60] show a uniform $O(\log\log n)$ state, $O(\log n)$ time protocol (both bounds in expectation and with high probability) that computes and stores in each agent a value $\ell \in \mathbb{N}^+$ that, with high probability, is within additive constant $O(1)$ of $\lceil \log\log n \rceil$ (in particular, WHP $\ell \geq \lfloor \log\log n \rfloor - 3$ [25, Lemma 8]), so $2^\ell = \Theta(\log n)$. (This is the so-called *junta election* protocol used as a subroutine for a subsequent leader election protocol.) Furthermore, agents approach this estimate from below, propagating the maximum estimate by epidemic $\ell', \ell \to \ell, \ell$ if $\ell' < \ell$. This gives an elegant way to compose the size estimation with a subsequent nonuniform protocol $\mathcal{P}$ that requires the estimate: agents store their estimate `logn` of $\log n$ and use it in $\mathcal{P}$. Whenever an agent's estimate `logn` updates—always getting larger—it simply resets $\mathcal{P}$, i.e., sets

[10] That is, the Turing machine is allowed to use a bit more than the space necessary simply to read and write the input and output states, but not significantly more (constant-factor). This allows it to do simple operations, such as integer multiplication, that require more than constant space, without "cheating" by allowing the internal memory usage of the Turing machine to vastly exceed that required to represent the states.

[11] We say "expected" because this protocol has a non-zero probability of using an arbitrarily large number of states. The number of states will be $O(\log n \log\log n)$ in expectation and with high probability.

the entire state of $\mathcal{P}$ to its initial state. We can then reason as though all agents actually started with their final convergent value of `logn`.[12]

To make our protocol uniform, but remove its correctness in the case of a tie, as we explain below, all agents conduct this size estimation, stored in the field `logn`, in parallel with the majority protocol $\mathcal{P}$ of section 3.3. Each agent resets $\mathcal{P}$ to its initial state whenever `logn` updates. This gives the stated $O(\log n)$ time bound and $O(\log n \log\log n)$ state bound. Note that in Phase 0, agents count from $\texttt{counter} = \Theta(\log n)$ down to 0. It is sufficient to set the constant in the Θ sufficiently large that all agents with high probability receive by epidemic the convergent final value of `logn` significantly before any agent with the same convergent estimate counts down to 0.

Acknowledging that, with small probability, the estimate of $\log n$ could be too low for Phase 4 to be correct, we simply remove Phase 4 and do not attempt to detect ties. So if we permit undefined behavior in the case of a tie (as many existing fast majority protocols do), then this modification of the algorithm otherwise retains stably correct, $O(\log n)$ time behavior, while increasing the state complexity to $O(\log n \log\log n)$.

3.7.2. How to stably compute ties. With low but positive probability, the estimate of $\log n$ could be too small. For most phases of the algorithm, this would merely amplify the probability of error events (e.g., Phase 1 doesn't last long enough for agents to converge on biases $\{-1, 0, +1\}$) that later phases are designed to handle. However, the correctness of Phase 4 (which detects ties) requires agents to have split through at least $\log n$ exponents in Phase 3. Since the population-wide bias doubles each time the whole population splits down one exponent, the only way for the whole population to split through $\log n$ exponents is for there to be a tie (i.e., the population-wide bias is 0, so can double unboundedly many times). In this one part of the algorithm, for correctness we require the estimate to be at least $\log n$ with probability 1. (It can be much greater than $\log n$ without affecting correctness; an overestimate merely slows down the algorithm.)

To correct this error, we will introduce a stable backup size estimate, to be done in Phase 4. Note that there are only a constant number of states with $\texttt{phase} = 4$: Clock agents do not store a counter in this phase, and Main agents that stay in this phase must have $\texttt{bias} \in \left\{0, \pm\frac{1}{2^L}\right\}$. Thus

[12]One might hope for a stronger form of composition, in which the size estimation *terminates*, i.e., sets an initially False Boolean flag to True only if the size estimation has converged, in order to simply prohibit the downstream protocol $\mathcal{P}$ from starting with an incorrect estimate of $\log n$. However, when both states A and B are initially $\Omega(n)$, this turns out to be impossible; $\Omega(n)$ agents will necessarily set the flag to True in $O(1)$ time, long before the $O(\log n)$ time required for the size estimation to converge [47, Theorem 4.1].

we can use an additional $\Theta(\log n)$ states for the agents that are currently in $\texttt{phase} = 4$ to stably estimate the population size. If they detect that their estimate of L was too small, they simply go to the stable backup Phase 10.

Stable computation of $\lfloor \log n \rfloor$. The stable computation of $\log n$ has all agents start in state L_0, where the subscript represents the agent's estimate of $\lfloor \log n \rfloor$. We have the following transitions: for each $i \in \mathbb{N}$, $L_i, L_i \to L_{i+1}, F_{i+1}$ and, for each $0 \leq j < i$, $F_i, F_j \to F_i, F_i$. Among the agents in state L_i, half make it to state L_{i+1}, reaching a maximum of L_k at $k = \lfloor \log_2 n \rfloor$.[13] All remaining F agents receive the maximum value k by epidemic. A very similar protocol was analyzed in [**28**, Lemma 12]. We give a quick analysis below for the sake of self-containment.

Time analysis of stable computation of $\lfloor \log n \rfloor$. The expected time is $\Theta(n \log n)$. For the upper bound, for each i, if j is the count of L_i agents, then the probability of a $L_i, L_i \to \ldots$ reaction is $O(j^2/n^2)$, so for any possible starting count k of L_i agents, it takes expected time at most $O\left(\frac{1}{n}\sum_{j=1}^{k} \frac{n^2}{j^2}\right) = O(n)$ for the count of L_i agents to get from k to 0 or 1. Assuming in the worst case that no $L_{i+1}, L_{i+1} \to \ldots$ reaction happens until all $L_i, L_i \to \ldots$ reactions complete, then summing over $\lfloor \log n \rfloor$ values of i gives the $O(n \log n)$ time upper bound to converge on one L_k agent, where $k = \lfloor \log n \rfloor$. It takes additional $O(\log n)$ time for the F agents to propagate k by epidemic.

For the lower bound, observe that for each value of i, the *last* reaction $L_i, L_i \to \ldots$ occurs when the count of L_i is either 2 (and will increase at most once more), or 3 (and will never increase again). This takes expected time $\Theta(n)$. Since the count of L_i will increase at most once more, at that time (just before the last $L_i, L_i \to \ldots$ reaction), the count of L_{i-1} is at most 3, otherwise *two* more L_i's could be produced, and this would not be the last $L_i, L_i \to \ldots$ reaction. Thus, to produce the L_i needed for this last $L_i, L_i \to \ldots$ reaction, the previous $L_{i-1}, L_{i-1} \to L_i, F_i$ reaction occurs when the count of L_{i-1} is at most 5, also taking $\Theta(n)$ time. Thus the final reaction at each level takes time $\Theta(n)$ and depends on a reaction at the previous level that also takes time $\Theta(n)$, so summing across $\lfloor \log n \rfloor$ levels gives $\Omega(n \log n)$ completion time.

[13] More generally, the unique stable configuration encodes the full population size n in binary in the following distributed way: for each position i of a 1 in the binary expansion of n, there is one agent L_i. Thus, these remaining agents lack the space to participate in propagating the value $k = \lfloor \log n \rfloor$ by epidemic, but there are $\Omega(n)$ followers F to complete the epidemic quickly.

It is shown in [28, Lemma 12] that the time is $O(n \log^2 n)$ with high probability, i.e., slower than the expected time by a $\log n$ factor. Since the probability is $O(1/n^2)$ that we need to rely on this slow backup, even this larger time bound contributes negligibly to our total expected time.

3.7.3. Challenges in creating $O(\log n)$ state uniform algorithm. It is worth discussing some ideas for adjusting the uniform protocol described above to attempt to reduce its space complexity to $O(\log n)$ states. The primary challenge is to enable the population size n to be estimated without storing the estimate in any agent that participates in the main algorithm (i.e., the agent has role Main, Clock, or Reserve). If agents in the main algorithm do not store the size, then by [47, Theorem 4.1] they will provably go haywire initially, with agents in every phase, totally unsynchronized, and require the size estimating agents to reset them after having converged on a size estimate that is $\Omega(\log n)$.

The following method would let the Size agents reset main algorithm agents, without actually having to store an estimate of $\log n$ in the algorithm agents, but it only works with high probability. The size estimating agents could start a junta-driven clock as in [60], which is reset whenever they update their size estimate. Then, as long as there are $\Omega(n)$ Size agents, they could for a phase timed to last $\Theta(\log n)$ time, reset the algorithm agents by *direct* communication (instead of by epidemic). This could put the algorithm agents in a quiescent state where they do not interact with each other, but merely wait for the Size agents to exit the resetting phase, indicating that the algorithm agents are able to start interacting again. Since there are $\Omega(n)$ size-estimating agents, each non-size-estimating agent will encounter at least one of them with high probability in $O(\log n)$ time.

The problem is that the reset signal is not guaranteed to reach every algorithm agent. There is some small chance that a Main agent with a bias different from its `input` does not encounter a Size agent in the resetting phase, so is never reset. The algorithm from that point on could reach an incorrect result when the agent interacts with properly reset agents, since the sum of biases across the population has changed. In our algorithm, by "labeling" each reset with the value `logn`, we ensure that no matter what states the algorithm agents find themselves in during the initial chaos before size computation converges, every one of them is guaranteed to be reset one last time with the same value of `logn`. The high-probability resetting described above seems like a strategy that

could work to create a high probability uniform protocol using $O(\log n)$ time and states, though we have not thoroughly explored the possibility.

But it seems difficult to achieve probability-1 correctness using the technique of "reset the whole majority algorithm whenever the size estimate updates," without multiplying the state complexity by the number of possible values of `logn`. Since we did not need $\lfloor \log n \rfloor$ exactly, but only a value that is $\Theta(\log n)$, we paid only $\Theta(\log \log n)$ multiplicative overhead for the size estimate, but it's not straightforward to see how to avoid this using the resetting technique. Of course, one could imagine that the savings could come from reducing the state complexity of the main majority-computing agents in the nonuniform algorithm. However, reducing the nonuniform algorithm's state complexity to below the $\Omega(\log n)$ lower bound of [3] would provably require the algorithm to be not monotonic or not output dominant. (See section 3.8 for a discussion of those concepts.) Another possible approach is to intertwine more carefully the logic of the majority algorithm with the size estimation, to more gracefully handle size estimate updates without needing to reset the entire majority algorithm.

3.8. Conclusion and open problems

There are two major open problems remaining concerning the majority problem for population protocols.

Uniform $O(\log n)$-time, $O(\log n)$-state majority protocol. Our main $O(\log n)$ state protocol, described in section 3.3, is *nonuniform*: all agents have the value $\lceil \log n \rceil$ encoded in their transition function. The uniform version of our protocol described in section 3.7 uses $O(\log n \log \log n)$ states. It remains open to find a uniform protocol that uses $O(\log n)$ time and states.

Unconditional $\Omega(\log n)$ state lower bound for stable majority protocols. The lower bound of $\Omega(\log n)$ states for (roughly) sublinear time majority protocols shown by Alistair, Aspnes, and Gelashvili [3] applies only to stable protocols satisfying two conditions: *monotonicity* and *output dominance*.

Recall that a *uniform* protocol is one where a single set of transitions works for all population sizes; nonuniform protocols typically violate this by having an estimate of the population size (e.g., the integer $\lceil \log n \rceil$) embedded in the transition function. Monotonicity is a much weaker form of uniformity satisfied by nearly all known nonuniform protocols. While allowing different transitions

as the population size grows, monotonicity requires that the transitions used for population size n must also be correct for all smaller population sizes $n' < n$ (i.e., an *overestimate* of the size cannot hurt), and furthermore that the transitions be no slower on populations of size n' than on populations of size n (though the transitions designed for size n may be slower on size n' than the transitions intended for size n'). Typically the nonuniform estimate of $\log n$ is used by the protocol to synchronize phases taking time $\approx T = \Theta(\log n)$, by having each agent individually count from T down to 0 (a so-called "leaderless phase clock", our Standard Counter Subroutine). $\Theta(\log n)$ is the time required for an epidemic to reach the whole population and communicate some message to all agents before the phase ends. Most errors in such protocols are the result of some agent not receiving a message before a phase ends, i.e., the clock runs atypically faster than the epidemic. If the size estimate is significantly *larger* than $\log n$, this slows the protocol down, but only increases the probability that all agents receive intended epidemic messages before a phase ends; thus such protocols are monotone.

In our nonuniform protocol Nonuniform Majority, which gives each agent the value $L = \lceil \log n \rceil$, giving a different value of L does not disrupt the stability (only the speed) of the protocol, with one exception: Phase 3 must go for at least $\log n$ exponents, or else (i.e., if L is an underestimate) Phase 4 could incorrectly report a tie. If we consider running Phase 3 on a smaller population size n' than the size n for which it was designed, $L = \lceil \log n \rceil$ of $\lceil \log n' \rceil$ will be an *overestimate*, which does not disrupt correctness. Furthermore, since Clock agents are counting to L in each case, they are just as fast on population size n' as on size n. This implies that Phase 3, thus the whole protocol Nonuniform Majority, is monotone.

Output dominance references the concept of a *stable* configuration $\mathbf{c}$, in which all agents have a consensus opinion that cannot change in any configuration subsequently reachable from $\mathbf{c}$. A protocol is *output dominant* if in any stable configuration $\mathbf{c}$, adding more agents with states already present in $\mathbf{c}$ maintains the property that every reachable stable configuration has the same output (though it may disrupt the stability of $\mathbf{c}$ itself). This condition holds for all known stable majority protocols, including that described in this chapter, because they obey the stronger condition that adding states already present in $\mathbf{c}$ does not even disrupt its stability. Such protocols are based on the idea that two agents with the same opinion cannot create the opposite opinion, so stabilization happens exactly when all agents first converge on a consensus output.

To see that our protocol is output dominant, define a configuration to be *silent* if no transition is applicable (i.e., all pairs of agents have a null interaction); clearly a silent configuration is also stable. Although the definition of stable computation allows non-null transitions to continue happening in a stable configuration, many existing stable protocols have the stronger property that they reach a silent configuration with probability 1, including our protocol. It is straightforward to see that any silent configuration has the property required for output dominance, since if no pair of states in a configuration can interact nontrivially, their counts can be increased while maintaining this property. (One must rule out the special case of a state with count 1 that can interact with another copy of itself, which does not occur in our protocol's stable configurations.)

Monotonicity can be seen as a natural condition that all "reasonable" non-uniform protocols must satisfy, but output dominance arose as an artifact that was required for the lower bound proof strategy of [3]. It remains open to prove an unconditional (i.e., removing the condition of output dominance) lower bound of $\Omega(\log n)$ states for *any* stable monotone majority protocol taking polylogarithmic time, or to show a stable polylogarithmic time monotone majority protocol using $o(\log n)$ states, which necessarily violates output dominance. If the unconditional lower bound holds, then our protocol is simultaneously optimal for both time and states. Otherwise, it may be possible to use $o(\log n)$ states to stably compute majority in polylogarithmic stabilization time with a non-output-dominant protocol. In this case, there may be an algorithm simultaneously optimal for both time and states, or there may be a tradeoff.

$O(\log n)$ time, $o(\log n)$ state non-stable protocol. Berenbrink, Elsässer, Friedetzky, Kaaser, Kling, and Radzik [25] showed a non-stable majority protocol (i.e., it has a positive probability of error) using $O(\log \log n)$ states and converging in $O(\log^2 n)$ time. (See section 2.2 for more details.) Is there a protocol with $o(\log n)$ states solving majority in $O(\log n)$ time?

We close with questions unrelated to majority.

Fast population protocol for parity. The majority problem "$X_1 > X_2$?" is a special case of a *threshold* predicate, which asks whether a particular weighted sum of inputs $\sum_{i=1}^{k} w_i X_i > c$ exceeds a constant c. (For majority, $w_1 = 1, w_2 = -1, c = 0$; our protocol extends straightforwardly to other values of w_i, though not other values of c.) The threshold predicates, together with the *mod* predicates, characterize the *semilinear* predicates, which are precisely the predicates stably computable by $O(1)$ state protocols [11] with no time constraints (though $\Theta(n)$ time to stabilize

is sufficient for all semilinear predicates [12] and necessary for "most" [18]). A representative mod predicate is *parity*: asking whether an odd or even number of agents exist. (More generally, asking the parity of the number of agents with initial opinion A.) Like majority, parity is solvable by a simple protocol in $O(n)$ time,[14] and it is known to require $\Omega(n)$ time for any $O(1)$ state protocol to stabilize [18]. Techniques from [2] can be used to show that stabilization requires close to linear time even allowing up to $\frac{1}{2}\log\log n$ states. An interesting open question is to consider allowing $\omega(1)$ states in deciding parity. Can it then be decided in polylogarithmic time? Is there a parity protocol simultaneously optimal for both polylogarithmic time and states, such as the $O(\log n)$ time, $O(\log\log n)$ state protocol for leader election [26]? Or is there a tradeoff?

Fast population protocols for function computation. The transition $X, Q \to Y, Y$, starting with sufficiently many excess agents in state Q, computes the function $f(x) = 2x$, because if we start with x agents in state X, eventually $2x$ agents are in state Y, taking time $O(\log n)$ to stabilize [38]. The similar transition $X, X \to Y, Q$ computes $f(x) = \lfloor x/2 \rfloor$, but it takes time $\Theta(n)$ to stabilize, as does any $O(1)$-state protocol computing any linear function with a coefficient not in $\mathbb{N}$, as well as "most" non-linear functions such as $\min(x_1, x_2)$ (computable by $X_1, X_2 \to Y, Q$) and $\max(x_1, x_2)$ [18]. Can such functions be computed in sublinear time by using $\omega(1)$ states?

[14] Agents start in state L_1, undergo reactions $L_i, L_j \to L_{(i+j) \mod 2}, F_{(i+j) \mod 2}$; $L_i, F_j \to L_i, F_i$; $F_i, F_j \to F_i, F_i$, i.e., F agents adopt the parity of L agents, and two F agents with different parities adopt that of the sender. The first two reactions would take time $O(n \log n)$ for the last remaining L_i to update all F agents directly; the last reaction reduces this time to $O(n)$ [12, Theorem 5].

CHAPTER 4

Exact Size Counting in Population Protocols

4.1. Introduction

The limitation on time-efficient computation with constant-memory protocols motivates the study of population protocols with memory that can grow with n. A recent blitz of impressive results has shown that leader election [**2**, **3**, **5**, **20**, **24**, **25**, **26**, **27**, **30**, **60**, **61**, **79**, **91**, **91**] and exact majority [**3**, **5**, **7**, **20**, **24**, **25**] can be solved in polylog(n) time using polylog(n) states leading to time and space optimal solutions for both problems [**26**, **50**]. Notably, most of these protocols requires an approximate estimate of the population size n to be encoded into each agent (commonly a constant-factor upper bound on $\lfloor \log n \rfloor$ or $\lfloor \log \log n \rfloor$).

Although, there is a uniform leader election [**60**] in the literature, the reliance of many existed protocols on storing the population size in advance, *partially* motivates our study of the *size counting problem* of computing the population size n. The problem is clearly solvable by an $O(n)$ time protocol using a straightforward leader election: Agents initially assume they are leaders and the count is 1. When two leaders meet, one agent sums their counts while the other becomes a follower, and followers propagate by epidemic the maximum count. No faster protocol was previously known.

Our study is further motivated by the desire to understand the power of *uniform* computation in population protocols. In which the same transition algorithm is used for all populations, though the number of states may vary with the population size (formalized with Turing machines; see section 1.2.) A uniform protocol can be deployed into *any* population without knowing in advance the size, or even a rough estimate of the size. The original, $O(1)$-state model [**10**, **11**, **12**], is uniform since there is a single transition function. Because we allow memory to grow with n, our model's power exceeds that of the original, but is strictly less than that of the nonuniform model of most papers using $\omega(1)$ states.

The formal model is given in section 1.2. Here we sketch the relevant intuition. The agents' transition algorithm is modeled as a Turing machine. When two agents interact they are both

in an initial configuration of the Turing machine, their input is the tape content in the halting configuration of the last time each of them had an interaction. (We assume the Turing machine state and tape head positions are not transmitted, since they can be assumed WLOG identical in every halting configuration.) The space usage (in bits) s is defined as normal for Turing machines: the maximum number of tape cells that are written during the computation. The number of states is then 2^s, where s is the maximum space usage of any agent during an execution of the protocol. For ease of understanding, we will use standard population protocol terminology and not refer explicitly to details of the Turing machine definition except where needed. Therefore a *state* $s \in \Lambda$ always refers to the TM tape content of an agent (leaving out TM state and tape head positions since these are identical in all initial configurations), where Λ is the set of all staets, a *configuration* $\mathbf{c} \in \mathbb{N}^\Lambda$ is a vector indexed by a state, where $\mathbf{c}(s)$ is the *count* of state s in the population.

We furthermore assume that each agent has access to independent uniformly random bits, assumed to be pre-written on a special read-only tape (this allows the Turing machine to be deterministic even though it is computing a nondeterministic relation). This is different from the traditional definition of population protocols, which assumes a deterministic transition function. In our case, we have a transition *relation* $\delta \subseteq \Lambda^4$. Several papers [2, 27] indicate how to use the randomness built into the interaction scheduler to provide nearly uniform random bits to the agents, using various *synthetic coin* techniques, showing that the deterministic model can effectively simulate the randomized model. In the interest of brevity and simplicity of presentation, we will simply assume in the model that each agent has access to a source of uniformly random bits.

Recall that n denotes the number of agents in the population. Repeatedly, a pair of agents is selected uniformly at random to interact, where they run the transition algorithm on the pair of states they were in prior to the interaction, and storing the output states for their next interactions. All references to "time" in this theis refer to parallel time, i.e., the number of interactions divided by n. An *execution* is a sequence of configurations $\mathbf{c}_0, \mathbf{c}_1, \ldots$ such that for all i, applying a transition to $\mathbf{c}_i$ results in $\mathbf{c}_{i+1}$. $\log n$ is the base-2 logarithm of n, and $\ln n$ is the base-e logarithm of n.

4.1.1. Contribution. Our main result is a uniform protocol that, with probability $\geq 1 - O(\frac{\log \log n}{n})$, counts the population size, converging in $6 \ln n \log \log n$ time using $2^{17} n^{60}$ states ($17 +$

$60 \log n$ bits), without an initial leader (all agents have an initially identical state). The protocol is *stabilizing*: the output is correct with probability 1, converging in expected time $7 \ln n \log \log n$.[1]

A key subprotocol performs leader election in time $O(\ln n \log \log n)$ with high probability and in expectation. It uses $O(n^{18})$ states, much more than the $O(\log \log n)$ known to be necessary [2] and sufficient [60] for sublinear time leader election. However, it is uniform, unlike many known sublinear-time leader election protocols [2, 27, 30]. It repeatedly increases the length of a binary string each agent stores, where the protocol, with probability at least $1 - O(1/n)$, takes $O(\log n)$ time once the length of this string reaches $\approx \log n$. The length increases in stages that each take $O(\log n)$ time.

In our main protocol, the agents doubles the string length each stage, so takes $\log \log n$ stages (hence $O(\log n \log \log n)$ time) to reach length $\log n$.

The protocol generalizes straightforwardly to trade off time and memory: by adjusting the rate at which the string length grows, the convergence time $t(n)$ is $O(f(n) \log n)$, where $f(n)$ is the number of stages required for the string length to reach $\log n$. For example, if the code length increments by 1 each stage, then $f(n) = \log n$, so $t(n) = \log^2 n$. In this case the state complexity would be $O(n^{30})$ for the full protocol and $O(n^9)$ for just the leader election portion. (See section 4.4.2.)

By squaring the string length each stage, $t(n) = \log n \log \log \log n$. By exponentiating the string length, $t(n) = \log n \log^* n$.

Even slower-growing $f(n)$ such as inverse Ackermann are achievable. However, the faster the string length grows each stage, the more it potentially overshoots $\log n$, increasing the space requirements.

For example, for $t(n) = \log n \log \log \log n$ by repeated squaring, the worst-case string length is $\log^2 n$, meaning $2^{O(\log^2 n)} = n^{O(\log n)}$ states. Multiplying length by a constant gives the fastest increase that maintains a polynomial number of states.

4.2. Exact population size counting protocol

In this section we prove our main theorem:

[1] The time to reach a stable configuration, from which the output *cannot* change, is $\Omega(n)$ for our protocol. We leave open the question of a protocol that reaches a stable configuration in sublinear time.

THEOREM 4.2.1. *There is a leaderless, uniform population protocol that stably solves the exact size counting problem. With probability at least* $1 - \frac{10+5\log\log n}{n}$*, the convergence time is at most* $6 \ln n \log\log n$*, and each agent uses* $17 + 60 \log n$ *bits of memory. The expected time to convergence is at most* $7 \ln n \log\log n$*.*

The stabilization time can be much larger, up to $O(n)$. (See section 4.4.1.) theorem 4.2.1 follows from theorem 4.4.3 and theorem 4.4.4, which respectively cover the "with high probability" and "stabilization and expected time" parts of theorem 4.2.1.

Protocol 16 ExactCounting(rec, sen)

```
▷ state: strings C (code), LC (leader code), Bool isLeader, ints M, ave, count, phase
▷ initial state of agent: C = LC = ε, isLeader = True, M = ave = count = phase = 1
UniqueID(rec, sen)
ElectLeader(rec, sen)
if rec.LC = sen.LC then                          ▷ separate restarts under different leaders
    DiscreteAveraging(rec, sen)
    timer(rec, sen)
```

The protocol is ExactCounting. There are four main subprotocols: UniqueID, ElectLeader, `DiscreteAveraging`, and `timer`, each discussed in detail in later subsections. ExactCounting runs in parallel on all agents, but within an agent, each subprotocol runs sequentially (for correctness each subprotocol must run in the given order). Most state updates use one-way rules for selected agents sen (*sender*) and rec (*receiver*). The only rule that is not one-way is `DiscreteAveraging`, in which both sender and receiver update their state. In all other cases, only the receiver potentially updates the state.

High-level overview of ExactCounting protocol. UniqueID eventually assigns to every agent a unique id, represented as a binary string called a *code* `C`. UniqueID requires $\Omega(n)$ time to converge, but it does not need to converge before it can be used by the other subprotocols. In fact, in other subprotocols, agents do not use each others' codes directly. Agents also have a longer code called a *leader code* `LC`, such that $2|\texttt{C}| = |\texttt{LC}|$ and, for any candidate leader, `C` is a prefix of `LC`. ElectLeader elects a leader by selecting the agent whose leader code is lexicographically largest. The code length $|\texttt{C}|$ will eventually be at least length $\log n$, so $|\texttt{C}|$ can be used to estimate an upper bound `M` on the value $3n^3$ to within a polynomial factor. `DiscreteAveraging` uses `M` in a leader-driven protocol that counts the population size exactly, which is correct so

long as $\texttt{M} \geq 3n^3$, by using a fast averaging protocol similar to the one studied by Mocquard et al. [79]. `DiscreteAveraging` must be restarted by the upstream UniqueID subprotocol many times, and in fact will be restarted beyond the $O(\log n \log\log n)$ time bound we seek. However, within $O(\log n \log\log n)$ time, `DiscreteAveraging` will converge to the correct population size. Subsequent restarts of `DiscreteAveraging` will re-converge to the correct output, but prior to convergence will have an incorrect output. `timer` is used to detect when `DiscreteAveraging` has likely converged, waiting to write output into the `count` field of the agent. This ensures that after the correct value is written, on subsequent restarts of `DiscreteAveraging`, the incorrect values that exist before `DiscreteAveraging` re-converges will not overwrite the correct value recorded during the earlier restart.

4.2.1. UniqueID. We assume that the two following subroutines are available:

For $x, y \in \{0,1\}^*$, Append(x, y) returns xy, and for $m \in \mathbb{Z}^+$, RandBits(m) returns a random string in $\{0,1\}^m$.

Subprotocol 17 UniqueID(rec, sen)

if |rec.C| < |sen.C| **then** ▷ If receiver's code shorter than sender's, make same length.
  ExtendCode(rec, |sen.C| − |rec.C|))
if rec.C = sen.C **then** ▷ If codes are the same, double the length.
  ExtendCode(rec, max(1, |rec.C|))

Subroutine 18 ExtendCode(rec, numBits)

if rec.`isLeader` **then** ▷ extend LC by twice numBits; take new C bits from LC
  newLC ← Append(rec.LC, RandBits(2 · numBits))
  SetNewLeaderCode(rec, newLC) ▷ described in section 4.2.2
  rec.C ← Append(rec.C, newLC[(|rec.C| + 1) .. (|rec.C| + numBits)])
else
  rec.C ← Append(rec.C, RandBits(numBits))

UniqueID can be viewed as traversing a labeled binary tree, until all agents reach a node unoccupied by any other agent. We say the *level* is the maximum depth (longest code length) of any agent in the population. Initiating a new level happens when two agents with the same code interact. The receiver doubles the length of its code with uniformly random bits, going twice as deep in the tree. To ensure each agent reaches the new level quickly, agents at deeper levels recruit other agents to that level by epidemic, which generate random bits to reach the same code length.

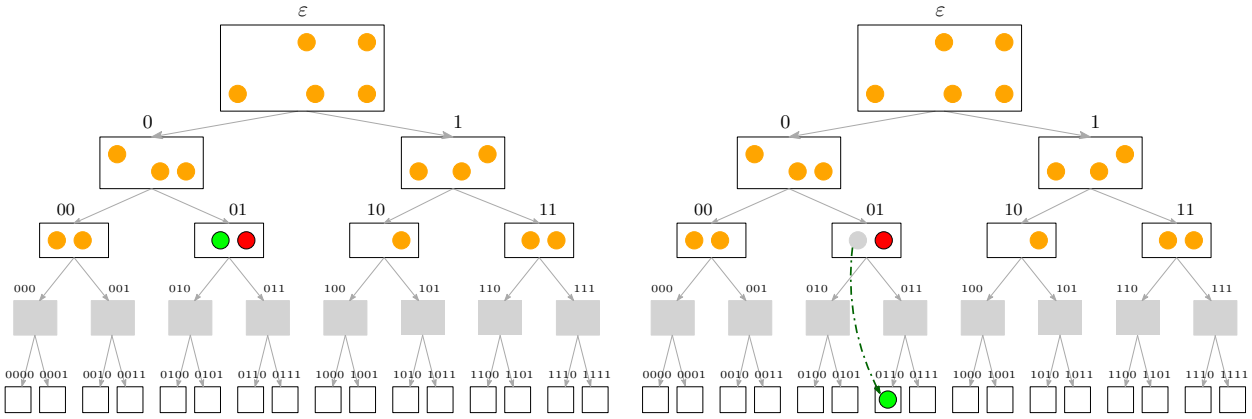
Subroutine 19 SetNewLeaderCode(rec, newLC)

```
rec.LC ← newLC
▷ restart Timer and Average protocols whenever LC changes.
rec.phase ← 1
rec.M ← 3 · 2^{3|rec.C|}
if rec.isLeader then
    rec.ave ← rec.M
else
    rec.ave ← 0
```

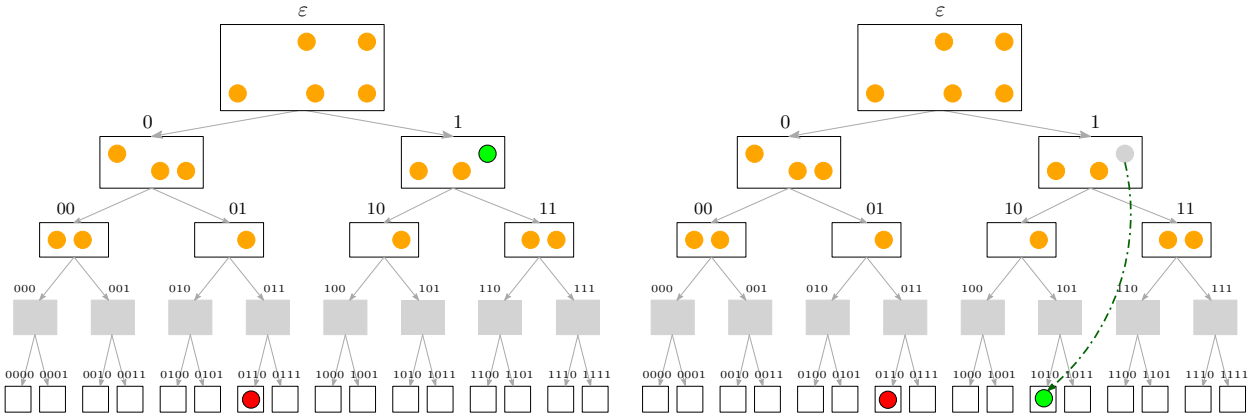
The key property of this protocol is that, in any level $\ell < \log n$, only $O(\log n)$ time is required to increase the level. Since we double the level when it increases, $\log \log n$ such doublings are required to reach level $\geq \log n$, so $O(\log n \log \log n)$ time. theorem 4.2.2 formalizes this claim, explaining how the length-increasing schedule can be adjusted to achieve a trade-off between time and memory.

Number of reachable configurations. Since all codes are generated randomly, the number of reachable configurations is infinite. This choice is merely to simplify analysis, allowing us to assume that all agents at a level have uniformly random codes. However, if a finite number of reachable configurations is desired (so that, for instance, the definition of stabilization we use is equivalent to definitions based on reachability), it is possible to modify UniqueID so that when two agents with the same code meet, they *both* append bits that are guaranteed to be different. The protocol still works in this case and in fact takes strictly less expected time for the codes to become unique. Viewing two agents with compatible codes (i.e., one code is a prefix of the other) as equivalent, each new level increases the number of equivalence classes by 1. Thus it is guaranteed that all agents will converge on unique codes of length at most $n - 1$, implying the reachable configuration space is finite.

The next lemma bounds the time for UniqueID to reach level at least $\log n$, assuming a generalized way of increasing the level, defining $f(n)$ to be the number of times the level must increase before reaching at least level $\log n$. Afterwards we state a corollary for our protocol, which doubles the level whenever it increases, so $f(n) = \log \log n$. By using this lemma with different choices of f, one can obtain a tradeoff between time and space; if the level increases more (corresponding to a slower-growing f) this takes less time to reach level at least $\log n$, but may overshoot $\log n$ and use more space.



(a) When two agents in the same node at level i interact, receiver moves to a random descendant at level $2i$.



(b) When sender is in a deeper level of the tree, receiver moves to a random descendant in its own subtree at the sender's level.

FIGURE 4.1. Agents moving through the binary tree (i.e., choosing binary codes) in accordance with the UNIQUEID subprotocol.

Intuitively, the lemma is proven by observing that the worst case is that the current level is $\log(n)-1$. It takes $O(\log n)$ time for all agents not yet at that level to reach it by epidemic. At that point the worst case is that codes are distributed to maximize expected time: exactly $n/2$ codes each shared by two agents. Then the expected time is constant for the first interaction between two such agents, starting the next level. Thus it takes time $O(\log n)$ to increase the level, hence $O(f(n)\log n)$ time for the level to increase from 0 to at least $\log n$.

LEMMA 4.2.2. *For all $n \in \mathbb{N}$, define $f(n)$ to be the number of times* UNIQUEID *must increase the level (last line of* UNIQUEID*) to reach level at least $\log n$. For all $\alpha > 0$, in time $5\alpha f(n) \ln n$, all agents reach level at least $\log n$ with probability at least $1 - 5f(n)n^{-\alpha}$.*

PROOF. Imagine an alternate process where at each level agents wait until all other agents also reach the same level before enabling transitions that start the next level (where two agents with the same code meet and the receiver will double its code length). The time for such a process stochastically dominates the time for our protocol, so we can use its time as an upper bound for our protocol. It suffices to show that, when all agents are at the same level, it takes constant time to start the next level. After initializing a level, by theorem 1.3.4, the new code length will spread by epidemic in time $\alpha_u \ln n$ with probability at least $1 - 4n^{-4\alpha_u+1}$.

Assume all agents are currently at level i. Denote by S_j the number of agents in node j of the tree at level i (i.e., they have the j'th code in $\{0,1\}^i$ in lexicographic order). The probability that the next interaction is between two agents at the same node (having equal codes) is minimized when $S_j = S_{j'} = n/2^i$ for all $1 \le j, j' \le 2^i$. Then for all $0 \le i < \log n$, if the current level is i,

$$\begin{aligned}
\Pr\left[\text{next interaction initializes new level}\right] &= \frac{\sum_{j=1}^{2^i} \binom{S_j}{2}}{\binom{n}{2}} = \frac{\sum_{j=1}^{2^i} \binom{n/2^i}{2}}{\binom{n}{2}} \\
&= \frac{2^i(n/2^i)(n/2^i-1)}{n(n-1)} = \frac{n/2^i - 1}{n-1} \\
&\ge \frac{n/2^{\log(n)-1} - 1}{n-1} = \frac{1}{n-1} > \frac{1}{n}.
\end{aligned}$$

Therefore, the expected number of interactions to start a new level is $\le n$, equivalently parallel time 1. This is a geometric random variable with success probability at least $\frac{1}{n}$. Then at any level $i < \log n$, for any $\alpha'_u > 0$,

$$\begin{aligned}
\Pr\left[\text{initializing next level take more than } \alpha'_u n \ln n \text{ interactions}\right] &= \left(1 - \frac{1}{n}\right)^{\alpha'_u n \ln n} \\
&< e^{-\alpha'_u \ln n} = n^{-\alpha'_u}.
\end{aligned}$$

By theorem 1.3.4, for any $\alpha_u > 0$, more than $\alpha_u \ln n$ time is required for all agents to reach this level by epidemic with probability at most $4n^{-\alpha_u/4+1}$. By the union bound over this event and the event *"once all agents are at a level, it takes more than time $\alpha'_u \ln n$ to start a new level"* (shown above to happen with probability at most $n^{-\alpha'_u}$), the time spent at each level is more than $\alpha_u \ln n + \alpha'_u \ln n = (\alpha_u + \alpha'_u) \ln n$ with probability at most $4n^{-\alpha_u/4+1} + n^{-\alpha'_u}$. Given $\alpha > 0$, let $\alpha'_u = \alpha$ and $\alpha_u = 4(\alpha+1)$. Then this probability bound is $4n^{-\alpha_u/4+1} + n^{-\alpha'_u} = 5n^{-\alpha}$.

By the union bound over all $f(n)$ levels visited in the tree, it takes more than time $f(n)(\alpha \ln n + 4\alpha \ln n) = 5\alpha f(n) \ln n$ time to reach level at least $\log n$ with probability at most $5f(n)n^{-\alpha}$. □

The next corollary is specific to our level-doubling schedule, used throughout the rest of the chapter, corresponding to $f(n) = \log \log n$ in theorem 4.2.2.

COROLLARY 4.2.3. *In the* UNIQUEID *protocol, for all* $\alpha > 0$*, in* $5\alpha \ln n \log \log n$ *time all agents reach level at least* $\log n$ *with probability at least* $1 - \frac{5 \log \log n}{n^\alpha}$.

The previous results show UNIQUEID quickly gets to level $\log n$. The next lemma states that it does not go too far past $\log n$. Intuitively, if the level is $2 \log n$, there are n^2 possible codes chosen uniformly at random among n agents, a standard birthday problem with probability $\frac{1}{e}$ of a collision, which drops off polynomially with the level beyond $2 \log n$.

LEMMA 4.2.4. *Let* $\epsilon > 0$*. If the current level is* $(2+\epsilon) \log n$*, all codes are unique with probability at least* $1 - \frac{1}{n^\epsilon}$.

PROOF. Consider the agents in order for agent $1, 2, \ldots$. The code of agent i collides with the code of some agent $1, 2, \ldots, i-1$ with probability $\frac{i-1}{c}$, where $c = 2^{(2+\epsilon) \log n} = n^{2+\epsilon}$ is the number of available codes. Then by the union bound,

$$\Pr[\text{at least one collision}] \leq \sum_{i=1}^{n} \frac{i-1}{c} = \frac{n(n-1)}{2c} < \frac{n^2}{n^{2+\epsilon}} = \frac{1}{n^\epsilon}.$$

□

However, since the code length doubles when it changes, not all values of ϵ in theorem 4.2.4 correspond to a level actually visited. It could overshoot by factor two, giving the following.

COROLLARY 4.2.5. *Let* $\epsilon > 0$*. The eventual code length of each agent is* $< (4 + 2\epsilon) \log n$ *with probability at least* $1 - \frac{1}{n^\epsilon}$.

PROOF. Recall that a new level of the tree is initiated when two agents with the same code interact. Since the level is doubled in this case, the code lengths exceed $(4 + 2\epsilon) \log n$ if there was a duplicate code at the power-of-two level k such that $(2 + \epsilon) \log n \leq k < (4 + 2\epsilon) \log n \leq 2k$. Over all k satisfying this inequality, the probability of a duplicate code is largest if $k = (2 + \epsilon) \log n$. Applying theorem 4.2.4 gives the stated probability bound. □

Subprotocol 20 ElectLeader(rec, sen)

$p \leftarrow \min(|\text{rec.LC}|, |\text{sen.LC}|)$
if rec.LC$[1..p]$ lexicographically precedes sen.LC$[1..p]$ **then**
▷ Propagate by epidemic the lexicographically greatest leader code
    rec.isLeader $\leftarrow$ False
    SetNewLeaderCode(rec, sen.LC)
if (not rec.isLeader) and ($|\text{rec.LC}| < |\text{sen.LC}|$) **then**
▷ Ensure all leader codes eventually have equal length
    SetNewLeaderCode(rec, sen.LC)

4.2.2. ElectLeader. ElectLeader works by propagating by epidemic the "winning" leader code, where a candidate leader drops out if they see an agent (whether leader or follower) with a leader code that beats its own. The trick is to define "win". We compare the shorter leader code with the same-length prefix of the other. If they disagree, the lexicographically largest wins. To ensure all leader code lengths are eventually equal, a follower with the shorter leader code replaces it with the longer one.[2]

The next lemma shows that the leader is probably unique when the population reaches level at least $\log n$. Let $k \in \mathbb{N}$ be such that $\log n \leq k$. When the candidate leaders generate new values of LC upon reaching level k, $|\text{LC}| = 2k \geq 2 \log n$. Since there are at least n^2 strings of length $2k \geq 2 \log n$, in the worst case, even if all n agents remain candidate leaders at that time, the probability that the lexicographically greatest leader code is duplicated is at most $\frac{1}{n}$.[3] Thus, with probability at least $1 - \frac{1}{n}$, one unique leader has the maximum leader code, and in $O(\log n)$ time this leader code reaches the remaining candidate leaders by epidemic, who drop out.

Lemma 4.2.6. *At any level $\geq \log n$, with probability $\geq 1 - \frac{1}{n}$, there is a unique leader.*

Proof. Every remaining candidate leader at level $\geq \log n$ has a leader code with at least $2 \log n$ bits. We say i and j *collide* if agents i and j are candidate leaders with the same leader code. Let $X_{i,j}$ be the indicator variable:

[2]Leaders with shorter codes do not replace with longer codes, because it may be that after adding new random bits to get to the current population level, that agent would have the lexicographically largest leader code. This is because a leader with a shorter leader code LC also has a shorter code C, so will eventually catch up in leader code length through the UniqueID protocol.
[3]This almost looks like a birthday problem, but we don't need all the remaining candidate leaders to have a unique leader code, only that the *largest* code appears only once.

$$X_{i,j} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ collide} \\ 0 & \text{otherwise} \end{cases}$$

Let $c \geq 2^{2\log n} = n^2$ be the number of possible leader codes. Note $\Pr[X_{i,j} = 1] = \frac{1}{c}$. Let $X_i = \sum_{j \neq i} X_{i,j}$; by linearity of expectation $\mathbb{E}[X_i] = \frac{n-1}{c}$. Since $c > n^2$, $\mathbb{E}[X_i] < \frac{n-1}{n^2} < \frac{1}{n}$. By Markov's inequality, $\Pr[X_i \geq 1] \leq \frac{1}{n}$, and the event $X_i \geq 1$ is equivalent to the event that the leader code of agent i is not unique. If we set i to be the agent with the lexicographically greatest leader code of any remaining candidate leader, we conclude that leader is unique with probability $\geq 1 - \frac{1}{n}$. □

By theorem 4.2.3, the protocol reaches level $\log n$ in $O(\log n \log\log n)$ time. By theorem 4.2.6, with high probability, in time $O(\log n \log\log n)$ ElectLeader converges (the second-to-last candidate leader is eliminated). Unfortunately ElectLeader is not terminating: the remaining leader does not know when it becomes unique. Thus it is not straightforward to compose it with the downstream protocols `DiscreteAveraging` and `timer`. Standard techniques for making the protocol terminating with high probability, such as setting a timer for a termination signal that probably goes off only after $K \log n \log\log n$ time for a large constant K, do not apply here, because when we start we don't know the value $\log n \log\log n$. Thus, it is necessary, each time a leader adds to its code length, to restart the downstream protocols.[4] This is done in SetNewLeaderCode, which is actually called by both ElectLeader *and* UniqueID, since extending `C` for a leader also requires extending `LC`, to maintain that $2|\texttt{C}| = |\texttt{LC}|$.

Subprotocol 21 `DiscreteAveraging`(rec, sen)

$\text{rec.ave}, \text{sen.ave} \leftarrow \lceil \frac{\text{rec.ave} + \text{sen.ave}}{2} \rceil, \lfloor \frac{\text{rec.ave} + \text{sen.ave}}{2} \rfloor$

4.2.3. Discrete Averaging. The previous subsections described how to set up a protocol (perhaps restarted many times) to elect a leader and to produce a value $\texttt{M} \geq 3n^3$. (With high probability we also have $\texttt{M} \leq 3 \cdot n^{18}$.) Thus we assume the initial configuration of this protocol is

[4] One might imagine restarts could be tied to the elimination of candidate leaders, which stops within $O(\log n \log\log n)$ time, rather than the extending of codes, which persists for $\Omega(n)$ time. However, the leader may become unique *before* level $\log n$, when $|\texttt{C}| < \log n$, so $\texttt{M} = 3 \cdot 2^{2|\texttt{C}|} < 3 \cdot n^3$ is not sufficiently large to ensure correctness and speed of `DiscreteAveraging`. (See theorem 4.2.7, which is applied with $c = 1$.)

one leader and $n-1$ followers, each storing this value $\mathtt{M}$, and that the goal is for all of them to converge to a value in $\mathtt{ave}$ such that $n = \lfloor \frac{\mathtt{M}}{\mathtt{ave}} + \frac{1}{2} \rfloor$ (i.e., $\frac{\mathtt{M}}{\mathtt{ave}} + \frac{1}{2}$ rounded to the nearest integer).

There is an existing *nonuniform* protocol [79] that can do the following in $O(\log n)$ time. Each agent starts with a bit $b \in \{0,1\}$ and a number $\mathtt{M} = \Omega(n^{3/2})$. Let n_b be the (unknown) number of agents storing bit b, so that $n_0 + n_1 = n$. The agents converge to a state in which they all report the value $n_1 - n_0$, the initial difference in counts between the two bits.

Their protocol requires that $\mathtt{M} \geq \lfloor n^{3/2}/\sqrt{2\delta} \rfloor$ to obtain an error probability of $\leq \delta$. The protocol is elegantly simple: agents with $b = 0$ start with an integer value $-\mathtt{M}$, while agents with $b = 1$ start with an integer value $\mathtt{M}$, and state space is$\{-\mathtt{M}, -\mathtt{M}+1, \ldots, \mathtt{M}-1, \mathtt{M}\}$. When two agents meet, they average their values, with one taking a floor and the other a ceiling in case the sum of the values is odd. If an agent holds value x, that agent's output is reported as $\lfloor \frac{nx}{\mathtt{M}} + \frac{1}{2} \rfloor$. This eventually converges to all agents sharing the population-wide average $(n_1 - n_0)\frac{\mathtt{M}}{n}$, and the estimates of this average get close enough for the output to be correct within $O(\log n)$ time [79].

Our protocol essentially inverts this, starting with a known $n_0 = 1$ and $n_1 = n-1$, computing the population size as a function of the average. The leader starts with value $\mathtt{ave} = \mathtt{M}$, and followers start with $\mathtt{ave} = 0$, and the state space is $\{0, 1, \ldots, \mathtt{M}\}$. The population-wide sum is always $\mathtt{M}$.[5] Eventually all agents have $\mathtt{ave} = \lceil \frac{\mathtt{M}}{n} \rceil$ or $\lfloor \frac{\mathtt{M}}{n} \rfloor$, which could take linear time in the worst case. We show below that with probability at least $1 - n^{-c}$, in $O(\log n)$ time, all agents' ave values are within n^c of $\frac{\mathtt{M}}{n}$. Each agent reports the population size as $\lfloor \frac{\mathtt{M}}{\mathtt{ave}} + \frac{1}{2} \rfloor$. This is the exact population size n as long as $\mathtt{M} \geq 3n^{c+2}$ and $\mathtt{ave}$ is within n^c of $\frac{\mathtt{M}}{n}$, as the following lemma shows.

LEMMA 4.2.7. *Let $c \geq 0$. If $M \geq 3n^{c+2}$, and $x \in \left[\frac{M}{n} - n^c, \frac{M}{n} + n^c\right]$, then $\lfloor \frac{M}{x} + \frac{1}{2} \rfloor = n$.*

PROOF. Since $\lfloor \frac{M}{x} + \frac{1}{2} \rfloor$ is monotone in x, it suffices to show this holds for the two endpoints of the interval. For the case $x = \frac{M}{n} - n^c$, since $x < \frac{M}{n}$, we have $n < \frac{M}{x}$, and

$$\begin{aligned}
\frac{M}{x} &= \frac{M}{\frac{M}{n} - n^c} = \frac{M}{\frac{M - n^{c+1}}{n}} = \frac{Mn}{M - n^{c+1}} \\
&\leq \frac{Mn}{M - M/(3n)} \quad \text{since } M \geq 3n^{c+2} \\
&= \frac{n}{1 - 1/(3n)} = \frac{n}{(3n-1)/(3n)} = \frac{3n^2}{3n-1} = n + \frac{1}{3(3n-1)} + \frac{1}{3} < n + \frac{1}{2}.
\end{aligned}$$

[5] Think of the leader starting with $\mathtt{M}$ "balls". Interacting agents exchange balls until they have an equal number, or within 1.

So $n < \frac{M}{x} < n + \frac{1}{2}$, so $\lfloor \frac{M}{x} + \frac{1}{2} \rfloor = n$. In the case $x = \frac{M}{n} + n^c$, a similar argument shows that $n - \frac{1}{2} < x < n$. □

The above results show that the count computed by `DiscreteAveraging` is correct if `M` is sufficiently large and `ave` is within a certain range of the true population-wide average $\frac{\mathtt{M}}{n}$. The next lemma, adapted from [79, Corollary 8], shows that each agent's `ave` estimate quickly gets within that range. That corollary is stated in terms of a general upper bound K on how far each agent's `ave` field starts from the true population-wide average. In our case, this is given by the leader, which starts with $\mathtt{ave} = \mathtt{M}$, while the true average is $\frac{\mathtt{M}}{n}$, so we choose $K = \mathtt{M} > \mathtt{M} - \frac{\mathtt{M}}{n}$ in Corollary 8 of [79], giving the following.

LEMMA 4.2.8 ([79]). *For all $\delta \in (0,1)$ and all $t \geq \ln(4M^2)$, with probability at least $1 - \delta$, after time t, each agent's* `ave` *field is in the interval* $\left[\frac{M}{n} - \sqrt{\frac{n}{2\delta}}, \frac{M}{n} + \sqrt{\frac{n}{2\delta}}\right]$.

COROLLARY 4.2.9. *Let $c > 0$ and let $\delta = \frac{1}{2n^{2c-1}}$. For all $t \geq \ln(4M^2)$, with probability at least $1 - \delta$, within time t, each agent's* `ave` *field is in the interval* $\left[\frac{M}{n} - n^c, \frac{M}{n} + n^c\right]$.

Setting $c = 1$ (so $\delta = \frac{1}{2n}$) gives the following corollary.

COROLLARY 4.2.10. *For all $t \geq \ln(4M^2)$, with probability at least $1 - \frac{1}{2n}$, within time t, each agent's* `ave` *field is in the interval* $\left[\frac{M}{n} - n, \frac{M}{n} + n\right]$.

4.2.4. `timer`. Note that `DiscreteAveraging` does not actually write the value $\lfloor \frac{\mathtt{M}}{\mathtt{ave}} + \frac{1}{2} \rfloor$ into the `count` field; that is the job of the `timer` protocol, which we now explain. The leader is guaranteed with high probability to become unique at least by level $\log n$ (theorem 4.2.6). However, since UNIQUEID likely continues after this point, although the leader is unique, when its level increases, the leader will again generate more bits for its leader code, updating its value `M`, initiating a restart of `DiscreteAveraging`. The problem is that although we can prove that the agents likely reach level $\log n$ in $O(\log n \log\log n)$ time, it may take much longer to reach subsequent levels. Thus, although the value `M` estimated at any level $k \geq \log n$ is large enough for `DiscreteAveraging` to be correct, if `DiscreteAveraging` were to blindly write $\lfloor \frac{\mathtt{M}}{\mathtt{ave}} + \frac{1}{2} \rfloor$ into `count` each time `ave` changes, the output will be disrupted while this restart of `DiscreteAveraging` converges.

We deal with this problem in the following way. When the leader restarts `DiscreteAveraging`, it simultaneously restarts `timer`, which is a *phase clock* as described by Angluin et al. [12]. `timer`

Subprotocol 22 Timer(rec, sen)

▷ run phase clock until `MaxPhase` = 1184 is reached
if rec.`isLeader` and (rec.`phase` = sen.`phase`) and (rec.`phase` < `MaxPhase`) **then**
    rec.`phase` ← rec.`phase` + 1
if (not rec.`isLeader`) and (rec.`phase` < sen.`phase`) **then**
    rec.`phase` ← sen.`phase`
newCount ← $\lfloor \frac{\text{rec.M}}{\text{rec.ave}} + \frac{1}{2} \rfloor$ ▷ $\frac{\texttt{M}}{\texttt{ave}}$ rounded to the nearest integer
▷ only write output if timer is done and new count is different
if (rec.`phase` = `MaxPhase`) and (rec.`count` ≠ newCount) and ($\texttt{M} \geq 3 \cdot \text{newCount}^3$) **then**
    rec.`count` ← newCount

is so named because we can find $\beta_l < \beta_u$ and `MaxPhase` such that `MaxPhase` phases of the phase clock will take time between $\beta_l \ln n$ and $\beta_u \ln n$ with high probability. So long as $\beta_l \ln n$ is greater than a high-probability upper bound on the running time of `DiscreteAveraging`, the timer likely will not go off (reach the final phase `MaxPhase`) until `DiscreteAveraging` has converged. It is only once `timer` has reached phase `MaxPhase` that `count` is written, and then only if the new calculated size differs from the previous value in `count`.

There is one additional check done before writing to `count`: if newCount = $\lfloor \frac{\texttt{M}}{\texttt{ave}} + \frac{1}{2} \rfloor$, we must have $\texttt{M} \geq 3 \cdot \text{newCount}^3$ in order to write to `count`. In particular, if $\texttt{M} \geq 3n^3$, then newCount cannot be n unless $\texttt{M} \geq 3 \cdot \text{newCount}^3$. This is an optimization to save space. `DiscreteAveraging` is only guaranteed to get the correct size n efficiently if $\texttt{M} \geq 3n^3$. However, when `ave` is small before convergence (e.g., 1) then $\lfloor \frac{\texttt{M}}{\texttt{ave}} + \frac{1}{2} \rfloor$ can be as large as `M`, requiring $18 \log n$ bits. But if $\lfloor \frac{\texttt{M}}{\texttt{ave}} + \frac{1}{2} \rfloor = n$ (i.e., is correct) then this value requires at most $\log n$ bits. Since `M` could be as large as $3 \cdot n^{18}$, requiring $O(1) + 18 \log n$ bits, this implies `count` could be as large as n^6, requiring $6 \log n$ bits.

4.3. Simulation results

Simulations for the ExactCounting protocol are shown in fig. 4.2.

4.4. Proofs for correctness of ExactCounting protocol

The following is adapted from [12, Corollary 1]. It relates the number of phases in a phase clock to upper and lower bounds on the likely time spent getting to that phase. Our proof appeals entirely to Corollary 1 of [12] but, unlike [12], the exact relationship between the constants is given in the lemma statement.

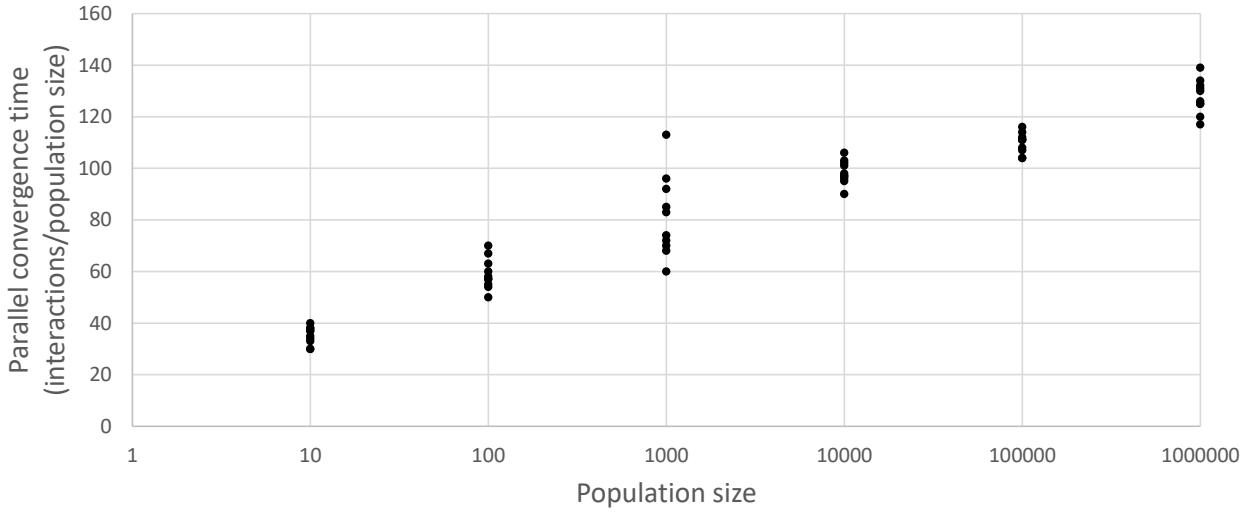


FIGURE 4.2. Simulated convergence time of EXACTCOUNTING. The dots indicate the convergence time of individual experiments. The population size axis is logarithmic, so exactly $c \log_{10} n$ time complexity would correspond to a line of slope c. Since $\log \log n$ is "effectively constant" (< 5) for the values of n shown, we expect the plot to appear roughly linear.

LEMMA 4.4.1 ([12]). *Let $\beta_l, \epsilon_l, \epsilon_u > 0$, and define $p = \max(8\epsilon_l, 32\beta_l)$ and $\beta_u = 4p(\epsilon_u + 2)$. Let T_p be the time needed for a phase clock with $\geq p$ phases to reach phase p. Then for all sufficiently large n, $\Pr[T_p < \beta_l \ln n] < \frac{1}{n^{\epsilon_l}}$ and $\Pr[T_p > \beta_u \ln n] < \frac{1}{n^{\epsilon_u}}$.*

PROOF. Based on [12, Corollary 1], setting (variables of [12, Corollary 1] on the left, and our variables on the right) $c = \epsilon_l, d = \beta_l, k = p$, and $a = 1/16$, then by choosing $p = \max(8\epsilon_l, 32 \cdot \beta_l)$, we have $\Pr[T_p < \beta_l \ln n] < \frac{1}{n^{\epsilon_l}}$. By theorem 1.3.4, for all $\alpha_u > 0$, the epidemic corresponding to each phase i will complete (all agents reach phase i) in time $> \alpha_u \ln n$ with probability $< 4n^{-\alpha_u/4+1}$. Since the time to complete the epidemic is an upper bound on the time for the leader to interact with an agent in phase i (which could happen before the epidemic completes), we also have that the phase takes time $> \alpha_u \ln n$ with probability $< 4n^{-\alpha_u/4+1}$. By the union bound over all p phases, there exists a phase $1 \leq i \leq p$ taking time $> \alpha_u \ln n$ with probability $< 4pn^{-\alpha_u/4+1}$. Since at least one phase must exceed time $\alpha_u \ln n$ for the sum to exceed $p\alpha_u \ln n$, $\Pr[T_p > p\alpha_u \ln n] < 4pn^{-\alpha_u/4+1}$. Let $\alpha_u = 4(\epsilon_u + 2)$. Substituting $\beta_u = p\alpha_u = 4p(\epsilon_u + 2)$ gives $\Pr[T_p > \beta_u \ln n] < 4pn^{-(4(\epsilon_u+2))/4+1} = 4pn^{-\epsilon_u - 1} < n^{-\epsilon_u}$, which completes the proof. □

The next lemma says that `DiscreteAveraging` and `timer` "happen the way we expect": first `DiscreteAveraging` converges, before `timer` ends and records the output of `DiscreteAveraging`, all in $O(\log n)$ time.

When we say "`DiscreteAveraging` converges", this refers to the `DiscreteAveraging` protocol running in isolation, not as part of a larger protocol that might restart it. That is to say, it may be that `DiscreteAveraging` converges, but ExactCounting has not converged, since ExactCounting then restarts `DiscreteAveraging` and subsequently changes the `count` field. Intuitively, it follows by a simply union bound on the probability that `DiscreteAveraging` is too slow (theorem 4.2.10) or `timer` is too fast (theorem 4.4.1).

Lemma 4.4.2. *For any level* $\geq \log n$*, if it takes* $\geq 14208 \ln n$ *time to start the next level, with probability* $\geq 1 - \frac{3}{n}$*, first* `DiscreteAveraging` *converges to the correct output, then* `timer` *ends and writes* n *into* `count`*, in* $\leq 14208 \ln n$ *time.*

Proof. theorem 4.2.5 applied with $\epsilon = 1$ shows that agents codes' length are $\leq 6 \log n$ with probability $\geq 1 - \frac{1}{n}$. theorem 4.2.6 shows that after level ℓ the leader is unique with probability $\geq 1 - \frac{1}{n}$. Since $\ell \geq \log n$ the value of M is will be $\geq 3n^3$. By theorem 4.2.7, if `DiscreteAveraging` converges, then $\lfloor \frac{\texttt{M}}{\texttt{ave}} + \frac{1}{2} \rfloor$ exactly n.

By the union bound on theorem 4.4.1 and theorem 4.2.10, with probability $\leq \frac{1}{n} + \frac{1}{n}$ the timer takes more than $14208 \ln n$ time, or `DiscreteAveraging` takes more than $37 \ln n$ time to converge. Negating these conditions gives conclusion of the lemma.

To show that `timer` does not end until `DiscreteAveraging` converges, we apply theorem 4.4.1 again, but using the time lower bound for `timer`. Letting $\epsilon_l = 1$ and $\beta_l = 37$, theorem 4.4.1 gives that for $p = 32\beta_l = 1148$, with probability $\geq 1 - \frac{1}{n}$, `timer` does not end before $\beta_l \ln n$ time. Applying the union bound to this case and the previous two cases then gives probability $1 - \frac{1}{n}$ as desired. □

Finally, we can prove the "with high probability" portion of the main theorem.

Theorem 4.4.3. *With probability at least* $1 - \frac{10+5 \log \log n}{n}$*,* ExactCounting *converges to the correct output within* $6 \ln n \log \log n$ *time, and each agent uses at most* $17 + 60 \log n$ *bits.*

A formal proof is given in the full version of our paper [51].

PROOF SKETCH. We sketch the ideas while omitting exact bounds on time and probability. Statements below are "with high probability". Define k to be the unique power of two such that $\log n \leq k < 2\log n$. By theorem 4.2.3, level k is reached quickly, so it suffices to prove fast convergence after the event that k is reached. By theorem 4.2.6, the leader is unique at level k, therefore also $2k$ and $4k$, and since $k \geq \log n$, $\texttt{M} \geq 3n^3$. So by theorem 4.4.2, `DiscreteAveraging` will converge within time $t = 14208 \ln n$.

We look at three subcases: among levels $k, 2k, 4k$, one is the earliest among the three where $> t$ time is spent. By theorem 4.2.4, level $4k$ is not exceeded since codes are unique. Since codes are unique at level $4k$, at $> t$ (in fact, infinite time) will be spent at level $4k$. But it could be that the protocol also spends time $> t$ at level k or $2k$. Whichever is the first among these three to spend time $> t$, since the previous spent less time, it takes time $\leq 2t$ to reach the first level taking time $> t$. By theorem 4.4.2, `DiscreteAveraging` converges in time $\leq t$, and by the time *upper* bound of theorem 4.4.1, `timer` reaches phase `MaxPhase` in time $\leq t$ and records the output of `DiscreteAveraging`.

If we are at level k or $2k$, then we might go to a new level. If this is guaranteed to happen within $O(\log n)$ time, then we would not need the `timer` protocol. We could simply claim that `DiscreteAveraging` will converge at the last level reached, whether k, $2k$, or $4k$. The problem that `timer` solves is that EXACTCOUNTING may reach level k quickly, `DiscreteAveraging` converges quickly, yet a small number of duplicate nodes remain, say 2. It takes $\Omega(n)$ time for them to interact and increase to level $2k$, which restarts `DiscreteAveraging`. By the time *lower* bound of theorem 4.4.1, in each of these restarts `timer` will not reach phase `MaxPhase` until `DiscreteAveraging` reconverges, so the `count` field will not be overwritten. Thus convergence happened at the *first* level where we spent time $> t$, even if there are subsequent restarts. □

4.4.1. ExactCounting converges in fast expected time. Most of the technical difficulty of our analysis is captured by the "with high probability" results stated already. EXACTCOUNTING is also stabilizing, meaning that with probability 1 it gets to a correct configuration that is *stable* (the output cannot change). Probability 1 correctness is required for the expected correct convergence time to be finite, and indeed it asymptotically matches the high probability convergence time of

$O(\log n \log\log n)$. However, the protocol takes longer to stabilize, up to $O(n)$ time, since it does not stabilize until UniqueID stabilizes.[6]

The next theorem shows a fast expected convergence time, and it completes the second portion of the main result, theorem 4.2.1.

Theorem 4.4.4. ExactCounting *converges in expected time* $7 \ln n \log\log n$.

First we establish some other claims necessary to prove theorem 4.4.4. Recall that a protocol *stabilizes* if it converges to the correct output with probability 1.

Lemma 4.4.5. ExactCounting *stabilizes to the correct population size.*

Proof. Since each agent generates code bits uniformly at random, any pair of agents has probability 0 to generate the same infinite sequence of bits. So with probability 1 all agents eventually have unique codes, and UniqueID stabilizes. We now show that implies ElectLeader stabilizes.

Since there are n agents, UniqueID cannot terminate until at least level $\log n$. So when UniqueID terminates, $|\texttt{C}| \geq \log n$, so $\texttt{M} = 3 \cdot 2^{3|\texttt{C}|} \geq 3n^3$. Note that `DiscreteAveraging` also has an equivalence between converging and stabilizing: once all agents' `ave` fields are within a certain interval, they cannot leave that interval. So by theorem 4.2.7, `DiscreteAveraging`, if it stabilizes, will stabilize to values of `ave` such that $\lfloor \frac{\texttt{M}}{\texttt{ave}} + \frac{1}{2} \rfloor = n$. We claim that `DiscreteAveraging` stabilizes with probability 1, which is shown below. Furthermore, `timer` reaches `MaxPhase` with probability 1, since the only way to avoid incrementing the phase of an agent is forever to avoid any interaction between it and an agent at the next phase, which happens with probability 0. This implies that with probability 1, the correct population size is eventually written into the field `count`.

It remains to show the claim that `DiscreteAveraging` stabilizes with probability 1. Define the potential function Φ for any configuration $\mathbf{c}$ by $\Phi(\mathbf{c}) = \sum_a |a.\texttt{ave} - \texttt{M}/n|$, where the sum is over each agent a in the population. The `DiscreteAveraging` protocol stabilizes by the time Φ reaches its minimum value,[7] which is either n or 0 depending on whether n divides `M`, when all agents have ave

[6]Prior to that, it is possible, with low probability, after n is written into each agent's `count` field, for a subsequent restart of `DiscreteAveraging` to write incorrect values, if the corresponding restart of `timer` completes too quickly. So no configuration is stable until UniqueID converges and triggers the final restart.

[7]If $\texttt{M} \gg 2n^2$, then `DiscreteAveraging` can stabilize prior to this time, since the `ave` values do not have to reach their final convergent values for the output function $\lfloor \frac{\texttt{M}}{\texttt{ave}} + \frac{1}{2} \rfloor$ to converge, by theorem 4.2.7.

$= \lfloor \mathtt{M}/n \rfloor$ or $\lceil \mathtt{M}/n \rceil$. We claim that Φ is nonincreasing with each transition of `DiscreteAveraging`. When two agents meet, there are two cases: 1) both of their `ave` fields are $\geq \lceil \mathtt{M}/n \rceil$, or both are $\leq \lfloor \mathtt{M}/n \rfloor$, and 2) one of their `ave` fields is $<$ (resp., $\leq$) $\lfloor \mathtt{M}/n \rfloor$, and the other is $\geq$ (resp., $>$) $\lceil \mathtt{M}/n \rceil$. Taking the average of their `ave` fields, in case (1) does not change Φ, and in case (2) decreases Φ, so Φ is nonincreasing.

It remains to show that Φ will reach its minimum value with probability 1. If the protocol has not converged, then there must be some agent with an `ave` field not equal to either $\lfloor \mathtt{M}/n \rfloor$ or $\lceil \mathtt{M}/n \rceil$. But since the population-wide sum of the `ave` values is always `M`, this implies that case (2) holds for some pair of agents. With probability 1, such a pair of agents must eventually meet, decreasing Φ. So with probability 1, Φ eventually reaches its minimum value. □

UniqueID stabilizes when all agents have a unique code since, by inspection of the UniqueID protocol, this implies that the codes no longer can change. The next lemma shows that this happens at most linear time. A complete proof is given in the full version of our paper [51].

Lemma 4.4.6. UniqueID *stabilizes in expected time at most* $1.03n$.

Proof of theorem 4.4.4. By theorem 4.4.3, ExactCounting converges to the correct answer in time $6 \ln n \log\log n$ with probability at least $1 - \frac{10+5\log\log n}{n}$.

By theorem 4.4.6, UniqueID converges in expected time at most $1.03n$. Once it has converged, it takes expected time $O(\log n)$ for `DiscreteAveraging` to converge and `timer` to write the correct output if it has not already been written. The sum of these times is at most $1.04n$ for sufficiently large n. We can bound the expected time as

$$\begin{aligned}
& \Pr[\text{convergence in time } \leq 6 \ln n \log\log n] \cdot 6 \ln n \log\log n + \\
& \Pr[\text{convergence in time } > 6 \ln n \log\log n] \cdot 1.04n \\
= \; & \left(1 - \frac{10 + 5\log\log n}{n}\right) \cdot 6 \ln n \log\log n + \frac{10 + 5\log\log n}{n} \cdot 1.04n \\
< \; & 6 \ln n \log\log n + 1.04(10 + 5\log\log n) \\
< \; & 7 \ln n \log\log n.
\end{aligned}$$

□

4.4.2. Increasing time to minimize state complexity. ExactCounting generalizes easily to trade off time and memory: by adjusting the rate at which the code length grows, the convergence time $t(n) = O(f(n) \log n)$, where $f(n)$ is the number of stages required for the code length to reach $\log n$. The minimum state complexity is achieved when the code length increments by 1 each stage, so that $f(n) = \log n$ and $t(n) = \log^2 n$. In this case, a straightforward adaptation of theorem 4.2.4, letting $\epsilon = 1$, indicates that with probability at least $1 - \frac{1}{n}$, all codes are unique by level $3 \log n$. Carrying through the string length and integer bounds from the main argument gives state complexity $O(n^{30})$ for the full protocol and $O(n^9)$ for just the leader election.

4.5. Conclusion and further discussions

We have shown a uniform population protocol computing the exact population size using $\approx 60 \log n$ bits memory (i.e., poly(n) states) and $O(\log n \log \log n)$ time.

By removing the `DiscreteAveraging` and `timer` subprotocols, the remainder is a uniform protocol electing a leader in $O(\log n \log \log n)$ time and $\approx 18 \log n$ bits of memory (for `C` and `LC`).

Berenbrink, Kaaser, and Radzik [28] improved the space complexity as well as the time complexity of our exact counting protocol. Their protocol uses $O(n \log n)$ states and converges in $O(\log n)$ time both WHP [28]. They also explained how to modify the protocol to achieve stabilization in $O(\log n)$ time while using $O(n \log n \log \log n)$ states with error detection schemes that point agents to switch to the naïve slow (but stable) section 2.3.1 as a backup.

To reduce state complexity, one could imagine a variant of the size counting problem that allows for $< n$ states per agent, if the population size is represented in a distributed fashion. For example, one could imagine $1 + \lfloor \log n \rfloor$ agents having a pair $(s, b) \in \mathbb{N} \times \{0, 1\}$, where b is a bit of the output count and $s \in \{0, \ldots, \lfloor \log n \rfloor\}$ is its significance, with $s < 0$ in all other $n - \log n$ agents to represent that they do not represent the output.

CHAPTER 5

Approximate Size Counting in Population Protocols

5.1. Introduction

We study *uniform* population protocols where each agent uses an *identical* transition algorithm that does not depend on the population size n. Many existing polylog(n) time protocols for leader election and majority computation are nonuniform: to operate correctly, they require all agents to be initialized with an approximate estimate of n (specifically, the value $\lfloor \log n \rfloor$).

Our first main result is a uniform protocol for calculating $\log(n) \pm O(1)$ with high probability in $O(\log^2 n)$ time and $O(\log^4 n)$ states ($O(\log \log n)$ bits of memory). The protocol is not *terminating*: it does not signal when the estimate is close to the true value of $\log n$. If it could be made terminating with high probability, this would allow composition with protocols requiring a size estimate initially. We do show how our main protocol can be indirectly composed with others in a simple and elegant way, based on *leaderless phase clocks*, demonstrating that those protocols can in fact be made uniform.

However, our second main result implies that the protocol *cannot* be made terminating, a consequence of a much stronger result: a uniform protocol for *any* task requiring more than constant time cannot be terminating even with probability bounded above 0, if infinitely many initial configurations are *dense*: any state present initially occupies $\Omega(n)$ agents. (In particular no leader is allowed.) Crucially, the result holds no matter the memory or time permitted.

Finally, we show that *with* an initial leader, our size-estimation protocol can be made terminating with high probability, with the same asymptotic time and space bounds.

The original model [10] assumed a set of states and transitions that is constant with respect to n. However, for important distributed computing problems such as leader election [53], majority computation [2], and computation of other functions and predicates [19] no constant-state protocol can stabilize in sublinear time with probability 1.[1] This motivated the study of protocols in which

[1] A protocol *stabilizes* when it becomes unable to change the output. A protocol *converges* in a given random execution when the output stops changing, though it could take longer to subsequently stabilize. Known time

the set of states and transitions grows with n (essentially adding a non-constant *memory* to each agent). Such protocols achieve leader election and majority computation using $O(\text{polylog}(n))$ time, while keeping the number of states "small": typically $O(\text{polylog}(n))$ [2,3,5,7,20,20,24,24,25,25, 26,27,28,30,50,60,61,79,91] although $O(\log \log n)$ states suffice for leader election [26,60] and exact majority [50].

Unfortunately, many of these sublinear-time protocols [2,3,5,20,24,25,26,27,30,50,61,79,91] are *nonuniform*: the set of states and transitions are allowed to depend arbitrarily on n (this is not true of all, see for example recent fast, low-memory leader election protocols [60,61]). This capability is used to initialize each agent with an approximate estimate of n (the value $\lfloor \log n \rfloor$) required by the protocols. [2] A representative example portion of such a protocol is shown in fig. 5.1: each agent has an internal "counter", which increments upon each encounter with an x. When the counter reaches $\log n$, the protocol terminates (or moves to a different "stage").

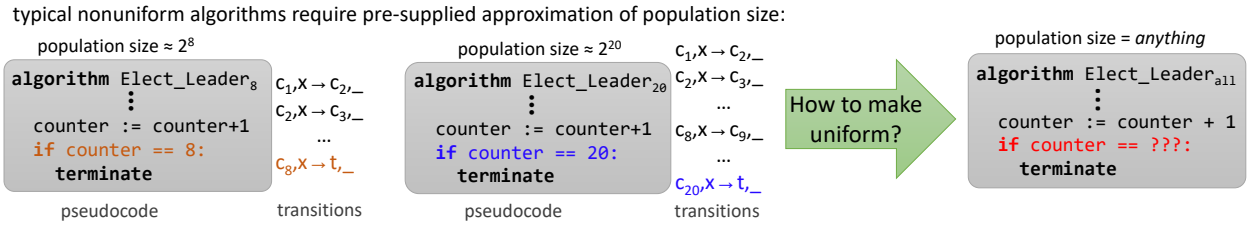


FIGURE 5.1. Many population protocols with $\omega(1)$ states use nonuniform algorithms: the value $\log_2 n$ is "hardcoded" into the reactions. Above, "`terminate`" could mean terminate the whole algorithm, or it could mean "move to the next stage of the algorithm".

More desirable would be a *uniform* protocol in which each agent's local algorithm for computing the outputs, given the inputs, has no knowledge of n. Such an algorithm may produce outputs longer than its inputs, retaining the ability to use a number of states that grows with the population size. A uniform protocol can be deployed into *any* population without knowing in advance the size, or even a rough estimate thereof.

Although the original model of population protocols, stipulating a constant set of states, is uniform, when the number of states vary with the population size in this manner, little is known

lower bounds [2, 19, 53] are on stabilization, not convergence. Recently Kosowski and Uznanski [69] achieved a breakthrough result, showing constant-state protocols for leader election and all decision problems computable by population protocols (the *semilinear* predicates), which converge with high probability in polylog(n) time, and for any $\epsilon > 0$, probability 1 protocols for the same problems that converge in $O(n^\epsilon)$ expected time. The latter protocols require $\Omega(n)$ time to stabilize, as would any constant-state protocol due to the cited time lower bounds.

[2] In fact, uniform protocols with "dense" initial configurations *cannot* properly terminate. (see [47])

about the computational abilities and limitations of uniform protocols (with a few interesting exceptions [36, 37, 74, 75]).

5.1.1. Contribution. Nonuniform protocols in the literature [2, 3, 5, 27, 30] initialize each agent with the value $\lfloor \log n \rfloor$. Hence we study the problem of computing an approximate estimate of $\log n$. Our first main result, theorem 5.2.1, is a uniform protocol, starting from a configuration where all n agents are in an identical state, that with high probability computes $\log n \pm O(1)$ (storing the value in every agent), using $O(\log^2 n)$ time and $O(\log^4 n)$ states.[3]

One might hope to use this protocol as a subroutine to "uniformize" existing nonuniform protocols for leader election and majority [2, 3, 5, 27, 30].[4] Suppose the size-estimating protocol could be made terminating, eventually producing a termination "signal" that with high probability does not appear until the size estimate has converged. This would allow composition with other protocols requiring the size estimate. It has been known since the very beginning of the population protocol model [10] that termination cannot be guaranteed with probability 1. However, some leader-driven protocols can be made terminating with high probability, including simulation of register machines [12] or exact (but slow) population size counting [74].

Despite this difficulty in directly composing size estimation with a downstream protocol (or several stages/subprotocols composed in series), we present a general and simple method of composition (via restarting), based on a "leaderless phase clock" using a weaker log population size estimate s (called `logSize2` in the pseudocode in section 5.2) obtained initially (where $\log n - \log \ln n \leq s \leq 2 \log n$ w.h.p.).[5] Based on s and the expected convergence time of the downstream protocol, each agent once per interaction increments a counter c, from 0 up to $f(s)$, and the first agent to reach $f(s)$ signals the entire population to terminate (or move to the next stage). $f(s)$ is chosen large enough that no agent reaches $f(s)$ before the downstream protocol converges. The entire downstream protocol is reset if the initial size estimate s changes. With the above scheme, agents need to store the variables s, c, and possibly also $f(s)$ (in our case $f(s) = O(s)$ so it need not be stored explicitly, but if $f(s) = \text{poly}(s)$, for example, $f(s)$ may need to be stored separately from s). If the

[3]It appears difficult to compute $\lfloor \log n \rfloor$ exactly, rather than within a positive additive constant, since for all k, such a protocol could distinguish between the exact population sizes $2^k - 1$ and 2^k.
[4]Some protocols for leader election [60, 61] are uniform, but other protocols [2, 5, 27, 30] have the benefit of simplicity and may possibly be easier to reason about and compose with other protocols.
[5]The first leaderless phase clock for population protocols was proposed in [3]. Ours is different, based on [91]. Both are nonuniform, relying on an estimate of $\log n$.

downstream protocol requires $t(n)$ time to converge, then agents also set their threshold $f(s) > t(n)$ (where $f(s)$ is "large" compared to $t(n)$). This requires $O(f(s)^2 \cdot \log n)$ states will be added to the state complexity of the protocol, or $O(\log^2 n)$ if $f(s) = O(\log n)$ (as in our case) since $f(s)$ need not be stored explicitly. To compose multiple downstream stages/subprotocols in series, we also need a way to compute and possibly store the number K of stages (in our case $K = \Theta(\log n)$, also chosen as a constant times s, so K need not be stored explicitly), and we need to store an index indicating which stage we are on. For K stages, this multiplies the state complexity by K if $K = O(\log n)$ and K^2 otherwise (since K must be stored explicitly in the latter case).

5.1.2. Definition of correctness and time. The notion that a protocol's configuration "has the correct answer" is problem-specific. For leader election, it means there is a single leader agent. For predicate computation, it means all agents have the correct Boolean output. In this chapter, since our goal is to approximate $\log n$ within additive factor 5.7, we say a configuration is *correct* if the `output` field of each agent is within 5.7 of $\log n$.[6]

The following definitions match those used in the literature, when other notions of "correct" are substituted. Let $\mathcal{E} = (\mathbf{c}_0, \mathbf{c}_1, \ldots)$ be an infinite execution. A configuration $\mathbf{c}$ is *stably correct* if every configuration reachable from $\mathbf{c}$ is correct.[7] We say $\mathcal{E}$ *converges* at interaction i if $\mathbf{c}_i$ is not correct and for all $j > i$, $\mathbf{c}_j$ is correct. We say $\mathcal{E}$ *stabilizes* at interaction i if $\mathbf{c}_i$ is not stably correct and for all $j > i$, $\mathbf{c}_j$ is stably correct. A protocol can converge and/or stabilize with probability 1 or a smaller probability. However, if the set of reachable configurations is bounded with probability 1 (which is the case for the protocols discussed in this chapter), then for any $p \in [0, 1]$, a protocol converges with probability p if and only if it stabilizes with probability p.[8] For a computational task T equipped with some definition of "correct", we say that a protocol $\mathcal{P}$ *stably computes* T

[6]We note that our notion of function approximation differs from that of Belleville, Doty, and Soloveichik [19]. They use a *distributed* output convention, where the output of a function $f : \mathbb{N}^d \to \mathbb{N}$ is encoded as the population count of agents in a special output state y. Thus one must examine the entire population to know the output. In our *local* output convention, each agent has a field encoding a value from the function's range. The output is undefined if some agents have different values, and defined to be their common value otherwise. This is similar to how Boolean predicate output with range $\{0, 1\}$ is encoded in population protocols [10].

[7]Belleville, Doty, and Soloveichik [19] also consider function approximation, but define a configuration to be stable if the output *cannot* change, whereas we allow it to change within a small interval around the correct value. The time lower bound techniques of [19] do not apply to our more relaxed notion of stability.

[8]Let C and S respectively be the set of stabilizing and converging executions. Clearly $S \subseteq C$. Although $S \subsetneq C$ is possible, we argue that $\Pr[C \setminus S] = 0$. Suppose a protocol converges in an execution $(\mathbf{c}_0, \mathbf{c}_1, \ldots)$ at interaction i (so $\mathbf{c}_j$ is correct for all $j > i$). If did not stabilize, then for all $j > i$, some incorrect configuration $\mathbf{d}_j$ would be reachable from $\mathbf{c}_j$. Let $p_j > 0$ denote the probability of reaching $\mathbf{d}_j$ from $\mathbf{c}_j$. The set of reachable configurations is bounded with probability 1, so $\min_{j>i} p_j$ is well-defined and positive. The probability of never reaching any $\mathbf{d}_j$ is then 0.

with probability p if, with probability p, it stabilizes (equivalently, converges). If p is omitted, it is assumed $p = 1$. However, when measuring time complexity, convergence and stabilization may be much different. We say that $\mathcal{P}$ converges (respectively, stabilizes) in (parallel) time $t(n)$ with probability p if, with probability p, it produces an execution that converges (resp., stabilizes) by interaction i, where $i/n \leq t(n)$. Many protocols converge much faster than they stabilize, such as those that combine a fast, error-prone subprotocol with a slow, error-free protocol, e.g., [12,38,69]. However, for the protocol of this chapter, convergence and stabilization coincide. We use the term "converge" throughout the chapter to refer to this event.

Many papers separately measure high-probability time convergence and expected time to converge. Our protocol has positive probability of error, but we argue that expected time is a meaningful notion only with error probability 0, which is why we do not measure expected time. The only reasonable definition of "time until correctness" on a non-converging execution is ∞. So with $\Pr[\text{doesn't converge}] > 0$, the expected convergence time is $\mathbb{E}[\text{time}|\text{converges}] \cdot \Pr[\text{converges}] + \mathbb{E}[\text{time}|\text{doesn't converge}] \cdot \Pr[\text{doesn't converge}] = \mathbb{E}[\text{time}|\text{converges}] \cdot \Pr[\text{converges}] + \infty = \infty$. One could imagine measuring only $\mathbb{E}[\text{time}|\text{converges}]$. However, conditioning can artificially "speed up" the process.[9]

5.2. Fast protocol for estimating population size within constant additive error

In this section we describe a uniform protocol for computing the value of $\log n$ with an additive error, i.e., estimating the population size to within a constant multiplicative factor. We say a population protocol is *leaderless* if all agents start in the same state.

THEOREM 5.2.1. *There is a uniform leaderless population protocol that converges in time* $O(\log^2 n)$ *with probability* $\geq 1 - 1/n^2$, *uses* $O(\log^4 n)$ *states with probability* $\geq 1 - O(\log n)/n$, *and stores in each agent an integer* k *such that* $|k - \log n| \leq 5.7$ *with probability* $\geq 1 - 9/n$.

We note that the protocol has a positive probability of error. It is open to find a protocol using polylog(n) time/states computing $\log(n) \pm O(1)$ with probability 1.

[9]Consider a hypothetical protocol that runs a parallel subprotocol S that completes quickly and, upon completion, somehow prevents the main protocol M from converging. The main protocol, on the other hand, may somehow detect if it completes before S does, and if so, M then shuts S down. Many executions will not converge, but those that do must be very fast in order to converge before S completes. Thus conditioning on convergence "anthropically speeds up" convergence [1]. This is an extreme example that has the property that the probability of correctness is reduced by S, but it nevertheless shows that measuring conditional expected time can be problematic.

The protocol is described and its time and state complexity analyzed in section 5.2.2. Much of the analysis of the approximation involves proving a bound on the moment-generating function of a maximum of geometric random variables, enabling the Chernoff technique can be applied to sums of such variables. This is quite nontrivial and shown in section 5.7.

5.2.1. Intuition. Alistarh et al. [2] describe a protocol for estimating $\log n$ within a constant multiplicative factor. A $\frac{1}{2}$-geometric random variable is the number of flips needed to get one head when flipping a fair coin. In their protocol, each agent generates an independent geometric random variable $\mathbf{G}_i$, then propagates the maximum $\mathbf{M} = \max_{1 \le i \le n} \mathbf{G}_i$ by *epidemic*: transitions of the form $i, j \to j, j$ for $i \le j$, which in $O(\log n)$ time "infect" all agents with the maximum. It is known that $\mathbb{E}[\mathbf{M}] \approx \log n$ [56], and $\log n - \log \ln n \le \mathbf{M} \le 2 \log n$ with probability $\ge 1 - O(1)/n$ (theorem 5.7.7).

We take the obvious extension of this approach: do this K times and take an average. The estimated average is within $O(1)$ of $\log n$ so long as $K = \Omega(\log n)$ (theorem 5.7.10). One problem to solve first is how to calculate K; after all, $K = \Theta(\log n)$ scales with n, so with a uniform protocol it cannot be encoded into the agents at the start. The agents estimate it using the protocol of [2]. Since that protocol is converging but not terminating (provably it cannot be made terminating by [47]), each time an agent updates its value of K, it reinitializes the remainder of its state.

However, a trickier problem remains: a naïve approach to implement "averaging of K numbers" requires storing $K = \Theta(\log n)$ numbers in each agent, each having value $\Theta(\log n)$, implying the number of states is $\Theta((\log n)^{\log n}) = \Theta(n^{\log \log n})$. This is even more than the $O(n^{60})$ sufficient to quickly compute *exactly* n [51]. To overcome this problem, we use a "leaderless phase clock" similar to those of [3, 80, 91], but uniform. Unlike the phase clock used by [3, 80], our leaderless phase clock simply increments a counter on every interaction. This simultaneously gives an elegant way to compose our protocols with downstream protocols requiring the size estimate. Agents count their number of interactions and compare it with a threshold value $\Theta(\log n)$. Whenever their number of interaction passes the threshold they will move to the next round similar to the protocol described above (the population with a leader). The threshold is calculated in the following way. In our protocol, agents start generate a geometric random variable called `logSize2` and propagate the maximum `logSize2` among themselves. After agents agree on the `logSize2` variable, a constant multiple 95.`logSize2` is the threshold in their leaderless phase clock. This lets the agents

synchronize epochs of the algorithm, each taking $O(\log n)$ time, and prevent the next epoch from starting until the previous has concluded.

The probabilistic clock inside agents might go off very soon at the very beginning of the protocol, but after $O(\log n)$ time all agents will store the maximum generated `logSize2` and their leaderless phase clock will eventually converge to a stable one which goes off after completion of a predefined constant factor of $\log n$; to handle this, each time an agent updates its value of `logSize2`, the remainder of its state is reset and it begins the rest of the protocol anew. Restarting the downstream protocol is a known technique in population protocols also used in [60] to compose two leader elimination subprotocols. The agents then generate K additional geometric random variables in sequence, taking their sum. Upon completing the generation and propagation of the K'th number, the agent divides the sum by K and stores the result in their output field. Composition with a downstream protocol is as simple as letting that protocol be the last phase. However, since our protocol has a positive probability of failure, this would translate to the downstream protocol as well.

The time is $O(\log^2 n)$ by the following rough analysis (details follow). We propagate K numbers one after each other and for each epidemic $O(\log n)$ time is required. Since we set $k = O(\log n)$ then the protocol will take $O(\log^2 n)$ total time to complete.

5.2.2. Formal specification of protocol. Our protocol uses uniform random bits in multiple places. We assume agents have access to independent uniformly random bits. In the protocol, agents start by dividing in two groups of `S` and `A`. `A` agents are responsible for the most part of the algorithm including generating geometric random variables and propagating their maximums while the `S` agents only provide memory to store the sum of K maximum geometric random variables. We split the state space such that `A` agents and `S` agents are responsible to store different variables. The space multiplexing is a common approach used in population protocols to reduce the space complexity of the protocols [6].

Agents initially have no role (`X`), and partition into roles via $\mathtt{X}, \mathtt{X} \to \mathtt{A}, \mathtt{S}$. Since this takes $\Theta(n)$ time to complete, we add transitions $\mathtt{A}, \mathtt{X} \to \mathtt{A}, \mathtt{S}$ and $\mathtt{S}, \mathtt{X} \to \mathtt{S}, \mathtt{A}$, converging in $O(\log n)$ time, with the price of deviating from $\frac{n}{2}$ for each role. By theorem 5.2.2 this deviation is $O(\sqrt{n \ln n})$,

increasing the size estimation error by merely a constant additive factor.[10] All agents start at $\texttt{epoch} = 0$. The A agents generate one geometric random variable (called logSize2) and continue by propagating the maximum among the whole population. Since we use this logSize2 value for all early estimation of $\log n$, each time an agent finds out there was a greater value for the logSize2 than its own, it will reset all other computations that might have happened.

By theorem 5.2.8, the maximum logSize2 amongst the population is a 2 factor estimation of $\log n$. When any agent updates its logSize2 with a new maximum, it restarts the entire downstream protocol via Restart. Once the maximum logSize2 value is generated in the population, it propagates (triggering Restart) by epidemic in $O(\log n)$ time. The logSize2 variable could be used to estimate K, which is the number of independent additional geometric random variables each agent will generate. We also use logSize2 to set the leaderless phase clock inside each agent. In each epoch, the A agents will generate one new geometric random variable and propagate its maximum. They count their number of interactions in each epoch using the time variable. At the end of an epoch, when time reaches 95.logSize2, the A agents accumulate the value of the maximum gr into the sum of a S agent. The A agents increase their epoch variable by one and set $\texttt{time} = 0$ after either passing the geometric random variable to a S agent or interacting with a S agent in a higher phase. Separately, S agents are responsible to propagate the maximum sum and maximum epoch among themselves.

In the Log-Size-Estimation protocol, all agents in role A will finally generate $K = 5 \cdot \texttt{logSize2}$ geometric random variable and let the S agents to store a sum of maximum one generated for each phase. Once all agents reach $\texttt{epoch} = 5 \cdot \texttt{logSize2}$ they set $\texttt{protocolDone} = \text{True}$ and $\texttt{output} = \frac{\texttt{sum}}{\texttt{epoch}} + 1$. We use $|\texttt{A}|$, $|\texttt{S}|$ for the cardinality of A and S agents respectively.

The following lemma shows that close to half of agents end up in role A.

Lemma 5.2.2. *Let $a > 0$. In the* Log-Size-Estimation *protocol the cardinality of agents with* A *role is in the interval of $\left[\frac{n}{2} - a, \frac{n}{2} + a\right]$ with probability $\geq 1 - e^{-2a^2/n}$.*

Proof. All agents in the Log-Size-Estimation protocol start in role X. In the Partition-Into-A/S protocol agents will be assigned to their new roles. Finally, all agents participate in the

[10] This mechanism of splitting the population approximately in two works for our protocol, because the number of A agents is likely to be so close to $n/2$ that our estimate of $\log n$ is reduced by an additive factor likely to be very close to -1.

Protocol 23 Log-Size-Estimation(rec, sen)

▷ initial state of agent:
`role` = `X`,
`time` = 0, `sum` = 0, `epoch` = 0,
`gr` = 1, `logSize2` = 1,
`protocolDone` = False
Partition-Into-A/S(rec, sen)
if rec.`role` = `A` **then**
rec.`time` ← rec.`time` + 1
Check-if-Timer-Done-and-Increment-Epoch(rec)
if sen.`role` = `A` **then**
sen.`time` ← sen.`time` + 1
Check-if-Timer-Done-and-Increment-Epoch(sen)
Propagate-Max-Clock-Value(rec, sen)
Propagate-Incremented-Epoch(rec, sen)
if one agent have `role` = `S` and one have `role` = `A` **then**
Update-Sum(rec, sen)
if both agents have `role` = `A` **then**
Propagate-Max-G.R.V.(rec, sen)
if sen.`protocolDone` **then**
`output` ← $\frac{\texttt{sum}}{\texttt{epoch}} + 1$

Subprotocol 24 Partition-Into-A/S(rec, sen)

▷ Partition the population in two almost equal size subpopulations.
if sen.`role` = `X`, rec.`role` = `X` **then**
sen.`role` ← `A`
sen.`logSize2` ← one geometric random variable
rec.`role` ← `S`
else if sen.`role` = `A`, rec.`role` = `X` **then**
rec.`role` ← `S`
else if sen.`role` = `S`, rec.`role` = `X` **then**
rec.`role` ← `A`
rec.`logSize2` ← one geometric random variable

Subprotocol 25 Propagate-Max-Clock-Value(`agent1`, `agent2`)

▷ Maximum generated geometric variable for `logSize2` will be propagated.
if `agent1.logSize2` < `agent2.logSize2` **then**
`agent1.logSize2` ← `agent2.logSize2`
Restart(`agent1`)
else if `agent2.logSize2` < `agent1.logSize2` **then**
`agent2.logSize2` ← `agent1.logSize2`
Restart(`agent2`)

Log-Size-Estimation protocol either having role `A` or `S`. Thus, after completion of the Partition-Into-A/S protocol $|\texttt{A}| + |\texttt{S}| = n$ holds.

Subprotocol 26 RESTART(agent)

`time` $\leftarrow 0$, `sum` $\leftarrow 0$, `epoch` $\leftarrow 0$
`gr` $\leftarrow$ one geometric random variable
`protocolDone` $\leftarrow$ FALSE

Subprotocol 27 PROPAGATE-MAX-G.R.V.(agent1, agent2)

▷ Maximum generated geometric variable for `gr` will be propagated.
if agent1.epoch = agent2.epoch **then**
  if agent1.gr < agent2.gr **then**
    agent1.gr $\leftarrow$ agent2.gr
  else if agent2.gr < agent1.gr **then**
    agent2.gr $\leftarrow$ agent1.gr

Subprotocol 28 CHECK-IF-TIMER-DONE-AND-INCREMENT-EPOCH(agent)

▷ Agents compare their `time` value to the specified threshold.
if agent.time = $95 \times$ agent.logSize2, agent.protocolDone = FALSE, agent.updatedSUM = TRUE **then**
  agent.epoch $\leftarrow$ agent.epoch + 1
  MOVE-TO-NEXT-G.R.V(agent)
if agent.epoch = $5 \times$ agent.logSize2 **then**
  agent.protocolDone $\leftarrow$ TRUE

Subprotocol 29 PROPAGATE-INCREMENTED-EPOCH(agent1, agent2)

▷ The maximum `epoch` will be propagated.
if both agents have `role` = `A` **then**
  if agent1.epoch < agent2.epoch **then**
    agent1.epoch $\leftarrow$ agent2.epoch
    MOVE-TO-NEXT-G.R.V(agent1)
  else if agent2.epoch < agent1.epoch **then**
    agent2.epoch $\leftarrow$ agent1.epoch
    MOVE-TO-NEXT-G.R.V(agent2)
else if both agents have `role` = `S` **then**
  if agent1.epoch < agent2.epoch **then**
    agent1.epoch $\leftarrow$ agent2.epoch
    agent1.sum $\leftarrow$ agent2.sum
  else if agent2.epoch < agent1.epoch **then**
    agent2.epoch $\leftarrow$ agent1.epoch
    agent2.sum $\leftarrow$ agent1.sum

The percentage of agents that change to role `A` is an average of the percentage of `A`'s produced by the first rule (always exactly 1/2) and the percentage produced by the next two rules. The next two rules ensure that if the percentage of `A`'s so far produced is greater than 1/2, then `A` is less likely to be produced next than a fair coin flip (since, conditioned on the next interaction

Subprotocol 30 MOVE-TO-NEXT-G.R.V(`agent`)

▷ The `agent` move to the next `epoch`:
`agent.time` ← 0
`agent.gr` ← one geometric random variable
`agent.updatedSUM` = FALSE

Subprotocol 31 UPDATE-SUM(`agent1`, `agent2`)

▷ The `agent` accumulates the current value of `gr` in `sum`:
a ← agent with `role` = `A`
s ← agent with `role` = `S`
if a.`epoch` = s.`epoch`, a.`time` ≥ 95· a.`logSize2`, and a.`protocolDone` = FALSE **then**
  s.`epoch` ← s.`epoch` + 1
  s.`sum` ← s.`sum`+ a.`gr`
  a.`updatedSUM` = TRUE
else if a.`epoch` < s.`epoch` **then**
  a.`updatedSUM` = TRUE

being between one `X` and one non-`X`, the probability of producing `A` is exactly $\frac{|\mathtt{S}|}{|\mathtt{A}|+|\mathtt{S}|}$, i.e., smaller if there are more `A`'s than `S`'s), and vice versa. Thus, the distribution of the percentage difference $\left|\frac{|\mathtt{A}|}{n} - 1/2\right|$ of `A`'s is stochastically dominated by the difference between the percentage of heads of a fair-coin binomial distribution $\mathbf{B}(n, 1/2)$ and $\mathbf{B}$'s expected percentage of 1/2. Therefore we can use a binomial distribution to bound the upper and lower tails of the distribution of the number of eventual `A`'s. For any a the Chernoff bound says for any $a > 0$, $\Pr[\mathbf{B}(n, 1/2) \geq n/2 + a] \leq e^{-2a^2/n}$ and $\Pr[\mathbf{B}(n, 1/2) \leq n/2 - a] \leq e^{-2a^2/n}$. □

COROLLARY 5.2.3. *In the* LOG-SIZE-ESTIMATION *protocol the cardinality of agents with* `A` *role is in the interval of* $\left[\frac{n}{3}, \frac{2n}{3}\right]$ *with probability* $\geq 1 - e^{-n/18}$.

In each epoch, one geometric random variable (in the first epoch `logSize2` and in the subsequent epochs `gr`) is generated and its maximum will be propagated by epidemic among the population. We set the time of each epoch equal to the required time of generating one plus the time for completion of an epidemic. To analyze the time complexity of our protocol, we require the time bounds for completing an epidemic from the paper [12]. The current form is taken from [51]. For all $n \in \mathbb{N}^+$, let $H_n = \sum_{k=1}^{n} \frac{1}{k}$ denote the n'th harmonic number. Note that $\ln n \leq \frac{n-1}{n} H_{n-1} \leq 1 + \ln n$.

The following corollary describes an epidemic in a subpopulation. This refers to some subset S of the population executing epidemic transitions only among themselves, which slows down the epidemic by only a constant factor if $|S| = \Omega(n)$.

Corollary 5.2.4. *Let $c \geq 1$. Suppose an epidemic happens among a subpopulation of $a = n/c$ agents. Let $\mathbf{T}$ denote the time to complete such an epidemic. Then for any $\alpha_u > 0$, $\Pr[\mathbf{T} > \alpha_u \ln a] < a^{-(\alpha_u - 4c)^2/12c}$.*

Proof. The probability that in the next interaction the scheduler picks two agents from the subpopulation a is $\binom{a}{2}/\binom{n}{2} = \frac{a(a-1)}{n(n-1)}$. By theorem 1.3.5, if $\mathbf{T}'$ denotes the time to complete an epidemic in population size a, then $\mathbb{E}[\mathbf{T}'] = \frac{a-1}{a} H_{a-1}$. Since we have $\frac{n(n-1)}{a(a-1)}$ expected interactions in the whole population of size n in order to obtain one interaction within the subpopulation, the expected time to complete this epidemic (counting total interactions in the whole population) is $\mathbb{E}[\mathbf{T}] = \frac{n(n-1)}{a(a-1)} \cdot \mathbb{E}[\mathbf{T}'] = \frac{n(n-1)}{a(a-1)} \frac{a-1}{a} H_{a-1} = \frac{n(n-1)}{a^2} H_{a-1} \geq c^2 \ln a$. By the Chernoff bound we have:

$$\begin{aligned} \Pr[\mathbf{T} \geq (1+\delta)\mathbb{E}[\mathbf{T}]] &\leq e^{-\delta^2 \mathbb{E}[\mathbf{T}]/3} \\ &\leq e^{-\delta^2 \cdot c^2 \ln(a)/3} \\ &= a^{-c^2\delta^2/3} \end{aligned}$$

Setting $\alpha_u = 4c(1+\delta)$, $\Pr[\mathbf{T} \geq \alpha_u \ln a] \leq a^{-((\alpha_u/4c)-1)^2 c^2/3}$. □

Setting $c = 3$ and $\alpha_u = 24$ in theorem 5.2.4 gives the following.

Corollary 5.2.5. *Suppose an epidemic happens among a subpopulation of $n/3$ agents with time $\mathbf{T}$. Then $\Pr[\mathbf{T} > 24 \ln n] < 27 n^{-3}$.*

The next lemma bounds the number of interactions an agent has in a given time, and it is the basis of the leaderless phase clock we use. It follows from a simple Chernoff bound on the number of interactions involving a single agent in a given window of time.

Lemma 5.2.6. *Let $C \geq 3$ and $D = 2C + \sqrt{12C}$. In time $C \ln n$, with probability $\geq 1 - 1/n$, each agent has at most $D \ln n$ interactions.*

Proof. Fix an agent a. Let $\mathbf{I}_a$ be the number of interactions involving a during $Cn \ln n$ total interactions ($C \ln n$ time). The probability that any given interaction involves a (either receiver or sender) is exactly $\frac{2}{n}$, so $\mathbf{I}_a$ is distributed binomially, and $\mathbb{E}[\mathbf{I}_a] = Cn \ln n \cdot \frac{2}{n} = 2C \ln n$. Applying the Chernoff bound, for any $0 < \delta \leq 1$,

$$\Pr[\mathbf{I}_a \geq (1+\delta) 2C \ln n] \leq e^{-\delta^2 2C \ln(n)/3} = n^{-2C\delta^2/3}.$$

Let $D = 2C + \sqrt{12C}$, and let $\delta = \frac{D}{2C} - 1$. (Note $\delta \leq 1$ so long as $C \geq 3$.) Then $(1+\delta)2C \ln n = D \ln n$ and $n^{-2C\delta^2/3} = n^{\frac{-(D-2C)^2}{6C}} = n^{-2}$. Then $\Pr[\mathbf{I}_a \geq D \ln n] \leq n^{-2}$. By the union bound, $\Pr[(\exists a)\ \mathbf{I}_a \geq D \ln n] \leq n^{-1}$. So, by setting $D = 2C + \sqrt{12C}$ we bound the probability that each agent has more than $D \ln n$ interactions in time $C \ln n$ to be $\leq 1/n$. □

Corollary 5.2.7. *Each agent has* $\geq 65 \ln n$ *interactions in time* $24 \ln n$ *with probability* $\leq 1/n$.

By theorem 5.2.6 each agent has at most $\left(2 \cdot 24 + \sqrt{12 \cdot 24}\right) \ln n \leq 65 \ln n \leq 94 \log n$ interactions in the time that it takes to generate and propagate maximum of one geometric random variable. Thus, each agent should count up to $94 \log n$ for its leaderless phase clock, to ensure that with high probability none reaches that count until the maximum geometric random variable is known to all agents. However, agents are not aware of any prior approximation of $\log n$. In the Log-Size-Estimation protocol, agents use their `logSize2` variable for this approximation. As mentioned, all the agents in role `A` start by generating one geometric random variable `logSize2`. The maximum in the population is used as a weak (constant factor) approximation of $\log n$. theorem 5.7.7 says that the maximum of $|\mathtt{A}|$ geometric random variables is in the interval of $[\log |\mathtt{A}| - \log\ln |\mathtt{A}|, 2 \log |\mathtt{A}|]$ with probability at least $1 - 1/|\mathtt{A}|$. However, we are using the `logSize2` and `gr` variables as an approximation of $\log n$ rather than $\log |\mathtt{A}|$. theorem 5.2.8 will give us a bound over the `logSize2` value with respect to $\log n$. theorem 5.2.9, theorem 5.2.10, and theorem 5.2.11 use this lemma for a bound over `gr`, `time`, and `epoch` values.

Lemma 5.2.8. *The* `logSize2` *value generated by* Generate-Clock *is in the interval of* $[\log n - \log\ln n, 2 \log n + 1]$ *with probability at least* $1 - 1/n - e^{-n/18}$.

Proof. By theorem 5.2.3, $n/3 \leq |\mathtt{A}| \leq 2n/3$ with probability $\geq 1 - e^{-n/18}$. We apply the result of theorem 5.7.7 and substitute n with $|\mathtt{A}|$:

$$\log(n/3) - \log\ln(n/3) \leq \texttt{logSize2} \leq 2\log(2n/3)$$

$$\log n - 1.6 - \log\ln n - 0.4 \leq \texttt{logSize2} \leq 2\log n + 2\log(2/3)$$

$$\log n - \log\ln n - 2 \leq \texttt{logSize2} \leq 2\log n - 1$$

$$\log n - \log\ln n - 2 \leq \texttt{logSize2} \leq 2\log n - 1$$

So, by setting `logSize2` = `logSize2` + 2 for all agents, this variable is in the interval of $[\log n - \log\ln n, 2\log n + 1]$ with probability $\geq 1 - 1/n - e^{-n/18}$. □

Corollary 5.2.9. *The* `logSize2` *(*`gr`*) value generated by* Generate-Clock *(*Generate-G.R.V*) is in the interval of* $[\log n - \log\ln n - 2, 2\log n - 1]$ *with probability at least* $1 - 1/n - e^{-n/18}$.

Corollary 5.2.10. *The number of interactions in each epoch in the* Log-Size-Estimation *is in the interval* $[95\log n - 95\log\ln n, 189\log n]$ *with probability* $\geq 1 - 1/n - e^{-n/18}$.

Proof. By theorem 5.2.7, agents should count up to $96\ln \leq 139\log n$ before moving to the next epoch. if we set the threshold of the `time` to 95·`logSize2`, $95\log n - 95\log\ln n \geq 93\log n$ then the `time` variable will be in the interval of $[95\log n - 95\log\ln n, 188\log n + 95]$ with high probability ($188\log n + 95 \leq 189\log n$ for $n \geq 2$). □

Corollary 5.2.11. *The number of epochs in the* Log-Size-Estimation *is in the interval* $[5\log n - 5\log\ln n, 11\log n]$ *with probability* $\geq 1 - 1/n - e^{-n/18}$.

Proof. By theorem 5.7.10, to achieve the additive error of 4.7 for our protocol the number of geometric random variables should be $\geq 4\log n$. By setting the threshold of the number of phases to $5 \times$ `logSize2`, for $n \geq 200$, $5\log n - 5\log\ln n \geq 4\log n$. The number of phases will be in the interval of $[5\log n - 5\log\ln n, 10\log n + 5]$ with high probability ($10\log n + 5 \leq 11\log n$ for $n \geq 2$). □

The next Lemma bounds the space complexity of our main protocol by counting the likely range taken by the variables in Log-Size-Estimation.

Lemma 5.2.12. Log-Size-Estimation *uses* $O(\log^4 n)$ *states with probability* $\geq 1 - O(\log n)/n$.

Proof. With probability at least $1 - O(1/n)$ (see individual lemma statements for constants in the O), the set of values possibly taken on by each field are given as follows:

`logSize2`	$\{1, \ldots, 2\log n + 1\}$	theorem 5.2.8
`gr`	$\{1, \ldots, 2\log n\}$	theorem 5.2.9
`time`	$\{0, \ldots, 191\log n\}$	theorem 5.2.10
`epoch`	$\{0, \ldots, 11\log n\}$	theorem 5.2.11
`sum`	$\{0, \ldots, 22\log^2 n\}$	theorem 5.2.9, theorem 5.2.11

In our protocol we used space multiplexing to reduce the number of states agents use. The `A` agents are responsible to generate geometric random variables and propagate the maximum among themselves. Thus, they store `logSize2`, `gr`, `time`, and `epoch` variables. While the `S` agents are only responsible to hold the sum of all geometric maximas and they store `logSize2`, `epoch`, and `sum`. After each agent sets `protocolDone` = True, it no longer needs to store the value in `gr` or `epoch` and can use that space to store the result of `sum`/`epoch` $+ 1$ as the output. Although we are using the explained space multiplexing to reduce the number of states used by the agents, both `A` and `S` agents need to store `logSize2` and `epoch` to stay synchronized. Note that the probability that each geometric random variable is greater than $2 \log n$ is less than $1/n$, by the union bound the probability that any of them is greater than $2 \log n$ is less than $\frac{11 \log n}{n}$. □

The next corollary bounds the time complexity of protocol Log-Size-Estimation; the main component of the time complexity is that $\Theta(\log n)$ geometric random variables must be generated and propagated by epidemic among the population, each epidemic taking $\Theta(\log n)$ time.

Corollary 5.2.13. *The* Log-Size-Estimation *protocol converges in* $O(\log^2 n)$ *time with probability at least* $1 - 1/n^2$.

Proof. By theorem 5.2.5, with probability $\geq 1 - (27/n^3)$ propagating the maximum of the `logSize2` variable takes at most $24 \ln n$ time.

By theorem 5.2.11, at most $11 \log n$ geometric random variables will be generated, and by theorem 5.2.5, with probability $\geq 1-(27/n^3)$ a given variable takes at most $24 \ln n$ time to propagate its maximum (total of $11 \log n \cdot 24 \ln n$ time). Note that, it takes constant time for `A` agents to interact with a `S` agent and move to the next `epoch`.

By the union bound over all epochs, the probability that generating $11 \log n + 1$ geometric random variables and propagating their maximum takes more than $(11 \log n + 1) \cdot 24 \ln n$ time is $\geq 1 - \frac{O(\log n)}{n^3} \geq 1 - 1/n^2$ for large values of n. □

The following result is a Chernoff bound on sums of random variables, each of which is the maximum of independent geometric random variables (with probability of success $\frac{1}{2}$). It is a corollary of theorem 5.7.10.

LEMMA 5.2.14. *Let $K \geq 4\log n$ and a be a number in the interval of $[n/2 - \sqrt{n\ln n}, n/2 + \sqrt{n\ln n}]$. Let $\texttt{sum}/K$ be the average of K $\frac{1}{2}$-geometric random variables. Then* $\Pr\left[\left|\frac{\texttt{sum}}{K} + 1 - \log n\right| \geq 5.7\right] \leq \frac{6}{n}$.

PROOF. By theorem 5.7.10, $\Pr\left[\left|\frac{\texttt{sum}}{K} - \log a\right| \geq 4.7\right] \leq \frac{2}{a}$. Since $n/2 - \sqrt{n\ln n} \leq a \leq n/2 + \sqrt{n\ln n}$ then $\log n - 2 \leq \log a \leq \log n$. Hence: $\Pr\left[\left|\frac{\texttt{sum}}{K} + 1 - \log n\right| \geq 4.7 + 1\right] \leq \frac{2}{a} \leq \frac{6}{n}$. □

LEMMA 5.2.15. *In the* LOG-SIZE-ESTIMATION *protocol, with probability* 1*, all agents converge to the same value C in their* `output` *field. Furthermore,* $\Pr[|C - \log n| \geq 5.7] \leq 9/n$.

PROOF. The convergence of all agents to a common value of C with probability 1 is evident from inspection of the protocol. The LOG-SIZE-ESTIMATION protocol calculate $\log n$ within an additive error of 5.7 if:

- The `logSize2` variable generated at the beginning of the protocol is $\geq \log n - \log\ln n$ (not too small). By theorem 5.2.8, the value of `logSize2` can be less than $\log n - \log\ln n$ with probability at most $1/n + e^{-n/18}$.
- The number of agents in role `A` is less than $n/2 - \sqrt{n\ln n}$ or greater than $n/2 + \sqrt{n\ln n}$. By theorem 5.2.2, $a \leq \frac{n}{2} - \sqrt{n\ln n}$ or $a \geq \frac{n}{2} + \sqrt{n\ln n}$ with probability at most $e^{-2\ln n} = 1/n^2$.
- The `A` agents propagate each geometric random variables among themselves within $24\ln n$ time. By theorem 5.2.5, an epidemic might take more than $24\ln n$ time with probability at most $\frac{27}{n^3}$. In the LOG-SIZE-ESTIMATION protocol there are $O(\log n)$ epidemics in total. Thus, by the union bound the probability that any of them take more than $24\ln n$ time is at most $O(\log n)/n^3 \leq 1/n^2$ for large values of n.
- An `epoch` terminates before completion of one epidemic. This can happen if one agent has too many interaction in an `epoch`. By theorem 5.2.7, for all agents, the probability that any of them have more than $65\ln n$ interaction in $24\ln n$ time is $\leq 1/n$.
- By theorem 5.2.14, the average of K geometric random variables among `A` agents might be out of the interval of $[\log n - 5.7, \log n + 5.7]$ with probability at most $6/n$.

By the union bound, the probability that `output` reports a value with an additive error more than 5.7 is less than $8/n + 2/n^2 + e^{-n/18} \leq 9/n$ □

Finally, we combine these results to prove the main result of this section, theorem 5.2.1.

THEOREM 5.2.16. *There is a uniform leaderless population protocol that converges in time* $O(\log^2 n)$ *with probability* $\geq 1 - 1/n^2$*, uses* $O(\log^4 n)$ *states with probability* $\geq 1 - O(\log n)/n$*, and stores in each agent an integer* k *such that* $|k - \log n| \leq 5.7$ *with probability* $\geq 1 - 9/n$*.*

PROOF. By theorem 5.2.13 LOG-SIZE-ESTIMATION take $O(\log^2 n)$ time to converge with probability at least $1 - 1/n^2$. By theorem 5.2.12, LOG-SIZE-ESTIMATION protocol uses $O(\log^4 n)$ states with probability at least $1 - O(\log n)/n$.

Finally, theorem 5.2.15 guaranties the `output` obtained by `A` agents is with an additive error 5.7 of $\log n$ with probability at least $1 - 9/n$.

□

5.2.3. Probability-1 estimation of upper bound on $\log n$**.** It is not clear how to make our main protocol correct with probability 1, meaning that it guarantees the estimate k obeys $|k - \log n| \leq 5.7$. The protocol could err in either direction and make k too large or too small, depending on the sampled values of the geometric random variables.

However, for many applications using an estimate of $\log n$, an *upper* bound is sufficient to ensure correctness (though being too large may slow things down). A straightforward modification of our protocol guarantees that $k \geq \log n$ with probability 1, while preserving the high-probability asymptotic time complexity. We run a slow, exact backup protocol that stabilizes to k_{ex} such that $2^{k_{\text{ex}}-1} < n \leq 2^{k_{\text{ex}}}$. This is accomplished by transitions $\ell_i, \ell_i \to \ell_{i+1}, f_{i+1}$ for all i and $f_i, f_j \to f_i, f_i$ for all $j < i$, where all agents start with ℓ_0. After $O(n)$ time all agents store k_{ex} in their subscript. Note this approaches k_{ex} from below. Then modify our main protocol estimating $\log n$ to add 3.7 (with probability $\geq 1 - 2 \cdot e^{\kappa/2 - t/4}$ where κ is the number of geometric random variables; by theorem 5.7.4, and setting $\alpha = 1$) in theorem 5.7.8 since we can leverage the theorem 5.7.6 for one side error) to its estimate of $\log n$, calling the result k; with high probability $k \geq \log n$. We then get a guaranteed upper bound on $\log n$ by reporting $\max(k, k_{\text{ex}})$ at any moment: the former converges to $k \geq \log n$ with probability of failure $O(\log(n)/n)$. If this fails (i.e., if $k < \log n$), the value k_{ex} is guaranteed eventually to exceed k. The contribution to the expected time of the latter case is negligible, so the expected convergence time remains $O(\log^2 n)$.

Note that k may exceed $\log n$ by an arbitrary amount, with low probability. In the terminology of section 5.1.2, we have changed the definition of "correct" from "$|k = \log n| \leq 5.7$" to "$k \geq \log n$",

and showed probability 1 of correctness under the new definition. (Note that we still guarantee that $k \leq \log n + O(1)$ with high probability, where the $O(1)$ constant is now $5.7 + 3.7 = 9.4$.) An interesting open question is to find a polylogarithmic protocol that guarantees k is within $O(1)$ of $\log n$ with probability 1.

5.3. Tools for making nonuniform protocols uniform

Part of the practical motivation behind the study of the counting problem comes from the existence of nonuniform protocols and a desire to create uniform variants of them. Since most nonuniform protocols require advance knowledge of $\log n$, the basic technique for making such a protocol uniform is to first compute an estimate of $\log n$ using a protocol from section 2.4, then to compose this with the existing nonuniform protocol, replacing its estimate of $\log n$ with this computed value. We break this section into two pieces. First, we recall a few size approximation protocols from section 2.4, comparing them in accuracy, space, and simplicity with specific suggestions for how to account for these properties in choosing one to be composed with a nonuniform protocol. In the next part, assuming we have a protocol that computes an estimate of the population size, we show how to compose it with a nonuniform protocol.

5.3.1. Comparison of approximate counting protocols. In this part, we compare techniques that solve the approximate counting problem. See table 2.3 for a detailed comparison. For most nonuniform protocols (e.g., a leaderless phase clock [**3**]), a value that is $\Theta(\log n)$ suffices. Thus we lead the discussion with protocols that approximate $\log n$ within a multiplicative factor error. Although additive approximate error is often unnecessary, in some circumstances, one may require the estimate be to exactly $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$. For example, the uniform majority protocol of [**50**] requires the estimate to be at least $\lceil \log n \rceil$ with probability 1.

Simple 2-approximation: [section 2.4.2] Taking the maximum of n geometric random variables provides a 2-factor approximation of $\log n$ [**2**,**47**,**56**] with probability at least $1 - O(1/n)$,[11] and takes $O(\log n)$ time to converge. Although this approach is very simple and straightforward for composition, the space complexity of the protocol is not bounded with probability 1, and the agents use $O(\log n)$ states WHP.

[11] In fact the lower bound is stronger: the maximum is between $\log_2 n - \log_2(n) \cdot \ln n$ and $2 \cdot \log_2 n$ with probability at least $1 - O(1/n)$; see [**47**, Lemma 3.8].

Minimal space overhead: [section 2.4.3] Recall that the maximum level l^* in the junta election protocol is $\lfloor \log\log n \rfloor - 3 \leq l^* \leq \log\log n + 4(a+1)$ with probability at least $1 - 1/n^a$ [25,60]. Despite the large multiplicative approximation factor for computing $\lceil \log n \rceil$, the junta election protocol [60] imposes minimal, $O(\log\log n)$, space overhead and converges in $O(\log n)$ time. Moreover, the protocol provides a junta of n^ϵ for $0 \leq \epsilon < 1$ leaders that can simulate a "junta-driven phase clock" to synchronize agents in phases of length $\Theta(\log n)$ time [60]. See [6, 12] for more details about the phase clock.

Maximizing accuracy, always correct: Two protocols from section 2.4.1 and section 2.4.6 compute $\lfloor \log n \rfloor$ (or $\lceil \log n \rceil$). Both protocols provide probability-1 correctness using $O(n \log n)$ and $O(\log^2 n)$ time respectively. The former is much simpler and is used as a "slow backup" subroutine in the $O(\log n)$ time protocol of [50]. Although it is much slower than $O(\log n)$ time, since it is needed only with low probability, it contributes negligibly to the expected time.

5.3.2. Composition of an uniform counting protocol with a nonuniform protocol. Most of the time, we can construct a uniform protocol from a nonuniform protocol through composition with a uniform approximate counting protocol. Even though we are unaware of any black-box theorem that proves the correctness of the restarting technique under any circumstance, the authors of [28,47,50,51,60] used the procedure discussed below and proved it correct with an ad-hoc analysis. We explain in a general way how to use these approximate counting protocols to make a nonuniform protocol uniform in the next part.

Note that all of the approximate counting protocols mentioned in section 2.4, except approximating with a leader explained in section 2.4.5, are not terminating. In other words, the agents are not aware of the completion of the protocol. Although termination is impossible in the uniform model of population protocols [47], we can try composing two protocols without using termination of the upstream ones. For a concrete discussion, consider protocols **U** and **D** such that **U** is a uniform approximate counting and **D** is a general nonuniform protocol; **U** is the *upstream* protocol whose output is given as input to the *downstream* protocol **D**. To construct a uniform protocol, we summarize a simple restarting technique that has been used widely [28, 51, 60] to compose protocols **U** and **D**. In this technique, we run both protocols in parallel in the population. If the `count` fields of the protocol **U** in the agents' memory change, then a signal will be propagated (by

epidemic) through the population to notify all agents with the updated `count`. This signal will stop the protocol **D** (or parts of protocol **D** that are dependent on the population size) and reinitialize it with the updated `count`, which eventually will be an approximation of $\log n$.

Despite the difference between the initial configuration of a nonuniform protocol and the one after the restart signal (agents do not have the same state), any agents who participate in the last execution of protocol **D** restarts their memory (related fields concerning protocol **D**) to the initial values. Thus, a high probability correct counting scheme can also guarantee the correctness of the downstream protocol **D** WHP.

5.4. Simulation results

5.5. Simulation

Simulation results are shown in fig. 5.2.

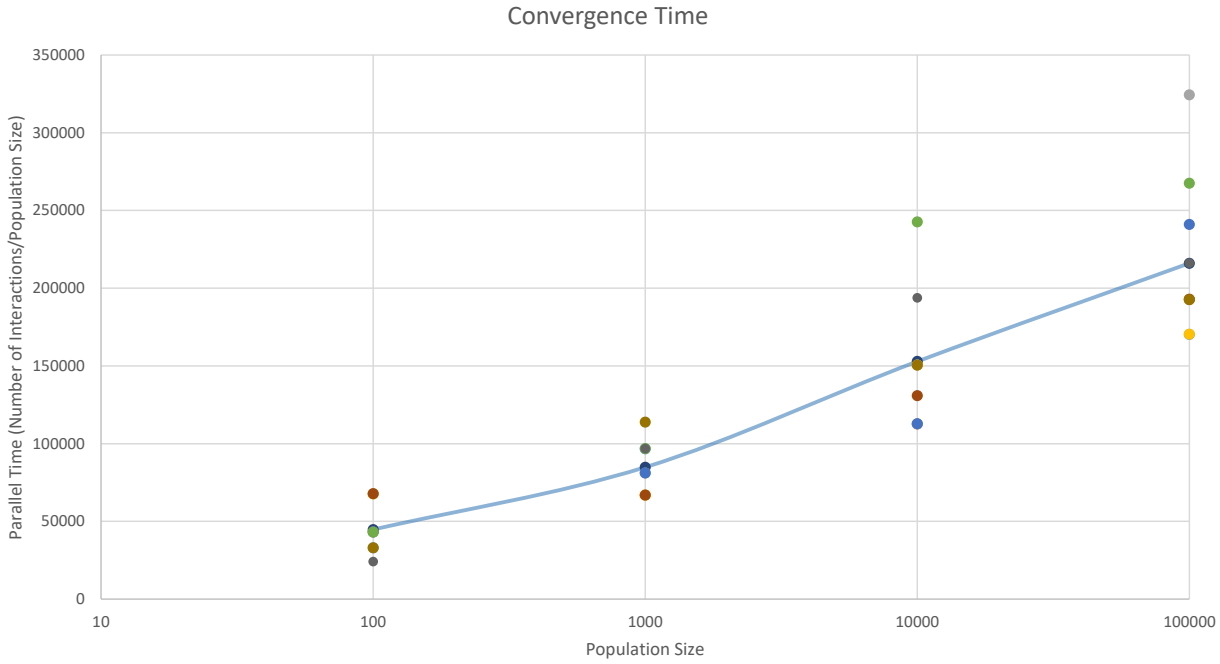


FIGURE 5.2. Simulated convergence time of the protocol. Although the proofs give only that the estimate of $\log n$ is likely to get within additive error of 5, in practice the estimate is always within 2, so this is how we define convergence in the experiment. The dots indicate the convergence time of individual experiments. The population size axis is logarithmic (i.e., exactly $O(c \log_{10} n)$ time complexity would correspond to a straight line with slope c). The circular dots in the plot are 10 experiments at each value of $n \in \{10^2, 10^3, 10^4, 10^5\}$. The convergence in the LOG-SIZE-ESTIMATION protocol happens when all agents reach $\texttt{epoch} = 5 \cdot \texttt{logSize2}$.

5.6. Proofs for correctness of size estimation protocol

This section contains proofs of lemmas required to analyze the correctness and time/space complexity of the size estimation protocol.

5.7. Chernoff bound on sums of maxima of geometric random variables

This section is aimed at proving theorem 5.7.10, a Chernoff bound on the tails of K independent random variables, each of which is the maximum of N independent geometric random variables with success probability $1/2$. When applied to prove correctness of the protocol of section 5.2.2, N is a value very likely to be near $n/2$, i.e., half the population size.

5.7.1. Sub-exponential random variables.

DEFINITION 5.7.1. *Let $\alpha, \beta > 0$ and let $\mathbf{X}$ be a random variable. We say $\mathbf{X}$ is* α-β-sub-exponential *if, for all $\lambda > 0$, $\Pr[|\mathbf{X} - \mathbb{E}[\mathbf{X}]| \geq \lambda] \leq \alpha e^{-\lambda/\beta}$.*

The following lemma is well-known; we prove it explicitly since the exact form is convenient for our purposes but is more general than typically expressed. It shows that exponential tail bounds for $\Pr[|\mathbf{X} - \mathbb{E}[\mathbf{X}]| > \lambda]$ give bounds on the moment-generating functions of the random variables $\mathbf{X} - \mathbb{E}[\mathbf{X}]$ and $\mathbb{E}[\mathbf{X}] - \mathbf{X}$. The proof is modeled on Rigollet's proof of the analogous lemma for sub-gaussian random variables proven in [85].

LEMMA 5.7.2 ([85]). *Let $\mathbf{X}$ be a α-β-sub-exponential random variable. Then for all $s \in \left[-\frac{1}{2\beta}, \frac{1}{2\beta}\right]$, we have $\mathbb{E}\left[e^{s(\mathbf{X}-\mathbb{E}[\mathbf{X}])}\right], \mathbb{E}\left[e^{s(\mathbb{E}[\mathbf{X}]-\mathbf{X})}\right] \leq 1 + 2\alpha\beta^2 s^2$.*

PROOF. Let $k \in \mathbb{N}^+$. Then

$$\begin{aligned}
\mathbb{E}\left[|\mathbf{X} - \mathbb{E}[\mathbf{X}]|^k\right] &= \int_0^\infty \Pr\left[|\mathbf{X} - \mathbb{E}[\mathbf{X}]|^k \geq \lambda\right] d\lambda = \int_0^\infty \Pr\left[|\mathbf{X} - \mathbb{E}[\mathbf{X}]| \geq \lambda^{1/k}\right] d\lambda \\
&\leq \int_0^\infty \alpha e^{-\lambda^{1/k}/\beta} d\lambda = \alpha\beta^k k \int_0^\infty e^{-u} u^{k-1} du \qquad \text{substituting } u = \beta\lambda^{1/k} \\
&= \alpha\beta^k k\Gamma(k) = \alpha\beta^k k!,
\end{aligned}$$

where $\Gamma(k) = \int_0^\infty e^{-u}u^{k-1}du$ is the *gamma function*, known to equal $(k-1)!$ for $k \in \mathbb{N}^+$. Then for all $s \in \left[-\frac{1}{2\beta}, \frac{1}{2\beta}\right]$,

$$
\begin{aligned}
\mathbb{E}\left[e^{s(\mathbf{X}-\mathbb{E}[\mathbf{X}])}\right] &= \mathbb{E}\left[\sum_{k=0}^{\infty}\frac{(s(\mathbf{X}-\mathbb{E}[\mathbf{X}]))^k}{k!}\right] && \text{Taylor expansion of the exponential function}\\
&= \sum_{k=0}^{\infty}\frac{s^k\mathbb{E}\left[(\mathbf{X}-\mathbb{E}[\mathbf{X}])^k\right]}{k!} && \text{dominated convergence theorem}\\
&= 1+s\underbrace{\mathbb{E}[\mathbf{X}-\mathbb{E}[\mathbf{X}]]}_{=0}+\sum_{k=2}^{\infty}\frac{s^k\mathbb{E}\left[(\mathbf{X}-\mathbb{E}[\mathbf{X}])^k\right]}{k!}\\
&\le 1+\sum_{k=2}^{\infty}\frac{|s|^k\mathbb{E}\left[|\mathbf{X}-\mathbb{E}[\mathbf{X}]|^k\right]}{k!} && \text{odd terms can only get larger}\\
&\le 1+\sum_{k=2}^{\infty}\frac{|s|^k\alpha\beta^k k!}{k!}=1+\alpha\sum_{k=2}^{\infty}(|s|\beta)^k=1+\alpha s^2\beta^2\sum_{k=0}^{\infty}(|s|\beta)^k\\
&\le 1+\alpha\beta^2 s^2\sum_{k=0}^{\infty}\frac{1}{2^k} && \text{since } |s|\le\frac{1}{2\beta}\\
&= 1+2\alpha\beta^2 s^2.
\end{aligned}
$$

The bound for $\mathbb{E}\left[e^{s(\mathbb{E}[\mathbf{X}]-\mathbf{X})}\right]$ is derived by a similar argument. □

The following Chernoff bound is well-known, but stated in a more convenient form for our purposes.

LEMMA 5.7.3. *Let $\alpha,\beta>0$ and $K\in\mathbb{N}^+$. Let $\mathbf{X}_1,\ldots,\mathbf{X}_K$ be i.i.d. α-β-sub-exponential random variables. Define $\mathbf{S}=\sum_{i=1}^{K}\mathbf{X}_i$. Then for all $t\ge 0$,*

$$\Pr\left[|\mathbf{S}-\mathbb{E}[\mathbf{S}]|\ge t\right]\le 2\frac{(1+\alpha/2)^K}{e^{t/(2\beta)}}.$$

PROOF. Then for all $s, t > 0$,

$$\begin{aligned}
\Pr[\mathbf{S} - \mathbb{E}[\mathbf{S}] > t] &= \Pr\left[e^{s(\mathbf{S}-\mathbb{E}[\mathbf{S}])} > e^{st}\right] \\
&\leq \frac{\mathbb{E}\left[e^{s(\mathbf{S}-\mathbb{E}[\mathbf{S}])}\right]}{e^{st}} && \text{Markov's inequality} \\
&= e^{-st}\mathbb{E}\left[e^{s\left(\left(\sum_{i=1}^{K}\mathbf{X}_i\right)-\mathbb{E}[\mathbf{S}]\right)}\right] \\
&= e^{-st}\mathbb{E}\left[e^{s\sum_{i=1}^{K}(\mathbf{X}_i-\mathbb{E}[\mathbf{X}_i])}\right] && \text{linearity of expectation} \\
&= e^{-st}\mathbb{E}\left[\prod_{i=1}^{K} e^{s(\mathbf{X}_i-\mathbb{E}[\mathbf{X}_i])}\right] \\
&= e^{-st}\prod_{i=1}^{K}\mathbb{E}\left[e^{s(\mathbf{X}_i-\mathbb{E}[\mathbf{X}_i])}\right]. && \text{independence of the } \mathbf{X}_i\text{'s}
\end{aligned}$$

By theorem 5.7.2, for all $|s| \leq \frac{1}{2\beta}$, $\mathbb{E}\left[e^{s(\mathbf{X}_i-\mathbb{E}[\mathbf{X}_i])}\right] \leq 1 + 2\alpha\beta^2 s^2$, so letting $s = \frac{1}{2\beta}$,

$$\Pr[\mathbf{S} - \mathbb{E}[\mathbf{S}] > t] \leq e^{-st}\left(1 + 2\alpha\beta^2 s^2\right)^K = e^{-t/(2\beta)}\left(1 + \alpha/2\right)^K.$$

The proof that $\Pr[\mathbb{E}[\mathbf{S}] - \mathbf{S} \geq t] < e^{-t/(2\beta)}\left(1 + \alpha/2\right)^K$ is symmetric. By the union bound, $\Pr[|\mathbf{S} - \mathbb{E}[\mathbf{S}]| \geq t] < 2 \cdot e^{-t/(2\beta)}\left(1 + \alpha/2\right)^K$. □

5.7.2. Geometric random variables and their maximum. We say $\mathbf{G}$ is a *p-geometric random variable* if it is the number of consecutive flips until the first H (including the H), when flipping a coin with $\Pr[H] = p$. Thus $\mathbb{E}[\mathbf{G}] = \frac{1}{p}$; in particular $\mathbb{E}[\mathbf{G}] = 2$ if $p = \frac{1}{2}$.

Defining $\mathbf{M} = \max_{1 \leq i \leq N} \mathbf{G}_i$, where each $\mathbf{G}_i$ is an i.i.d. $\frac{1}{2}$-geometric random variable, it is known [56] that $\mathbb{E}[\mathbf{M}] \approx \log N$. theorem 5.7.5 shows a tail bound on $\mathbf{M}$ for general p-geometric random variables, which we will later apply to the case $p = \frac{1}{2}$.

We first require a technical lemma relating $\mathbb{E}[\mathbf{M}]$ and $\log N$ more precisely, and more generally for p-geometric random variables for $p \neq \frac{1}{2}$. Let $H_n = \sum_{i=1}^{N} \frac{1}{N}$ be the N'th harmonic number. Let $\gamma = \lim_{N\to\infty}(H_n - \ln N) \approx 0.577$ be the Euler-Mascheroni constant; for all $N \geq 50$ we have $H_n - \ln N - \gamma \leq 0.01$.

LEMMA 5.7.4. *Let $\mathbf{G}_1, \ldots, \mathbf{G}_N$ be i.i.d. p-geometric random variables with $q = 1 - p \geq \frac{1}{e}$, $N \geq 50$, and let $\mathbf{M} = \max_{1 \leq i \leq N} \mathbf{G}_i$. Let $\epsilon_1 = 0.01$ and $\epsilon_2 = 0.0006$. Then for all $\lambda > 0$, $\frac{\ln(N)+\gamma}{\ln 1/q} + 1/2 -$*

$\epsilon_2 < \mathbb{E}[\mathbf{M}] < \frac{\ln(N)+\gamma+\epsilon_1}{\ln 1/q} + 1/2 + \epsilon_2$; *particularly for* $q = p = 1/2$, *we have:* $\log N + 1 < \mathbb{E}[\mathbf{M}] < \log N + 3/2$.

PROOF. Eisenberg [56] showed that if $q \geq \frac{1}{e}$, then $\frac{1}{\lambda}H_n - 0.0006 \leq \mathbb{E}[\mathbf{M}] - 1/2 < \frac{1}{\lambda}H_n + 0.0006$, where $q = e^{-\lambda}$, i.e. $\lambda = \ln(1/q)$. Thus $\frac{1}{\lambda}H_n + 1/2 - \epsilon_2 \leq \mathbb{E}[\mathbf{M}] < \frac{1}{\lambda}H_n + 1/2 + \epsilon_2$ i.e., $\frac{\ln N+\gamma}{\ln 1/q} + 1/2 - \epsilon_2 < \mathbb{E}[\mathbf{M}] < \frac{\ln N+\gamma+\epsilon_1}{\ln 1/q} + 1/2 + \epsilon_2$. □

LEMMA 5.7.5. *Let* $\mathbf{G}_1, \ldots, \mathbf{G}_N$ *be i.i.d. p-geometric random variables with* $q = 1 - p \geq \frac{1}{e}$, $N \geq 50$, *and let* $\mathbf{M} = \max_{1 \leq i \leq N} \mathbf{G}_i$. *Let* $\epsilon_1 = 0.01$ *and* $\epsilon_2 = 0.0006$. *Then for all* $\lambda > 0$, $\Pr[\mathbb{E}[\mathbf{M}] - \mathbf{M} \geq \lambda] \leq \exp\left(-q^{1/2+\epsilon_2-(\gamma+\epsilon_1)\ln q-\lambda}\right)$ *and* $\Pr[\mathbf{M} - \mathbb{E}[\mathbf{M}] \geq \lambda] \leq q^{\lambda-1/2-\epsilon_2-\gamma\ln q} + q^{2\lambda-1-2\epsilon_2-2\gamma\ln q}$.

PROOF. For each $t \in \mathbb{N}$, $\Pr[\mathbf{G}_i \geq t] = q^{t-1}$, so $\Pr[\mathbf{G}_i \leq t] = 1 - \Pr[\mathbf{G}_i \geq t+1] = 1 - q^t$.

Since the $\mathbf{G}_i$'s are independent, $\Pr[\mathbf{M} \leq t] = \prod_{i=1}^{N}(1-q^t) = (1-q^t)^N$.

Below we use theorem 5.7.4 and the inequalities $e^x\left(1 - \frac{x^2}{N}\right) \leq \left(1 + \frac{x}{N}\right)^N \leq e^x$ for $N > 1, |x| < N$.

Setting $t = \mathbb{E}[\mathbf{M}] - \lambda$, we have

$$\begin{aligned}
\Pr[\mathbf{M} \leq \mathbb{E}[\mathbf{M}] - \lambda] &= (1-q^t)^N \\
&= \left(1 - q^{(\mathbb{E}[\mathbf{M}]-\lambda)}\right)^N \\
&< \left(1 - q^{\log_{1/q} N-(\gamma+\epsilon_1)\ln q+1/2+\epsilon_2-\lambda}\right)^N \\
&= \left(1 - q^{\log_{1/q} N} q^{1/2+\epsilon_2-(\gamma+\epsilon_1)\ln q-\lambda}\right)^N \\
&= \left(1 - \frac{q^{1/2+\epsilon_2-(\gamma+\epsilon_1)\ln q-\lambda}}{N}\right)^N \\
&< \exp\left(-q^{1/2+\epsilon_2-(\gamma+\epsilon_1)\ln q-\lambda}\right)
\end{aligned}$$

The last inequality is true since $\left(1 + \frac{x}{N}\right)^N \leq e^x$.

Similarly, letting $t = \mathbb{E}[\mathbf{M}] + \lambda - 1$, we have

$$\begin{aligned}
& \Pr[\mathbf{M} \geq \mathbb{E}[\mathbf{M}] + \lambda] \\
= \; & 1 - \Pr[\mathbf{M} \leq \mathbb{E}[\mathbf{M}] + \lambda - 1] \\
= \; & 1 - \left(1 - q^t\right)^N \\
= \; & 1 - \left(1 - q^{\mathbb{E}[\mathbf{M}]+\lambda-1}\right)^N \\
< \; & 1 - \left(1 - q^{\log_{1/q} N + 1/2 - \epsilon_2 - \gamma \ln q + \lambda - 1}\right)^N \\
= \; & 1 - \left(1 - q^{\log_{1/q} N} q^{\lambda - 1/2 - \epsilon_2 - \gamma \ln q}\right)^N \\
= \; & 1 - \left(1 - \frac{q^{\lambda - 1/2 - \epsilon_2 - \gamma \ln q}}{N}\right)^N \\
< \; & 1 - \exp\left(-q^{\lambda - 1/2 - \epsilon_2 - \gamma \ln q}\right)\left(1 - \frac{q^{2(\lambda - 1/2 - \epsilon_2 - \gamma \ln q)}}{N}\right) \quad \text{since } e^x\left(1 - \frac{x^2}{N}\right) \leq \left(1 + \frac{x}{N}\right)^N \\
= \; & 1 - \exp\left(-q^{\lambda - 1/2 - \epsilon_2 - \gamma \ln q} + \ln\left(1 - \frac{q^{2\lambda - 1 - 2\epsilon_2 - 2\gamma \ln q}}{N}\right)\right) \\
\leq \; & q^{\lambda - 1/2 - \epsilon_2 - \gamma \ln q} - \ln\left(1 - \frac{q^{2\lambda - 1 - 2\epsilon_2 - 2\gamma \ln q}}{N}\right) \quad \text{since } 1 - e^x \leq -x \\
\leq \; & q^{\lambda - 1/2 - \epsilon_2 - \gamma \ln q} + \frac{2q^{2\lambda - 1 - 2\epsilon_2 - 2\gamma \ln q}}{N} \quad \text{since } \ln(1 - x) \geq -2x \text{ if } x < 0.7 \\
\leq \; & q^{\lambda - 1/2 - \epsilon_2 - \gamma \ln q} + q^{2\lambda - 1 - 2\epsilon_2 - 2\gamma \ln q} \quad \text{since } N \geq 2.
\end{aligned}$$

□

The following corollary for the special case of $p = \frac{1}{2}$ is used for our main result, showing that a maximum of $\frac{1}{2}$-geometric random variables is α-β-sub-exponential for $\alpha = 3.31, \beta = 2$.

COROLLARY 5.7.6. *Let $\mathbf{G}_1, \ldots, \mathbf{G}_N$ be i.i.d. $\frac{1}{2}$-geometric random variables, $N \geq 50$, and let $\mathbf{M} = \max_{1 \leq i \leq N} \mathbf{G}_i$. Then for all $\lambda > 0$, $\Pr[|\mathbf{M} - \mathbb{E}[\mathbf{M}]| \geq \lambda] < 3.31e^{-\lambda/2}$.*

PROOF. By theorem 5.7.5 and the union bound,

$$\begin{aligned}
\Pr[|\mathbf{M} - \mathbb{E}[\mathbf{M}]| \geq \lambda] \; & < \; \exp\left(-q^{1/2 + \epsilon_2 - (\gamma + \epsilon_1)\ln q - \lambda}\right) + q^{\lambda - 1/2 - \epsilon_2 - \gamma \ln q} + q^{2\lambda - 1 - 2\epsilon_2 - 2\gamma \ln q} \\
& = \; \exp\left(-2^{\lambda - (\gamma + \epsilon_1)\ln 2 - 1/2 - \epsilon_2}\right) + 2^{1/2 + \epsilon_2 - \lambda - \gamma \ln 2} + 2^{1 + 2\epsilon_2 - 2\lambda - 2\gamma \ln 2} \\
& < \; 3.31e^{-\lambda/2}. \quad \text{justified below}
\end{aligned}$$

To see the final inequality, note that

$$\begin{aligned}\exp\left(-2^{\lambda-(\gamma+\epsilon_1)\ln 2-1/2-\epsilon_2}\right) &< \exp\left(-2^{\lambda-1}\right)\\ &= \exp\left(-2^{\lambda}/2\right)\\ &\leq \exp\left(-\lambda/2\right)\end{aligned}$$

$$\begin{aligned}2^{1/2+\epsilon_2-\lambda-\gamma\ln 2} &= 2^{1/2+\epsilon_2-\gamma\ln 2}\cdot 2^{-\lambda}\\ &= 2^{1/2+\epsilon_2-\gamma\ln 2}\cdot 4^{-\lambda/2}\\ &< 2^{1/2+\epsilon_2-\gamma\ln 2}\cdot e^{-\lambda/2}\\ &< 1.1\cdot e^{-\lambda/2}.\end{aligned}$$

$$\begin{aligned}2^{1+2\epsilon_2-2\lambda-2\gamma\ln 2} &= 2^{1+2\epsilon_2-2\gamma\ln 2}\cdot 2^{-2\lambda}\\ &= 2^{1+2\epsilon_2-2\gamma\ln 2}\cdot 16^{-\lambda/2}\\ &< (1.1)^2\cdot e^{-\lambda/2}.\end{aligned}$$

So, their sum is less than $3.31e^{-\lambda/2}$. □

The following lemma bounds the maximum of N $\frac{1}{2}$-geometric random variables for the special cases of one lower and one upper threshold, which is stronger than the bounds given by theorem 5.7.6.

LEMMA 5.7.7. *Let* $\mathbf{G}_1,\ldots,\mathbf{G}_N$ *be i.i.d.* $\frac{1}{2}$*-geometric random variables,* $N \geq 50$*, and* $\mathbf{M} = \max_{1\leq i\leq N}\mathbf{G}_i$*. Then* $\Pr[\mathbf{M} \geq 2\log N] < N^{-1}$ *and* $\Pr[\mathbf{M} \leq \log N - \log\ln N] < N^{-1}$.

PROOF. For any i, $\Pr[\mathbf{G}_i \geq \log N - \log\ln N] = \left(\frac{1}{2}\right)^{(\log N-\log\ln N)} = \frac{\ln N}{N}$. Since the $\mathbf{G}_i$'s are independent, $\Pr[\mathbf{M} < \log N - \log\ln N] = \Pr[(\forall i)\ \mathbf{G}_i < \log N - \log\ln N] = \left(1-\frac{\ln N}{N}\right)^N \leq e^{-\ln N} = N^{-1}$. For the upper bound, for any i, $\Pr[\mathbf{G}_i \geq 2\log N] = \left(\frac{1}{2}\right)^{2\log N} = N^{-2}$. By the union bound, $\Pr[\mathbf{M} \geq 2\log N] = \Pr[(\exists i)\ \mathbf{G}_i \geq 2\log N] \leq N^{-1}$. □

LEMMA 5.7.8. *Let $N, K \in \mathbb{N}^+$, $N \geq 50$. Let $\mathbf{M}_1, \ldots, \mathbf{M}_K$ be i.i.d. random variables, each of which is the maximum of N i.i.d. $\frac{1}{2}$-geometric random variables. Define $\mathbf{S} = \sum_{i=1}^{K} \mathbf{M}_i$. Then for all $t \geq 0$, $\Pr\left[|\mathbf{S} - \mathbb{E}\left[\mathbf{S}\right]| \geq t\right] \leq 2 \cdot e^{K - t/4}$.*

PROOF. By theorem 5.7.6 and theorem 5.7.3, for $\alpha = 3.31 < 2e - 2$ and $\beta = 2$, we have

$$\Pr\left[|\mathbf{S} - \mathbb{E}\left[\mathbf{S}\right]| \geq t\right] \quad < \quad 2\frac{(1+\alpha/2)^K}{e^{t/(2\beta)}} < 2\frac{\left(1+\frac{2e-2}{2}\right)^K}{e^{t/4}} = 2 \cdot (e)^K \cdot e^{-t/4} = 2 \cdot e^{K - t/4}.$$

□

COROLLARY 5.7.9. *Let $a > 4$, $N \in \mathbb{N}^+$, $N > 50$, $K \geq \frac{\ln N}{\frac{a}{4} - 1}$, and $\delta_0 = 1/2 + \gamma/\ln 2 - \epsilon_2$. Let $\mathbf{M}_1, \ldots, \mathbf{M}_K$ be i.i.d. random variables, each of which is the maximum of N i.i.d. $\frac{1}{2}$-geometric random variables. Define $\mathbf{S} = \sum_{i=1}^{K} \mathbf{M}_i$. Then*

$$\Pr\left[\left|\frac{\mathbf{S}}{K} - \log N - \delta_0\right| \geq a\right] \leq \frac{2}{N}.$$

PROOF. We first manipulate the expression in the conclusion of the corollary to put it in a form where we can apply theorem 5.7.8.

$$\begin{aligned} & \Pr\left[\frac{\mathbf{S}}{K} - \log N - \delta_0 \geq a\right] \\ = \quad & \Pr\left[\mathbf{S} - K\left(\log N + 1/2 + \gamma/\ln 2 - \epsilon_2\right) \geq aK\right] \\ < \quad & \Pr\left[\mathbf{S} - \mathbb{E}\left[\mathbf{S}\right] \geq aK\right]. \end{aligned}$$

Since $K\left(\log N + 1/2 + \gamma/\ln 2 - \epsilon_2\right) \leq E[\mathbf{S}]$.

$$\begin{aligned} & \Pr\left[\log N + \delta_0 - \frac{\mathbf{S}}{K} \geq a\right] \\ = \quad & \Pr\left[K\left(\log N + 1/2 + \gamma/\ln 2 - \epsilon_2\right) - \mathbf{S} \geq aK\right] \\ = \quad & \Pr\left[K\left(\log N + 1/2 + \gamma/\ln 2 + \epsilon_1/\ln 2 + \epsilon_2\right) - \mathbf{S} \geq aK + \left(\epsilon_1/\ln 2 + 2\epsilon_2\right)K\right] \\ < \quad & \Pr\left[\mathbb{E}\left[\mathbf{S}\right] - \mathbf{S} \geq (a + \epsilon_1/\ln 2 + 2\epsilon_2)K\right] \\ < \quad & \Pr\left[\mathbb{E}\left[\mathbf{S}\right] - \mathbf{S} \geq aK\right]. \end{aligned}$$

Because $E[\mathbf{S}] < K\left(\log N + 1/2 + \frac{\gamma+\epsilon_1}{\ln 2} + \epsilon_2\right)$.

Since the events $\frac{\mathbf{S}}{K} - \log N - \delta_0 \geq a$ and $\log N + \delta_0 - \frac{\mathbf{S}}{K} \geq a$ are disjoint, and the events $\mathbf{S} - \mathbb{E}[\mathbf{S}] \geq aK$ and $\mathbb{E}[\mathbf{S}] - \mathbf{S} \geq aK$ are disjoint, the union bound holds with equality, so

$$\begin{aligned}
\Pr\left[\left|\frac{\mathbf{S}}{K} - \log N - \delta_0\right| \geq a\right] &= \Pr\left[\frac{\mathbf{S}}{K} - \log N - \delta_0 \geq a\right] + \Pr\left[\log N + \delta_0 - \frac{\mathbf{S}}{K} \geq a\right] \\
&< \Pr[\mathbf{S} - \mathbb{E}[\mathbf{S}] \geq aK] + \Pr[\mathbb{E}[\mathbf{S}] - \mathbf{S} \geq aK] \\
&= \Pr[|\mathbf{S} - \mathbb{E}[\mathbf{S}]| \geq aK].
\end{aligned}$$

Let $t = aK$. Applying theorem 5.7.8 with these values of K and t,

$$\begin{aligned}
\Pr\left[\left|\frac{\mathbf{S}}{K} - \log N - \delta_0\right| \geq a\right] &< \Pr[|\mathbf{S} - \mathbb{E}[\mathbf{S}]| \geq aK] \\
&= \Pr[|\mathbf{S} - \mathbb{E}[\mathbf{S}]| \geq t] \\
&\leq 2 \cdot e^{K - t/4} \\
&= 2 \cdot e^{K(1-\frac{a}{4})} \\
&= 2 \cdot e^{-K(\frac{a}{4}-1)} \\
&\leq 2 \cdot e^{-\frac{\ln N}{(\frac{a}{4}-1)}(\frac{a}{4}-1)} \\
&= 2 \cdot e^{-\ln N} \\
&= \frac{2}{N}.
\end{aligned}$$

□

For example, choosing $a = \ln 2 + 4 < 4.7$ means we can choose $K \geq \frac{\ln N}{\frac{a}{4}-1} = \frac{\ln N}{(\ln(2)+4)/4-1} = \frac{\ln N}{\ln(2)/4} = 4\log_2 N$:

Corollary 5.7.10. *Let $N \in \mathbb{N}^+$, $N \geq 50$, $K \geq 4\log N$. Let $\mathbf{M}_1, \ldots, \mathbf{M}_K$ be i.i.d. random variables, each of which is the maximum of N i.i.d. $\frac{1}{2}$-geometric random variables. Define $\mathbf{S} = \sum_{i=1}^{K} \mathbf{M}_i$. Then*

$$\Pr\left[\left|\frac{\mathbf{S}}{K} - \log N\right| \geq 4.7\right] \leq \frac{2}{N}.$$

pear

5.8. Conclusion and open problems

In this chapter, we gave a brief description of our protocol that approximately compute the population size n. We also discussed a technique for converting nonuniform protocols (those that assume agents are initialized with an approximate estimate of n) to uniform protocols, by composing size approximation with the nonuniform protocol.

While our focus in this chapter is the size counting problem, the mentioned protocols demonstrate general techniques that can help solve other problems and design new protocols. Alistarh and Gelashvilli [6] mentioned different ideas such as the space multiplexing used in [25, 28, 47, 50] and the junta-driven phase clock [60] as available building blocks to design new protocols. We also summarized two variations of the discrete averaging technique introduced in [7], tightly analyzed in [21, 28, 29, 79, 81], that has since been widely deployed in other protocols to solve the counting problem [28, 51, 78, 79] and the exact majority problem [7, 50, 78]. (See Protocol 0 and Protocol 4.)

Open questions with composition of two uniform protocols. We discussed how to make a nonuniform protocol uniform through composition with a uniform counting protocol that allows the nonuniform protocol to use the output of the counting protocol. Generally, in a composition of a uniform protocol **U** with a protocol **D**, protocol **D** might get restarted repeatedly. In each restart, the agents propagate a new signal with updated information about the size. The counting protocol might even generate a new restart signal before the previous signal hits all the agents. Having this in mind, if a protocol uses duplicate restart signals, restarted and deprecated agents could become indistinguishable. For example, using restart signals of constant size might create inconsistency in the population.

Unique (and perhaps monotonically increasing) restarting signals guarantees the correctness of the downstream protocol. Since eventually, all agents agree on the last (largest) restart signal and restart protocol **D** for the final execution. Even assuming a monotone increasing restart signal might change some probability bounds on the convergence time of protocols. The current literature lacks a general-purpose theorem that proves under what conditions of a downstream protocol the restarting technique works.

Collective output representation of the population size. All the counting protocols summarized in this chapter require all agents eventually to represent the computed count. If agents are required to store the value n, then there is clearly a linear-state lower bound, since $\log n$ bits are required

merely to write n. However, what if no agent individually stores all of n? Consider instead a *collective* representation of the population size, where some agents each store (for example) one bit of n, as well as the significance of the bit.

With this trick, the lower bound does not apply anymore. There might exist a protocol that solves the exact counting with $o(n)$ states. However, readout could be more difficult, since we cannot simply look at the memory of a single agent and read the population size; instead, we must sample a small subset of the population. For example, composing size computation with another protocol would be less straightforward since the agents who are computing the downstream protocol would not at any point have access to all the bits of n. One could imagine a protocol that spreads the output to the whole population almost equally: for example, having $O(n/\log n)$ agents responsible for each index of the binary expansion of n. With this trick, the output will be present dense enough among the population. Thus, a random sample of polylogarithmic agents would have enough information to reconstruct the value of n.

The $O(n \log n)$ time slow size approximation protocol (Protocol 2) collectively represents n: each remaining active agent holds a value k such that there is a 1 at significance k in the binary expansion of n. Is there a sublinear-time, sublinear-state protocol, so that the agents report the population size via this collective representation? A valid solution to the exact counting problem with a collective output also solves the *parity* problem: compute the least significant bit of n.

CHAPTER 6

Dynamic Population Protocols and Approximate Counting

6.1. Introduction

Biological systems are incredibly dynamic environments, so we must extend our understanding of population protocols to a model that considers such continual changes. A model capturing such phenomena would allow faulty agents that suddenly stop working, leave the network, or lose some of their stored information. In the conventional model of population protocols, we assume a constant population size with non-faulty agents. However, it is natural to consider biological applications that should tolerate the dynamic changes within the environment. Therefore, we studied population protocols robust to adversarial situations. Concretely, we develop a counting protocol that is robust to an adversary who can continuously:

(1) add agents in their initial state.

(2) observe the agents' memory and remove them arbitrarily.

The dynamic size counting problem. In contrast to most work, which assumes the population size n is fixed over time, we model an adversary that can add or remove agents arbitrarily and repeatedly during the computation. All agents start in the same state, including newly added agents. The goal is for each agent to approximately count the population size n, which we define to mean that all agents should eventually store the same output k in their states, which with high probability is within a constant multiplicative factor of $\log n$.[1] Once all agents have the same output k, they have *converged*. They maintain k as the output for some time called the *holding time* (after which they might alter k even if the population size has not changed). In response to a "significant" change in size from n_{prev} to n_{next}, agents should re-converge to a new output k' of $\log n_{\text{next}}$. (Agents are not "notified" about the change; instead they must continually monitor the population to test whether their current output is accurate.) Note that if n_{prev} is close to n_{next}

[1] *Nonuniform* protocols require agents to be initialized with an estimate k of $\log n$ in order to accomplish other tasks, such as a "leaderless phase clock" [2]. The bound $k = \Theta(\log n)$ is necessary and sufficient for correctness and speed in most cases [2, 3, 5, 20, 24, 26, 27, 30, 61, 79, 91].

(within a polynomial factor), then k may remain an accurate estimate of n_{next}, so agents may not re-converge in response to a small change.

Ideally the expected convergence time is small, and the expected holding time is large. With a fixed size population, it is common to require the output to *stabilize* to a value that never again changes after convergence, i.e., infinite holding time. However, this turns out to be impossible with an adversary that can remove agents (theorem 6.2.4). When changing from size n_{prev} to n_{next}, our protocol achieves expected convergence time $O(\log n_{\text{next}} + \log\log n_{\text{prev}})$ and expected holding time $\Omega(n_{\text{next}}^c)$, where c can be made arbitrarily large. The number of bits of memory used per agent is $O(\log^2(s) + (\log\log n)^2)$, where s is the maximum integer stored in the agents' memory after the change.

While it is common to measure population protocol memory complexity by counting the number of states (which is exponentially larger than the number of bits required to represent each state), that measure is a bit awkward here. Our protocol is uniform—the same transition rules for every population size—so has an infinite number of producible states. One could count expected number of states that will be produced, but this is a bit misleading: in time t each agent visits $O(t)$ states on average, so $O(t \cdot n)$ states total. Counting how many bits are required is more accurate metric of the actual memory requirements.

The loosely-stabilizing counting problem. The dynamic size counting problem has an equivalent characterization: rather than removing agents and adding them with a fixed initial state, the *loosely-stabilizing* adversary sets each agent to an arbitrary initial state in a fixed-size population. A protocol solves the dynamic size counting problem if and only if it solves the loosely-stabilizing counting problem, with the same convergence and holding times (theorem 6.2.5). Due to this equivalence, we analyze our protocol assuming a fixed population size and adversarial initial states. In this case our convergence time $O(\log n + \log M)$ is measured as a function of the population size n and the value M that is the maximum `estimate` value stored in agents' memory. From the perspective of the dynamic size counting problem, these "adversarial initial states" would correspond to the agent states after correctly estimating the *previous* population size, just prior to adding or removing agents.

6.2. Dynamic size counting protocol

In a population of size n, define $C(n, \epsilon_1, \epsilon_2)$ to be the set of correct configurations $\mathbf{c}$ such that every agent u in $\mathbf{c}$ obeys $\epsilon_1 \log n < u.\texttt{estimate} < \epsilon_2 \log n$. Let t_h be any time bound. Moreover, we define $L(n, t_h) \subset C(n, \epsilon_1, \epsilon_2)$ the subset of correct configurations such that as the expected time for protocol P starting from a configuration $\mathbf{l} \in L(n, t_h)$ to stay in $C(n, \epsilon_1, \epsilon_2)$ is at least $t_h(n)$.

Definition 6.2.1. *Let n_{prev} and n_{next} denote the previous and next population size. A protocol P solves the* dynamic size counting *problem if there are $\epsilon_1, \epsilon_2 > 0$, called the* accuracy*, such that if the population size changes from n_{prev} to n_{next}, the protocol reaches a configuration $\mathbf{l}$ in $L(n_{next}, t_h)$ with high probability. The time needed to do this is called the* convergence *time. Moreover, t_h, the time that the population stays in $C(n_{\text{next}}, \epsilon_1, \epsilon_2)$, is called the* holding *time.*

A population protocol is $(t_c(n), t_h(n))$-*loosely stabilizing* if starting from any initial configuration, the agents reach a correct configuration in $t_c(n)$ time and stay in the correct configuration for additional $t_h(n)$ time [89, 90]. In contrast to self-stabilizing [14, 34], subsequent interactions can move the agents to an incorrect configuration; however, the agents recover quickly from an incorrect configuration.

Given any starting configuration $\mathbf{s} \notin C(n, \epsilon_1, \epsilon_2)$ of size n, we define $f_c(\mathbf{s}, L(n, t_h))$ as the expected time to reach a correct configuration in $L(n, t_h)$.

Definition 6.2.2. *[90, Definition 2] Let $t_c(n, M)$ and $t_h(n)$ be functions of n, the largest integer value M in the initial configuration $\mathbf{s}$, and the set of correct configuration $C(n, \epsilon_1, \epsilon_2)$. A protocol P is a $t_c(n, M), t_h(n), \epsilon_1, \epsilon_2$ loosely-stabilizing population size counting protocol if there exists a set $L(n, t_h) \subset C(n, \epsilon_1, \epsilon_2)$ of configurations satisfying:*

For every n and every initial configuration $\mathbf{s} \notin C(n, \epsilon_1, \epsilon_2)$ of size n, $f_c(\mathbf{s}, L(n, t_h)) \leq t_c(n, M)$.

6.2.1. Basic properties of the dynamic size counting problem. We first observe that the key challenge in dynamic size counting is that the adversary may remove agents. If the adversary can only add agents, the problem is straightforward to solve with optimal convergence and holding times.

Observation 6.2.3. *Suppose the adversary in the dynamic size counting problem only adds agents. Then there is a protocol solving dynamic size counting with $O(\log n)$ convergence time (in expectation and with probability $\geq 1 - O(1/n)$) and infinite holding time.*

PROOF. Each agent in the initial state s generates a geometric random variable. After the last time that the adversary adds agents, resulting in n total agents, exactly n geometric random variables will have been generated. Agents propagate the maximum by epidemic using transition $a, b \to \max(a, b), \max(a, b)$, taking $3 \ln n$ time to reach all agents with probability

$\geq 1 - \frac{1}{n^2}$ [34, Corollary 2.8]. The maximum of n i.i.d. geometric random variables is in the range $[\log n - \log \ln n, 2 \log n]$ with probability $\geq 1 - \frac{1}{n}$ [47, Lemma D.7]. □

In contrast, if the adversary can *remove* agents, then even if it is guaranteed to do this exactly once, no protocol can be stabilizing, i.e., have infinite holding time.

OBSERVATION 6.2.4. *Suppose the adversary in the dynamic size counting problem will remove agents exactly once. Then any protocol solving the problem has finite holding time.*

PROOF. Suppose otherwise. Let the initial population size be n and the later size be $n' < n$. The protocol must handle the case where the adversary *never* removes agents, since in population size n this is equivalent to an adversary who starts with $n+1$ agents and immediately removes one of them. Thus if the adversary waits sufficiently long before the removal, then all agents stabilize to output $k = \Theta(\log n)$. In other words, no sequence of transitions can alter the value, including transitions occurring only among any subpopulation of size n'. So after the adversary removes $n - n'$ agents, the remaining n' agents are unable to alter the output k, a contradiction if n' is sufficiently small compared to n such that the output k is not a correct estimate for a population of size n'. □

Recall that we define M as the largest integer value the agents stored in the starting configuration $\mathbf{s}$. theorem 6.2.5 shows that the dynamic size counting problem is equivalent to the loosely-stabilizing counting problem. Due to this equivalence, our correctness proofs will use the loosely-stabilizing characterization. The proof is given in the full version [49].

LEMMA 6.2.5. *A protocol solves the dynamic size counting problem with convergence time $t_c(n, M)$ and holding time $t_h(n)$ if and only if it solves the loosely-stabilizing counting problem with convergence time $t_c(n, M)$ and holding time $t_h(n)$.*

PROOF SKETCH. Any states present in an adversarially prepared configuration $\mathbf{c}$ will be produced in large quantities from any sufficiently large initial configuration of all initialized states

s [47, Lemma 4.2]. The dynamic size adversary can then remove agents to result in $\mathbf{c}$, which the protocol must handle, showing it can handle an arbitrary initial configuration. □

6.2.2. High-level overview of dynamic size counting protocol. This section briefly describes our protocol for solving the dynamic size counting, defined formally in section 6.2.3. By theorem 6.2.5, it suffices to design a protocol solving the loosely-stabilizing counting problem for a fixed population size n. Our protocol uses the "detection" protocol of [4]. Consider a subset of states designated as a "source". A detection protocol alerts all agents whether a source state is present in the population.

In Protocol 32, the population maintains several dynamic *groups*, with the agent's group stored as a positive integer field `group`. The `group` values are not fixed: each agent changes its `group` field on every interaction, with equal probability either incrementing `group` or setting it to 1. We show that, no matter the initial group values, after $O(\log n)$ time the group values will be in the range $[1, 8 \log n]$ WHP. Furthermore, the distribution of `group` values is very close to that of n i.i.d. geometric random variables, in the sense that each agent's `group` value is independent of every other, with expected $n/2^i$ agents having $\texttt{group} = i$ if each agent has had at least i interactions.[2]

The agents store an array of "signal" integers in their `signals` field to track the existing `group` values in the population. Each agent in the i'th group is responsible for *boosting* the signal associated with i. The goal is to have $\texttt{signals}[i] > 0$ for all agents if and only if some agent has $\texttt{group} = i$.

The detection protocol of [4], explained below, provides a technique for agents to know which groups are still present. Once a signal for group k fades out, the agents speculate that there is no agent with $\texttt{group} = k$. Depending on the current value stored as `estimate` in agents' memory and the value k, this might cause re-calculating the population size. The agents are constantly checking for the changes in the `signals`. They re-compute `estimate` once there is a large gap between `estimate` and the first `group` i with $\texttt{signals}[i] = 0$. We call i the *first missing value* (stored in the field `FMV`). The `signals` array is updated as follows. An agent with $\texttt{group} = k$ sets $\texttt{signals}[k]$ to its maximum possible value $(3k+1)$; we call this *boosting*. Other groups k are updated between two agents u, v with $u.\texttt{signals}[k] = a$ and $v.\texttt{signals}[k] = b$ via *propagation* transitions that set both

[2]The difference is that a geometric random variable G obeys $\Pr[G = j] = 1/2^j$ for *all* $j \in \mathbb{N}^+$, but after i interactions an agent u can increment $u.\texttt{group}$ by at most i, so $\Pr[u.\texttt{group} = j] = 0$ if $j \gg i$.

agent's `signals`$[k]$ to $\max(a-1, b-1, 0)$. The paper [4] used a nonuniform protocol where each agent *already* has an estimate of $\log n$. They prove that if the state being detected (in our case, a state with `group` $= k$) is *absent* and the current maximum signal is c, then all agents will have signal 0 within $\Theta(c)$ time. However, if the state being detected is present, then the boosting transitions (occurring every $O(1)$ units of parallel time on average in the worst case that its count is only 1) will keep the signal positive in all agents with high probability. For this to hold, the maximum value set during boosting must be $\Omega(\log n)$; the nonuniform protocol of [4] uses its estimate of $\log n$ for this purpose.

Crucially, our protocol associates smaller maximum signal values to smaller group values (so many are much smaller than $\log n$), to ensure that a signal does not take abnormally long to get to 0 when its associated group value is missing. Otherwise, if we set each signal value to $\Omega(\log n)$ (based on the agent's current estimate of $\log n$) during boosting, then it would take time proportional to `estimate` (which could be much larger than the actual value of $\log n$) to detect the absence of a `group` value. Thus it is critical that we provide a novel analysis of the detection protocol, showing that the signals for smaller group values $k \ll \log n$ remain present with high probability. This requires arguing that the boosting reactions for such smaller values are happening with sufficiently higher frequency, due to the higher count of agents with `group` $= k$, compensating for the smaller boosting signal values they use.

6.2.3. Formal description of loosely-stabilizing counting protocol. Our dynamic counting protocol (Protocol 32) divides agents among several groups via the UpdateGroup subprotocol. The agents update their `group` from i to $i+1$ with probability $1/2$ or reset to group 1 with probability $1/2$. The number of agents at each group and the total number of groups are both random variables dynamically changing through time. We show that the total number of groups remains close to $\log n$ at all times with high probability.

The agents start with arbitrary (or even adversarial) `group` values but we show that WHP the set of `group` values will converge to $[1, 8\log n]$ within $O(\log n)$ time. Additionally, each agent stores an array of $O(\log n)$ signal values in their `signals` field. The goal is to maintain positive values in the `signals`$[i]$ if some agent has `group` $= i$. The agents store the index of the first `group` i with `signals`$[i] = 0$ in their `FMV` field. They use `FMV` as an approximation of $\log n$ and constantly compare it with their `estimate` value.

Depending on the `estimate` value stored in agents' memory, the agents maintain three main phases of computation:

NormalPhase:: An agent stays in the NormalPhase as long as there is a small gap between `estimate` and `FMV`: $0.25 \cdot \texttt{estimate} \leq \texttt{FMV} \leq 2.5\texttt{estimate}$.

WaitingPhase:: An agent switches from NormalPhase to WaitingPhase if it sees a large gap between the `FMV` and `estimate`: $\texttt{FMV} \notin \{0.25 \cdot \texttt{estimate}, \ldots, 2.5 \cdot \texttt{estimate}\}$. The purpose of WaitingPhase is to give enough time to the other agents so that by the end of the WaitingPhase for one agent, with high probability every other agent has also noticed the large gap between the `FMV` and `estimate` and entered WaitingPhase.

UpdatingPhase:: During the UpdatingPhase, every agent uses a new geometric random variable and propagates the maximum by epidemic. We set WaitingPhase long enough so that with high probability when the first agent switches to the UpdatingPhase, the rest of the population are all in WaitingPhase. By the end of UpdatingPhase, every agent switches back to NormalPhase.

Below we explain each subprotocol in more detail.

Protocol 32 DynamicCounting(u, v)

```
for agent ∈ {u, v} do
    UpdateGroup(agent)
SignalPropagation(u, v)
for agent ∈ {u, v} do
    UpdateMV(agent)
    SizeChecker(agent)
    if agent.phase ≠ NormalPhase then
        TimerRoutine(agent)
PropagateMaxEst(u, v)
for agent ∈ {u, v} do
    if agent.phase = NormalPhase then
        agent.estimate ← agent.GRV
```

In every interaction, both sender and receiver update their group according to the rules of the UpdateGroup subprotocol. If we look at the distribution of the `group` values after $O(\log n)$ time, there are about $n/2$ agents in group 1, $n/4$ agents in group 2, and $n/2^i$ agents in group i (see fig. 6.4). Note that the number of agents in each group decreases exponentially. Still, we ensure

that agents with larger `group` values use stronger signals to propagate, since there is less support for those groups.

Protocol 33 UpdateGroup(u)

$$\text{u.group} \leftarrow \begin{cases} \text{u.group} + 1 & \text{with probability } 1/2 \\ 1 & \text{with probability } 1/2 \end{cases}$$

To notify all agents about the set of all `group` values that are generated among the population, we use the detection protocol of [4] that is also used as a synchronization scheme in [22]. The agents store an integer for each `group` value that is generated by the population. The `signals` is an array of length $\Theta(\log n)$ such that a positive value in index i represents some agents in the population have generated $\texttt{group} = i$. Note that, as an agent updates its `group`, it boosts multiple signals based on its `group` value, e.g., an agent with $\texttt{group} = i$ helped boost all the indices $1, 2, 3, \ldots, i$ of `signals` in its last i interactions. We use the SignalPropagation protocol to keep the signal of group i positive as long as some agents have generated $\texttt{group} = i$.

Algorithm 34 SignalPropagation(u, v)

▷ Boosting:
$\text{u.signals}[\text{u.group}] \leftarrow (3 \cdot \text{u.group}) + 1$
$\text{v.signals}[\text{v.group}] \leftarrow (3 \cdot \text{v.group}) + 1$
▷ Propagate signal:
for $i \in \{1, 2, \ldots, \texttt{Max}(|\text{u.signals}|, |\text{v.signals}|\}$ **do**
  $m \leftarrow \texttt{Max}(\text{u.signals}[i], \text{v.signals}[i])$
  $\text{u.signals}[i], \text{v.signals}[i] \leftarrow \texttt{Max}(0, m - 1)$

Regardless of the initial configuration, the distribution of `group` values changes immediately (in $O(\log n)$ time), but it might take more time for the `signals` to get updated. It takes $O(i)$ time for $\texttt{signals}[i]$ to hit zero. The larger the index i, $\texttt{signals}[i]$ leaves the population slower. Hence, the agents look at the *first missing signal* that they observe among the array of all signals.

Algorithm 35 UpdateMV(u)

▷ Find the first appearance of a zero in $u.\texttt{signals}$ beyond index $\lceil \texttt{log}(\text{u.estimate}) \rceil$
$s \leftarrow \lceil \texttt{log}(\text{u.estimate}) \rceil$
$\text{u.FMV} \leftarrow \min\{i \in [s, |\text{u.signals}|] \mid \text{u.signals}[i] = 0\}$

Once there is a large gap between the first missing `group` (`FMV`) and the agents' estimation of $\log n$ (`estimate`), each agent individually moves to a *waiting phase* and waits for other agents to

catch the same gap between their `estimate` and `FMV`. Note that we time this phase as a function of `FMV` and not the `estimate` since the `estimate` is not valid anymore and might be much smaller or larger than the actual value of $\log n$.

Algorithm 36 SizeChecker(u)

```
if u.phase = NormalPhase and u.FMV ∉ {0.25 · u.estimate, ..., 2.5 · u.estimate} then
    u.phase ← WaitingPhase                ▷ Waiting for other agents to detect the size change
```

Eventually, all agents will notice the large discrepancy between `FMV` and `estimate` and move to the WaitingPhase. The WaitingPhase is followed by the UpdatingPhase (explained in TimerRoutine). In the UpdatingPhase, all agents generate one geometric random variable (stored in `GRV`) and propagate the maximum value. We assume the agents generate a geometric random variable in one line (line 4 in Protocol 37) for simplicity.[3]

Once the UpdatingPhase is completed, all agents will update their `estimate` to the maximum geometric random variable they have seen and switch to the NormalPhase again. Recall that the agents remain in the NormalPhase as long as their `FMV` and `estimate` are relatively close. They continue changing their `group` values and send group signals as described earlier.

Algorithm 37 TimerRoutine(u)

```
u.timer ← u.timer + 1
if u.timer > 12 · u.FMV then
    if u.phase = WaitingPhase then
        u.GRV ←a new geometric random variable
        u.timer ← 0, u.phase ← UpdatingPhase
    if u.phase = UpdatingPhase then
        u.estimate ← u.GRV
        u.timer ← 0, u.phase ← NormalPhase
```

Algorithm 38 PropagateMaxEst(u, v)

```
if u.phase = v.phase  &  u.phase ≠ WaitingPhase then
    u.GRV, v.GRV ← max(u.GRV, v.GRV)
```

Intuitively, for each `group` value, about $n/2^i$ agents will hold `group` $= i$, and boost `signals`$[i]$ by setting it to the max $= \Theta(i)$. As the value of i grows, the number of agents with `group` $= i$ decreases, but their signals get stronger since the agents enhance a `group` signal i proportional to i.

[3]Alternatively, the agents could generate a geometric random variable through $O(\log n)$ consecutive interactions, each selecting a random coin flip (**H** or **T**). In this alternative version, we should make the WaitingPhase longer.

In a normal run of the protocol, the agents expect to have positive values in $\texttt{signals}[i]$ for `group` values between $[\log \ln n, \log n]$.

6.3. Simulation results

In this section, we present our simulation results for Protocol 32. We present two separate simulation results, one starting with a uniformly random initial configuration (fig. 6.1) in which each agent starts with a random number (bounded by 60) in each of their `group` and `estimate` fields. Additionally, we set a random integer of $\Theta(i)$ in each index of their `signals`. In fig. 6.1, we depict the `group` and `signals` fields of the agents in a population size $n = 10^6$. We can observe the changes in the `group` and `signals` within multiple snapshots from the population.

In our second simulation, we initialized the population with default values (starting in the initialized setting), however, after the population converges to an `estimate` of $O(\log n)$ we remove agents uniformly at random to simulate an adversarial initialized population. The result of this simulation is shown in fig. 6.2.

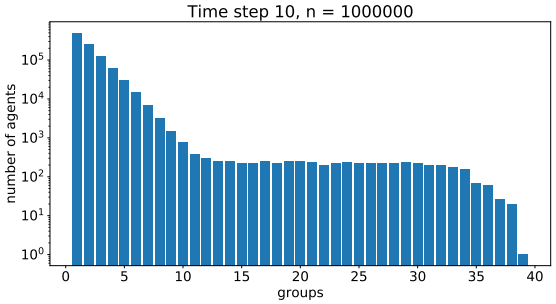
6.4. Analysis of dynamic counting protocol

6.4.1. Bound on the group values. Recall that the agents calculate a dynamic `group` value by following the rules of Protocol 33. As described in this protocol, the agents either move to the next `group` or return to `group` 1 with probability 1/2.[4]

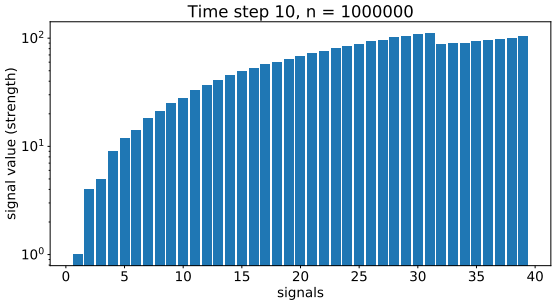
In this part, we analyze the distribution of `group` values. Note that the `group` values are rather chaotic at the beginning of the protocol since the agents might start holding any arbitrary `group` values that are much larger than $\log n$. However, after all agents reset back to $\texttt{group} = 1$, we can show for each $\texttt{group} = k$, $\Pr[\texttt{group} = k] \approx \frac{1}{2^k}$.

In the rest of this section, we assume the initialized setting for simplicity. Later, we show how we can generalize our results to any arbitrary initial configuration. We define $G_{u,t}$ as the `group` value of agent u at time t and $I(t, u)$ to represent the number of interactions involving this agent by the time t. Note that with this definition, $G_{u,t}$ is equal to k (for $k < I(t, u)$) if and only if agent u generates the sequence of $[HTTT \dots T]$ (**H** followed by $k - 1$ **T**s) during its last k interactions.

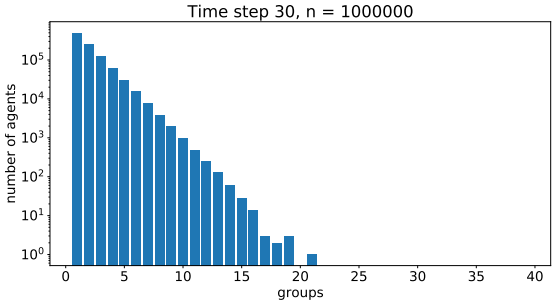
[4]The truncated version of this Markov chain (mapping all states $k + 1$, $k + 2$, ... to $k + 1$) is also known as the "winning streak" [72].



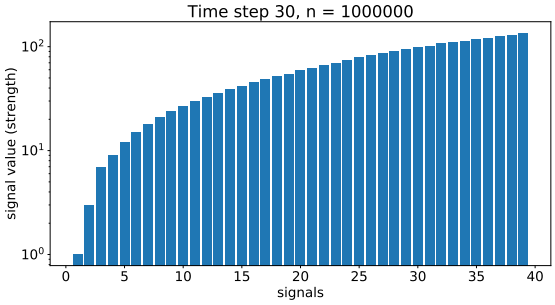
(a) The distribution of `group` values at $t = 10$. Observe that for all `group` values less than 10, the distribution is close to the stationary distribution in which about $\frac{n}{2^i}$ agents have `group` $= i$. However, it is too soon for all "unwanted" group values to disappear.



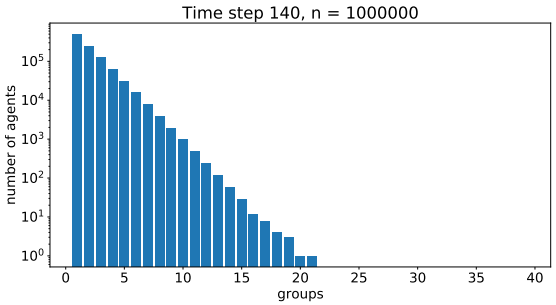
(b) Snapshot of the population at time 10: the distribution of `signals` values. The data shows $min(u.\texttt{signals}[i])$ for every index i and agent $u \in \mathcal{A}$. Note that, for every index i, `signals`$[i]$ now has a positive value.



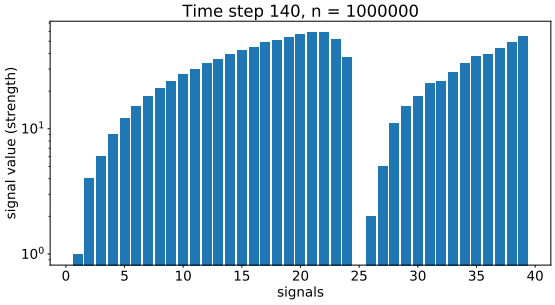
(c) Snapshot of the population at time 30: the distribution of `group` values reached the stationary distribution.



(d) Snapshot of the population at time 30: even though there is no agent with `group` > 25, the `signals` for all `group`s are still positive.

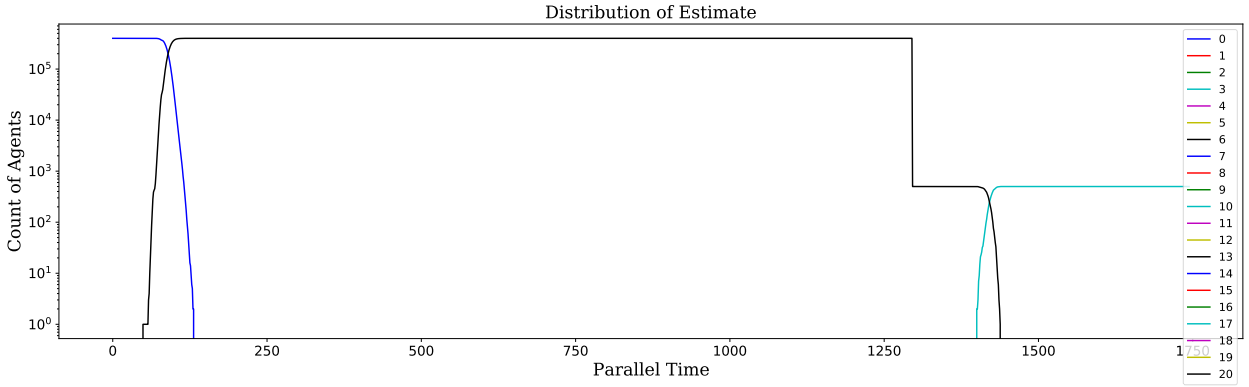


(e) Snapshot of the population at time 140: the distribution of `group` values reached the stationary distribution and stays the same for polynomial time.

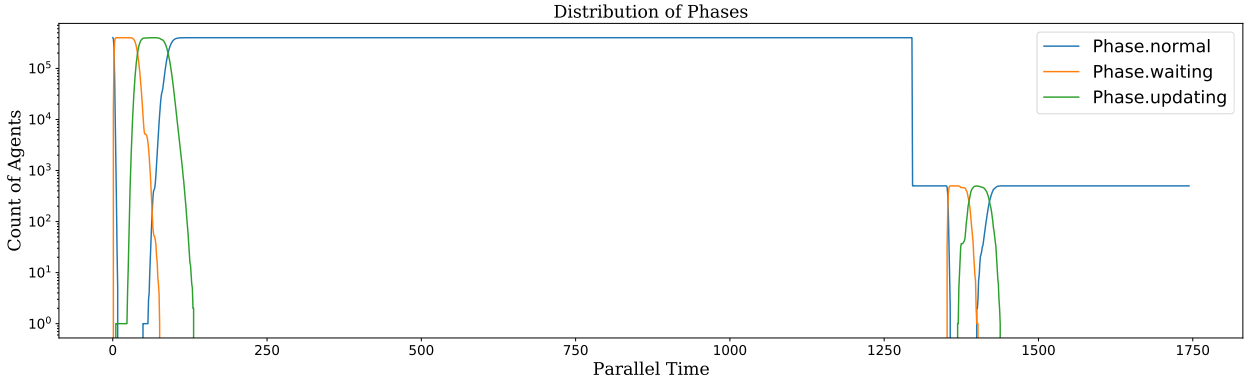


(f) Snapshot of the population at time 140: the emergence of the `FMV` $= 25$ which is in $O(\log n)$ for the entire population. Note that in our protocol the agents don't wait for all values of 25 and greater to disappear. Having `signals`$[i] = 0$ for one index i suffices to re-calculate the `estimate`.

FIGURE 6.1. Simulation results for population size $=10^6$. Initializing each agent with a random $1 \leq$ `group` < 30, and for every $1 \leq i \leq 60$, $0 \leq$ `signals`$[i] \leq 3 \cdot i + 1$ in NormalPhase, WaitingPhase, or UpdatingPhase with probability $1/3$. Plots of the `signals` for a randomly initialized population. The x-axis shows all the indices in the `signals` of the agents (bounded by 40 in the simulation). On the y-axis, and every index i, we take the minimum pairwise value of $u.\texttt{signals}[i]$ for all $u \in \mathcal{A}$.



(a) The distribution of `estimate` during the course of a computation. First, the agents agree on `estimate` $= 20$ and keep their estimate throughout the computation. This simulation shows how the agents update their `estimate` once we removed $O(n)$ agents from the original population.



(b) The distribution of `phase` values. This simulation shows how the agents update their `estimate` by going through WaitingPhase and UpdatingPhase consecutively.

FIGURE 6.2. Simulation results for population size $n = 400000 \approx 2^{18}$. Initializing each agent with `group` $= 1$, and an empty `signals` in NormalPhase. First, the agents calculate the `estimate` for $n = 2^{18}$, then at time 1350 we remove all but 500 agents randomly which results in updating the agents' `estimate` from 20 to 10.

Thus, we have:

$$\forall k \in \mathbb{N}, \quad 1 \leq k < I(t,u) : \Pr\left[G_{u,t} = k\right] = \frac{1}{2^k} \tag{6.1}$$

With this definition $G_{u,t}$ is undefined for any agents that has not generated H yet. In other words, the values $G_{u,t}$ are "close to geometric" in the sense that they are independent and have probability equal to a geometric random variable on all values $k < I(t,u)$.

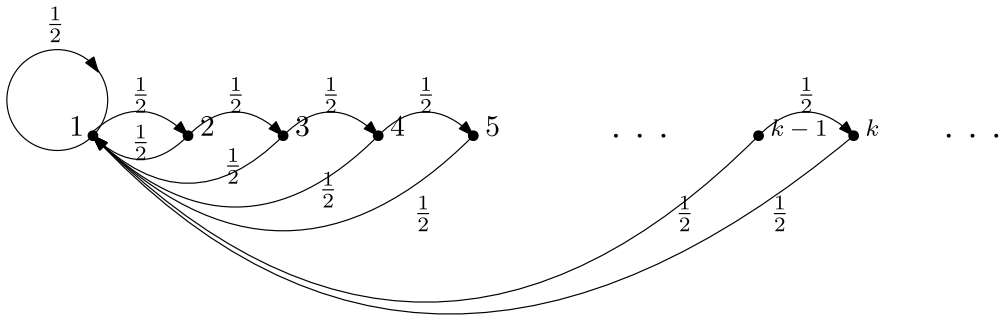


FIGURE 6.3. The infinite chain of `group` values.

OBSERVATION 6.4.1. *For agents $u_1, u_2, \ldots, u_n$, and the values $k_i < I(t, u_i)$, for $1 \leq i \leq n$:*

$$\Pr\left[G_{u_1,t} = k_1, G_{u_2,t} = k_2, \ldots, G_{u_n,t} = k_n\right] = \prod_{i=1}^{n} \Pr\left[G_{u_i,t} = k_i\right]$$

Next we bound the maximum `group` value that has been generated by any agent. Let $M_t = \max_{u \in \mathcal{A}} G_{u,t}$ be the maximum value of $G_{u,t}$ across the population at time t. A proof of the following lemma appears in the full version [49].

LEMMA 6.4.2. *Let $c \geq 2$ and let t be a time such that all agents have at least $c \log n$ interactions. In a population of size n, $\frac{1}{d} \log n \leq M_t$ with probability at least $1 - \exp\left(-n^{1-1/d}\right)$ and $M_t < c \log n$ with probability at least $1 - n^{1-c}$.*

Note that the maximum `group` value has a large variance. However, we can prove a tight bound for the first `group` value with no support; since to have $\texttt{FMV} = k$, for all values i that are less than k, $\exists u \in \mathcal{A}$ such that $u.\texttt{group} = i$.

So, we analyze the bounds for the first `group` value with no support, i.e., the value $\min\{k \in \mathbb{N}^+ \mid (\forall u \in \mathcal{A})\ u.\texttt{group} \neq k\}$. Considering n i.i.d. geometric random variables, the *first missing value* to be the smallest integer not appearing among the random variables. The first missing value has been studied in the literature [70, 71, 84] as the "the first empty urn" (see also "probabilistic counting" [58]) but for simplicity we use a loose bound for our analysis. The proof appears in the full version [49].

LEMMA 6.4.3. *Let $\delta > 0$, $0 < \epsilon < 1$ and let t be a time such that all agents have at least $(1 + \delta) \log n$ interactions. Define $\texttt{FMV}_t = \min\{k \in \mathbb{N} \mid (\forall u \in \mathcal{A})\ u.\texttt{group} \neq k\}$ at time t. Then,*

$\texttt{FMV}_t > (1-\epsilon)\log n$ *with probability at least* $(1-\epsilon)\log(n)\cdot\exp\left(-n^{\epsilon}\right)$ *and* $\texttt{FMV}_t \leq (1+\delta)\log n$ *with probability at least* $1-\left(\frac{1}{n^{\delta/2}}\right)^{(2+\delta)\log n}$.

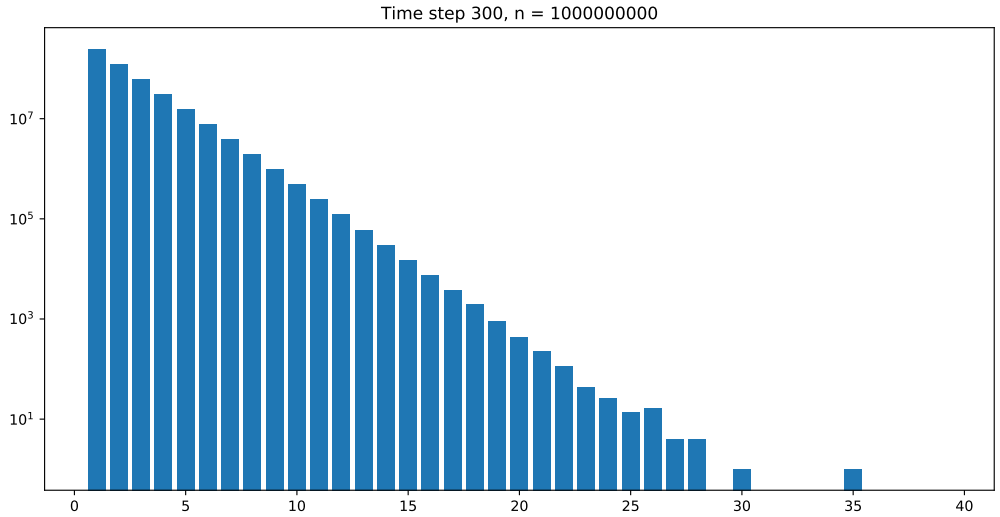


FIGURE 6.4. Showing the distribution of `group` values after 300 parallel-time in a population of size $n = 10^9$. The x-axis indicates the different `group` values while the y-axis indicates the number of agents in each group. Note that, we are using log-scale for the y-axis. In this snapshot of the population, $\texttt{FMV} = 29$. Even though, the maximum `group` value is 35 and is much larger than `FMV`.

6.4.2. Distribution of the groups. So far, we have proved bounds on the existing `group` values. However, in general, we need to show that at a given time $t = \Omega(\log n)$, there are about $\frac{n}{2^k}$ agents having $\texttt{group} = k$ WHP. The following lemma gives us a lower and upper bound for the number of agents in each `group`:

LEMMA 6.4.4. *Let* $c \geq 2$, $0 < \epsilon < 1$, $0 \leq \delta \leq 1$, *and let* t *be a time such that all agents have at least* $c\log n$ *interactions. Let* $1 \leq k \leq (1-\epsilon)\log n$, *then, the number of agents who hold* $\texttt{group} = k$, *is at least* $L_k = (1-\delta)\frac{n}{2^k}$ *with probability at least* $1-\exp\left(-\frac{\delta^2\cdot n^{\epsilon}}{2}\right)$ *and at most* $U_k = (1+\delta)\frac{n}{2^k}$ *with probability at least* $1-\exp\left(-\frac{\delta^2\cdot n^{\epsilon}}{3}\right)$.

PROOF SKETCH. The fraction of agents with $\texttt{group} = k$ is equal to the fraction of heads in of a binomial distribution $\mathsf{B}\left(n, 2^{-k}\right)$ with $\mu = \frac{n}{2^k}$, so the Chernoff bound applies. A complete proof is given in the full version [49]. □

The following theorem summarizes what we will use later about the distribution of the `group` values and the number of agents residing in each `group` at time t.

THEOREM 6.4.5. *Fix a time $t \geq d \ln n$ for $d > 30$, let M_t^* and $\texttt{FMV}_t$ be the maximum* `group` *value and the* `FMV` *at this time respectively. Then,*

- $0.9 \log n \leq M_t^* < 0.1d \log n$ *with probability at least* $1 - 2 \cdot n^{1-d/10} - 2 \cdot n^{1-\frac{2d}{3}}$.
- $0.9 \log n \leq \texttt{FMV}_t < 3 \log n$ *with probability at least* $1 - 4 \cdot n^{1-\frac{2d}{3}}$.
- *The number of agents who hold* `group` $= k$ *for* $1 \leq k \leq 0.9 \log n$, *is in* $\left[\frac{3 \cdot n}{2^{k+2}}, \frac{5 \cdot n}{2^{k+2}}\right]$ *with probability at least* $1 - 4 \cdot n^{1-\frac{2d}{3}}$.

6.4.3. Group detection. In the previous section, we show that the set of present `group` values among the population will quickly (in $O(\log n)$ time) enter a small interval of values ($[1, 8 \cdot \log n]$) consistent with the population size. In this section, we will prove the following:

- The agents *agree* about the presence of group values in $[\log \ln n, 0.9 \log n]$ after $O(\log n)$ time WHP.
- For a non-existing group value i, each agent will have $\texttt{signals}[i] = 0$ in $O(i + \log n)$ time WHP.

We designed Protocol 34 such that each agent in the i'th `group` *boosts* the associated signal value by setting $\texttt{signals}[i] = B_i$ (recall $B_i = \Theta(i)$). We will show by having at least L_i agents boosting $\texttt{signals}[i]$, the whole population learns about the existence of the i'th `group` in $O(\log n)$ time with high probability. Intuitively, although $\texttt{signals}[i]$ starts lower than $\texttt{signals}[j]$ for $i < j$, so potentially dies out more quickly, it is also boosted more often since more agents have group value i. Concretely, with L_i agents responsible to boost signal i, and for all indices $\log \ln n < i < 0.9 \log n$ in the `signals` of the agents, $\Pr[\texttt{signals}[i] = 0] < \exp\left(-\frac{2^{B_i}}{n/L_i}\right)$.

Intuitively, the next lemma shows that if the `group` values are distributed as in theorem 6.4.4, then the whole population will learn about all the present `group` values above $\log \ln n$ within $O(\log n)$ time. Note that $\Pr[u.\texttt{signals}[i] = j]$ is the probability that the agent u has value j in the i^{th} index of its `signals`. The following lemma is a restatement from [4, Section 5.1]. The proof appears in the full version [49].

LEMMA 6.4.6. *In the execution of Protocol 34, suppose that for each group value* $\log \ln n < i < 0.9 \log n$, *at least A_i agents hold* `group` $= i$. *For every agent $u \in \mathcal{A}$ let* $u.\texttt{signals}[i] = r_i$ *when* $u.\texttt{group} = i$. *Assuming each agent has at least r_i interactions, then for a fixed agent u and index i,* $\Pr[u.\texttt{signals}[i] = 0] \leq \left(1 - \frac{A_i}{n}\right)^{2^{r_i - 1}}$.

To use the previous lemma, we need to make sure that the agents wait for sufficiently long time such that each agent has at least r_i interactions. The next corollary uses theorem 6.4.6 to derive bounds for the entire protocol using bounds from theorem 6.4.4 for the distribution of the `group` values. Also, theorem 6.4.7 takes a union bound over *all* agents and group values i, and uses the concrete value $r_i = B_i = 3 \cdot i + 1$ used in our protocol. The proof appears in the full version [49].

COROLLARY 6.4.7. *For all $i > 0$ and for every agent $u \in \mathcal{A}$, assuming $B_i = 3 \cdot i + 1$ let $u.\texttt{signals}[i] = B_i$ if $u.\texttt{group} = i$. Suppose that for each group value $\log\ln n < i < 0.9 \log n$, at least L_i agents hold $\texttt{group} = i$. Let $\beta \geq 8$; then after $\beta \log n$ time, we have:*

$$\Pr\left[(\exists u \in \mathcal{A})(\exists i \in \{\log\ln n, \ldots, 0.9\log n\})\ u.\texttt{signals}[i] = 0\right] \leq 2 \cdot n^{1-0.9\beta}$$

Finally, we show that when there is no agent holding $\texttt{group} = i$, then $\texttt{signals}[i]$ will become zero in all agents "quickly" with an arbitrarily large probability. To be precise, with no agent boosting signal i, $\Pr\left[u.\texttt{signals}[i] = 0\right] \geq 1 - n^{-\alpha}$ within $\Theta(B_i + \alpha \ln n)$ time WHP in which B_i is the maximum value for signal i. The lemma is a restatement from [34, Lemma 3.3] and [4, Lemma 1].

LEMMA 6.4.8. *For every agent $u \in \mathcal{A}$ let $u.\texttt{signals}[i] = B_i$ when $u.\texttt{group} = i$. Assume that no agent sets its `group` to i from this point on. Then for all $\alpha \geq 1$, all agents will have $\texttt{signals}[i] = 0$ after $3n \ln\left(n^\alpha \cdot 3^{B_i}\right)$ interactions with probability at least $1 - n^{-\alpha}$.*

PROOF. Set $t = 3n \ln\left(n^\alpha \cdot 3^{B_i}\right)$ and $R_{max} = B_i$ in the proof of [34, Lemma 3.3]. □

6.4.4. Dynamic size counting protocol analysis. Recall that `estimate` denote the estimate of $\log n$ in agents' memory, and n is the true population size. In the previous section, we show that the set of present `group` values among the population will quickly (in $O(\log n)$ time) enter a small sub-interval of consecutive values in $[1, 3 \cdot \log n]$ consistent with the population size. This section will show that the `group` values will remain in that interval (with high probability for polynomial time). Moreover, the following two lemmas show how the agents update their `estimate` if it is far from $\log n$. The proofs appear in the full version [49].

Assuming the agents' `estimate` is much smaller than $\log n$, the next lemma shows that all the agents will notice the large gap between `estimate` and `FMV`. Hence, they will re-calculate their population size estimate.

LEMMA 6.4.9. *Let* $M = \max_{u \in \mathcal{A}} u.\texttt{estimate}$. *Assuming* $M \leq 0.22 \log n$, *then the whole population will enter* WaitingPhase *in* $O(\log n)$ *time with probability at least* $1 - O(n^{-2})$.

For the other direction, assume the population size estimate in agents' memory is much larger than $\log n$. We prove in the following lemma that all the agents will notice the large gap between `estimate` and `FMV`. Hence, they will re-calculate their population size estimation.

Note that in theorem 6.4.7, we proved for all `group` values i for $\log \ln n \geq i$, the `signals`$[i]$ will have a positive value in $O(\log n)$ time. However, we could not prove the same bound for values less than $\log \ln n$. So, inevitably the agents ignore their `signals` for values that are less than $\log \ln n$. Since the agents have no access to the value of $\log n$, they have to use `estimate` as an approximation of $\log n$. Thus, they ignore indices that are less than $\log M$ in `signals`: making `FMV` a function of $\max(\log M, \log n)$. For example, if the true population size is n but $M > 2^n$, then the agents should ignore the appearance of a zero in their `signals` for all indices i that are $\leq \log(M) = n$. The correct `FMV` happens at index $j = \Theta(\log n)$, but the agents stay in the NormalPhase as long as `signals`$[i]$ for $i \geq n$ are positive. In this scenario, it takes $O(n)$ time for the agents to switch to WaitingPhase since for each `signals`$[i]$, it takes $O(i)$ time to hit zero.

This scenario is inevitable with our current detection scheme since for indices i that are less than $\log \ln n$, the event of `signals`$[i] = 0$ happens frequently.

LEMMA 6.4.10. *Let* $M = \max_{u \in \mathcal{A}} u.\texttt{estimate}$. *Assuming* $M \geq 7.5 \cdot \log n$, *then the whole population will enter* WaitingPhase *in* $O(\log n + \log M)$ *time with probability at least* $1 - O(n^{-2})$.

In the next theorem (full proof given in [49]), we will show once there is a large gap between the maximum `estimate` among the population and the true value of $\log n$, the agents update their estimate in $O(\log n + \log M)$ time.

THEOREM 6.4.11. *Let* $M = \max_{u \in \mathcal{A}} u.\texttt{estimate}$. *Assuming* `estimate` $\geq 7.5 \log n$ *or* `estimate` $\leq 0.2 \log n$, *then every agent replaces its* `estimate` *with a new value that is in* $[\log n - \log \ln n, 2 \log n]$ *with probability* $1 - O(1/n)$ *in* $O(\log n + \log M)$ *time.*

PROOF SKETCH. By theorems 6.4.9 and 6.4.10, once an agent notices the large gap between `estimate` and `FMV`, they switch to WaitingPhase. We set WaitingPhase long enough so when the first agent moves to UpdatingPhase, there is no agent left in the NormalPhase. Thus, they all re-generate a new geometric random variable and store the maximum as their `estimate`. □

Recall that the adversary can initialize agents with large integer values to arbitrarily increase memory usage. Therefore, we should calculate the agents' memory concerning the fields defined in our protocol and the value that the adversary can set in them. Thus, we use s as the largest integer value that the adversary set in the agents' memory.

THEOREM 6.4.12. *Let* $M = \max u.\texttt{estimate}$ *for all* $u \in \mathcal{A}$. *There is a uniform leaderless loosely-stabilizing population protocol that WHP:*

(1) If $M > 7.5 \log n$ *or* $M < 0.2 \log n$ *reaches to a configuration with all agents set their* `estimate` *with a value in* $[\log n - \log \ln n, 2 \log n]$ *in* $O(\log n + \log M)$ *parallel time.*

(2) If $0.75 \log n < M < 2.25 \log n$, *then the agents hold a stable* `estimate` *during the following* $O(n^{15})$ *parallel time.*

(3) Assuming for every agent $u \in \mathcal{A}$, $\max(u.\texttt{estimate}, u.\texttt{GRV}, u.\texttt{group}, u.\texttt{signals}.size()) < s$ *in the initial configuration, then the protocol uses* $O(\log^2(s) + \log n \log\log n)$ *bits per agent.*

6.4.5. Space optimization. In this section, we explain how to reduce the space complexity of the protocol from $O(\log^2(s) + \log n \log\log n)$ to $O(\log^2(s) + (\log\log n)^2)$ bits per agent.

In Protocol 32, the agents keep track of all the present `group` values using an array of size $O(\log n)$ (stored in `signals`) by mapping every $\texttt{group} = i$ to $\texttt{signals}[i]$. We can reduce the space complexity of the protocol by reducing the `signals`' size. Let the agents map a $\texttt{group} = i$ to $\texttt{signals}[\lfloor \log i \rfloor]$. So, instead of monitoring all $O(\log n)$ group values, they keep $O(\log\log n)$ indices in their `signals`. Thus, reducing the space complexity to $O(\log^2(s) + (\log\log n)^2)$ bits per agent.

Recall that in Protocol 32, there are $\approx \frac{n}{2^i}$ agents with $\texttt{group} = i$ for $i \leq 0.9 \log n$ that help keep $\texttt{signals}[i]$ positive. However, with this technique, there will be $\approx \sum_{i=2^j}^{2^{j+1}} \frac{n}{2^i}$ agents that are helping $\texttt{signals}[i]$ to stay positive. So, every lemma in section 6.4.3 about Protocol 34 holds. Finally, we update Protocol 36 so that the agents compare their `estimate` with $2^{\texttt{LFMV}}$ in which `LFMV` is the smallest index $i > \log\log M$ such that $\texttt{signals}[i] = 0$. On the negative side of this optimization, we get a less sensitive protocol with respect to the gap between agents' `estimate` and $\log n$.

THEOREM 6.4.13. *Let* $M = \max u.\texttt{estimate}$ *for all* $u \in \mathcal{A}$. *There is a uniform leaderless loosely-stabilizing population protocol that WHP:*

(1) If $M > 15 \log n$ *or* $M < 0.1 \log n$ *reaches to a configuration with all agents set their* `estimate` *with a value in* $[\log n - \log \ln n, 2 \log n]$ *in* $O(\log n + \log M)$ *parallel time.*

(2) If $0.75 \log n < M < 2.17 \log n$*, then the agents hold a stable* `estimate` *during the following* $O(n^{15})$ *parallel time.*

(3) Assuming for every agent $u \in \mathcal{A}$*,* $\max(u.\texttt{estimate}, u.\texttt{GRV}, u.\texttt{group}, u.\texttt{signals}.size()) < s$ *in the initial configuration, then the protocol uses* $O(\log^2(s) + (\log\log n)^2)$ *bits per agent.*

6.5. Conclusion and open problems

In this chapter, we introduced the dynamic size counting problem. Assuming an adversary who can add or remove agents, the agents must update their `estimate` according to the changes in the population size. There are several open questions related to this problem.

Reducing convergence time. Our protocol's convergence time depends on both the previous (n_{prev}) and next (n_{next}) population sizes, though exponentially less on the former: $O(\log n_{\text{next}} + \log\log n_{\text{prev}})$. Is there a protocol with optimal convergence time $O(\log n_{\text{next}})$?

Increasing holding time. theorem 6.2.4 states that the holding time must be finite, but it is likely that much longer holding times than $\Omega(n^c)$ for constant c are achievable. For the loosely-stabilizing leader election problem, there is a provable tradeoff in the sense that the holding time is at most exponential in the convergence time [**65**, **92**]. Does a similar tradeoff hold for the dynamic size counting problem?

Reducing space. Our main protocol uses $O(s + (\log n)^{\log n})$ states (equivalent to $O(\log^2(s) + \log n \log\log n)$ bits). In section 6.4.5, we showed how we can reduce the state complexity of our protocol to $o(n^\epsilon)$ (equivalent to $O(\log^2(s) + (\log\log n)^2)$ bits) by mapping more than one `group` to each index of the `signals`. With this trick, we reduce the size of the `signals` from $O(\log n)$ to $O(\log\log n)$. Another interesting idea is to replace our $O(\log n)$ detection scheme to $O(1)$ detection protocol of [**55**] which puts a constant threshold on the values stored in each index. So, it may be possible to reduce the space complexity even more to $O(c^{O(\log n)})$ (with all $O(\log n)$ indices present) or $O(c^{O(\log\log n)}) = \text{polylog}(n)$ (using our optimization technique to have $O(\log\log n)$ indices in the `signals`).

However, the current protocol of [**55**] has a one-sided error that makes it hard to compose with our protocol. With probability $\epsilon > 0$, the agents might say signal i has disappeared even though there exists agents with $\texttt{group} = i$ in the population.

Additionally, in the presence of a uniform self-stabilizing synchronization scheme, one could think of consecutive rounds of independent size computation. The agents update their output if the new computed population size drastically differs from the previously computed population size. Note that the self-stabilizing clock must be independent of the population size since we allow the adversary to change the value of $\log n$ by adding or removing agents. To the best of our knowledge, there is no such synchronization scheme available to population protocols.

Bibliography

[1] Scott Aaronson. Computational complexity and the anthropic principle, 2006. https://www.scottaaronson.com/talks/anthropic.html.

[2] Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. Time-space trade-offs in population protocols. In *SODA 2017: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2560–2579. SIAM, 2017.

[3] Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *SODA 2018: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2221–2239. SIAM, 2018.

[4] Dan Alistarh, Bartłomiej Dudek, Adrian Kosowski, David Soloveichik, and Przemysław Uznański. Robust detection in leak-prone population protocols. In *DNA Computing and Molecular Programming*, pages 155–171. Springer International Publishing, 2017.

[5] Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *Proceedings, Part II, of the 42nd International Colloquium on Automata, Languages, and Programming - Volume 9135*, ICALP 2015, page 479–491. Springer-Verlag, 2015.

[6] Dan Alistarh and Rati Gelashvili. Recent algorithmic advances in population protocols. *SIGACT News*, 49(3):63–73, oct 2018.

[7] Dan Alistarh, Rati Gelashvili, and Milan Vojnović. Fast and exact majority in population protocols. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, page 47–56, New York, NY, USA, 2015. Association for Computing Machinery.

[8] Dan Alistarh, Martin Töpfer, and Przemysław Uznański. Fast and robust comparison in population protocols. In *PODC 2021: The ACM Symposium on Principles of Distributed Computing*, 2021.

[9] Talley Amir, James Aspnes, David Doty, Mahsa Eftekhari, and Eric Severson. Message Complexity of Population Protocols. In *34th International Symposium on Distributed Computing (DISC 2020)*, volume 179, pages 6:1–6:18, 2020.

[10] 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.

[11] Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *25th annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 292–299. ACM Press, 2006.

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

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

[14] Dana Angluin, James Aspnes, Michael J. Fischer, and Hong Jiang. Self-stabilizing population protocols. *ACM Trans. Auton. Adapt. Syst.*, 3(4):1–28, 2008.

[15] James Aspnes, Joffroy Beauquier, Janna Burman, and Devan Sohier. Time and space optimal counting in population protocols. In *20th International Conference on Principles of Distributed Systems (OPODIS 2016)*, volume 70, pages 13:1–13:17, 2017.

[16] Joffroy Beauquier, Janna Burman, Simon Claviere, and Devan Sohier. Space-optimal counting in population protocols. In *DISC 2015: International Symposium on Distributed Computing*, pages 631–646. Springer, 2015.

[17] Joffroy Beauquier, Julien Clement, Stephane Messika, Laurent Rosaz, and Brigitte Rozoy. Self-stabilizing counting in mobile sensor networks with a base station. In *Distributed Computing*, pages 63–76, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[18] Amanda Belleville, David Doty, and David Soloveichik. Hardness of computing and approximating predicates and functions with leaderless population protocols. In *ICALP 2017: 44th International Colloquium on Automata, Languages, and Programming*, volume 80 of *LIPIcs*, pages 141:1–141:14, 2017.

[19] Amanda Belleville, David Doty, and David Soloveichik. Hardness of computing and approximating predicates and functions with leaderless population protocols. In *ICALP 2017: 44th International Colloquium on Automata, Languages, and Programming*, volume 80 of *LIPIcs*, pages 141:1–141:14, 2017.

[20] Stav Ben-Nun, Tsvi Kopelowitz, Matan Kraus, and Ely Porat. An $O(\log^{3/2} n)$ parallel time population protocol for majority with $O(\log n)$ states. In *PODC 2020: Proceedings of the 39th Symposium on Principles of Distributed Computing*, page 191–199. Association for Computing Machinery, 2020.

[21] P. Berenbrink, T. Friedetzky, D. Kaaser, and P. Kling. Tight and simple load balancing. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 718–726, 2019.

[22] Petra Berenbrink, Felix Biermeier, Christopher Hahn, and Dominik Kaaser. Loosely-stabilizing phase clocks and the adaptive majority problem. In *SAND 2021: 1st Symposium on Algorithmic Foundations of Dynamic Networks*, 2021.

[23] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: The heavily loaded case. *SIAM Journal on Computing*, 35(6):1350–1385, 2006.

[24] Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. A Population Protocol for Exact Majority with $O(\log^{5/3} n)$ Stabilization Time and Theta(log n) States. In *32nd International Symposium on Distributed Computing (DISC 2018)*, volume 121, pages 10:1–10:18, 2018.

[25] Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. Time-space trade-offs in population protocols for the majority problem. *Distributed Computing*, Aug 2020.

[26] Petra Berenbrink, George Giakkoupis, and Peter Kling. Optimal time and space leader election in population protocols. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC, page 119–129, New York, NY, USA, 2020. Association for Computing Machinery.

[27] Petra Berenbrink, Dominik Kaaser, Peter Kling, and Lena Otterbach. Simple and efficient leader election. In *SOSA 2018: The 1st Symposium on Simplicity in Algorithms*, pages 9:1–9:11, 2018.

[28] Petra Berenbrink, Dominik Kaaser, and Tomasz Radzik. On counting the population size. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, page 43–52. Association for Computing Machinery, 2019.

[29] Petra Berenbrink, Ralf Klasing, Adrian Kosowski, Frederik Mallmann-Trenn, and Przemysław Uznański. Improved analysis of deterministic load-balancing schemes. *ACM Trans. Algorithms*, 15(1), November 2018.

[30] Andreas Bilke, Colin Cooper, Robert Elsässer, and Tomasz Radzik. Brief announcement: Population protocols for leader election and exact majority with $O(\log^2 n)$ states and $O(\log^2 n)$ convergence time. In *PODC 2017: Proceedings of the ACM Symposium on Principles of Distributed Computing*, page 451–453. Association for Computing Machinery, 2017.

[31] Michael Blondin, Javier Esparza, Stefan Jaax, and Antonín Kučera. Black ninjas in the dark: Formal analysis of population protocols. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, page 1–10, 2018.

[32] Olivier Bournez, Jérémie Chalopin, Johanne Cohen, Xavier Koegler, and Mikael Rabie. Population protocols that correspond to symmetric games. *International Journal of Unconventional Computing*, 9, 2013.

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

[34] Janna Burman, Ho-Lin Chen, Hsueh-Ping Chen, David Doty, Thomas Nowak, Eric Severson, and Chuan Xu. Time-optimal self-stabilizing leader election in population protocols. In *PODC 2021: Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 33–44. ACM, 2021.

[35] Costas Busch and Dariusz R. Kowalski. Byzantine-resilient population protocols, 2021.

[36] Shukai Cai, Taisuke Izumi, and Koichi Wada. Space complexity of self-stabilizing leader election in passively-mobile anonymous agents. In Shay Kutten and Janez Žerovnik, editors, *Structural Information and Communication Complexity*, pages 113–125, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[37] Ioannis Chatzigiannakis, Othon Michail, Stavros Nikolaou, Andreas Pavlogiannis, and Paul G. Spirakis. Passively mobile communicating machines that use restricted space. *Theoretical Computer Science*, 412(46):6469–6483, October 2011.

[38] Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, 2014. Special issue of invited papers from DNA 2012.

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

[40] Anne Condon, Monir Hajiaghayi, David Kirkpatrick, and Ján Maňuch. Approximate majority analyses using tri-molecular chemical reaction networks. *Natural Computing*, 19(1):249–270, 2020.

[41] Eric Severson David Doty, Mahsa Eftekhari. https://github.com/UC-Davis-molecular-computing/ppsim/blob/main/examples/majority.ipynb.

[42] Eric Severson David Doty, Mahsa Eftekhari. https://github.com/eftekhari-mhs/population-protocols/tree/master/Exact_Majority.

[43] Eric Severson David Doty, Mahsa Eftekhari. https://github.com/eftekhari-mhs/population-protocols/tree/master/Exact_Majority/animated_simulations.

[44] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Eric Ruppert. When birds die: Making population protocols fault-tolerant. In *Distributed Computing in Sensor Systems*, pages 51–66, 2006.

[45] Giuseppe A Di Luna, Paola Flocchini, Taisuke Izumi, Tomoko Izumi, Nicola Santoro, and Giovanni Viglietta. Population protocols with faulty interactions: the impact of a leader. *Theoretical Computer Science*, 754:35–49, 2019.

[46] Giuseppe A Di Luna, Paola Flocchini, Taisuke Izumi, Tomoko Izumi, Nicola Santoro, and Giovanni Viglietta. Fault-tolerant simulation of population protocols. *Distributed Computing*, 33(6):561–578, 2020.

[47] David Doty and Mahsa Eftekhari. Efficient size estimation and impossibility of termination in uniform dense population protocols. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, page 34–42. Association for Computing Machinery, 2019.

[48] David Doty and Mahsa Eftekhari. A survey of size counting in population protocols. *Theoretical Computer Science*, 894:91–102, 2021. Building Bridges – Honoring Nataša Jonoska on the Occasion of Her 60th Birthday.

[49] David Doty and Mahsa Eftekhari. Dynamic Size Counting in Population Protocols. In James Aspnes and Othon Michail, editors, *1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2022)*, volume 221, pages 13:1–13:18, 2022.

[50] David Doty, Mahsa Eftekhari, Leszek Gasieniec, Eric E. Severson, Grzegorz Stachowiak, and Przemyslaw Uznanski. A time and space optimal stable population protocol solving exact majority. In *FOCS '21: 62nd Annual IEEE Symposium on Foundations of Computer Science*, 2022.

[51] David Doty, Mahsa Eftekhari, Othon Michail, Paul G. Spirakis, and Michail Theofilatos. Brief Announcement: Exact Size Counting in Uniform Population Protocols in Nearly Logarithmic Time. In *32nd International Symposium on Distributed Computing (DISC 2018)*, volume 121, pages 46:1–46:3, 2018.

[52] David Doty and Monir Hajiaghayi. Leaderless deterministic chemical reaction networks. *Natural Computing*, 14(2):213–223, 2015.

[53] David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 31(4):257–271, 2018. Special issue of invited papers from DISC 2015.

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

[55] Bartłomiej Dudek and Adrian Kosowski. Universal protocols for information dissemination using emergent signals. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 87–99, New York, NY, USA, 2018. Association for Computing Machinery.

[56] Bennett Eisenberg. On the expectation of the maximum of iid geometric random variables. *Statistics & Probability Letters*, 78(2):135 – 143, 2008.

[57] Elsässer, Robert and Radzik, Tomasz and others. Recent results in population protocols for exact majority and leader election. *Bulletin of EATCS*, 3(126), 2018.

[58] Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

[59] Leszek Gąsieniec, David Hamilton, Russell Martin, Paul G Spirakis, and Grzegorz Stachowiak. Deterministic population protocols for exact majority and plurality. In *20th International Conference on Principles of Distributed Systems*, 2016.

[60] Leszek Gąsieniec and Grzegorz Stachowiak. Fast space optimal leader election in population protocols. In *SODA 2018: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, page 265–266, USA, 2018. Society for Industrial and Applied Mathematics.

[61] Leszek Gasieniec, Grzegorz Stachowiak, and Przemyslaw Uznanski. Almost logarithmic-time space optimal leader election in population protocols. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '19, page 93–102. Association for Computing Machinery, 2019.

[62] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

[63] Shafi Goldwasser, Rafail Ostrovsky, Alessandra Scafuro, and Adam Sealfon. Population stability: regulating size in the presence of an adversary. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 397–406. ACM, 2018.

[64] Rachid Guerraoui and Eric Ruppert. Names trump malice: Tiny mobile agents can tolerate byzantine failures. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, pages 484–495, 2009.

[65] Taisuke Izumi. On space and time complexity of loosely-stabilizing leader election. In *Structural Information and Communication Complexity*, pages 299–312. Springer International Publishing, 2015.

[66] Tomoko Izumi, Keigo Kinpara, Taisuke Izumi, and Koichi Wada. Space-efficient self-stabilizing counting population protocols on mobile sensor networks. *Theoretical Computer Science*, 552:99–108, 2014.

[67] Svante Janson. Tail bounds for sums of geometric and exponential variables. *Statistics and Probability Letters*, 135:1–6, April 2018.

[68] Márk Jelasity and Alberto Montresor. Epidemic-style proactive aggregation in large overlay networks. In *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, pages 102–109, 2004.

[69] Adrian Kosowski and Przemyslaw Uznanski. Population protocols are fast. *CoRR*, abs/1802.06872, 2018.

[70] Guy Louchard and Helmut Prodinger. The moments problem of extreme-value related distribution functions. *Algorithmica*, 2004.

[71] Guy Louchard, Helmut Prodinger, and Mark Daniel Ward. The number of distinct values of some multiplicity in sequences of geometrically distributed random variables. In *Discrete Mathematics and Theoretical Computer Science*, pages 231–256. Discrete Mathematics and Theoretical Computer Science, 2005.

[72] László Lovász and Peter Winkler. Reversal of markov chains and the forget time. *Combinatorics, Probability and Computing*, 7(2):189–204, 1998.

[73] George B Mertzios, Sotiris E Nikoletseas, Christoforos L Raptopoulos, and Paul G Spirakis. Determining majority in networks with local interactions and very small local memory. In *International Colloquium on Automata, Languages, and Programming*, pages 871–882. Springer, 2014.

[74] Othon Michail. Terminating distributed construction of shapes and patterns in a fair solution of automata. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 37–46, 2015. Also in *Distributed Computing*, 2017.

[75] Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. Terminating population protocols via some minimal global knowledge assumptions. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 77–89. Springer, 2012.

[76] Othon Michail, Paul G. Spirakis, and Michail Theofilatos. Simple and fast approximate counting and leader election in populations. In *Stabilization, Safety, and Security of Distributed Systems*, pages 154–169. Springer International Publishing, 2018.

[77] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press, 2005.

[78] Y. Mocquard, E. Anceaume, and B. Sericola. Optimal proportion computation with population protocols. In *IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pages 216–223, Oct 2016.

[79] Yves Mocquard, Emmanuelle Anceaume, James Aspnes, Yann Busnel, and Bruno Sericola. Counting with population protocols. In *14th IEEE International Symposium on Network Computing and Applications*, pages 35–42, 2015.

[80] Yves Mocquard, Bruno Sericola, and Emmanuelle Anceaume. Population protocols with convergence detection. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–8. IEEE, 2018.

[81] Yves Mocquard, Bruno Sericola, and Emmanuelle Anceaume. Explicit and tight bounds of the convergence time of average-based population protocols. In *International Colloquium on Structural Information and Communication Complexity*, pages 357–360. Springer, 2019.

[82] Yves Mocquard, Bruno Sericola, Samantha Robert, and Emmanuelle Anceaume. Analysis of the propagation time of a rumour in large-scale distributed systems. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pages 264–271, 2016.

[83] Yves Mocquard, Bruno Sericola, Frédérique Robin, and Emmanuelle Anceaume. Stochastic Analysis of Average Based Distributed Algorithms. *Journal of Applied Probability*, 58(2):394 – 410, June 2021.

[84] Helmut Prodinger. Philippe flajolet's early work in combinatorics. *arXiv preprint arXiv:2103.15791*, 2021.

[85] Philippe Rigollet. Lecture notes for MIT course 18.s997: High dimensional statistics, 2015. URL: https://ocw.mit.edu/courses/mathematics/18-s997-high-dimensional-statistics-spring-2015/lecture-notes/.

[86] Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 341–350. IEEE, 2012.

[87] David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4):615–633, 2008.

[88] Niranjan Srinivas, James Parkin, Georg Seelig, Erik Winfree, and David Soloveichik. Enzyme-free nucleic acid dynamical systems. *Science*, 358(6369), 2017.

[89] Yuichi Sudo, Ryota Eguchi, Taisuke Izumi, and Toshimitsu Masuzawa. Time-optimal loosely-stabilizing leader election in population protocols. In *DISC 2021: The 35th International Symposium on Distributed Computing*, 2021.

[90] Yuichi Sudo, Junya Nakamura, Yukiko Yamauchi, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Loosely-stabilizing leader election in population protocol model. In *Structural Information and Communication Complexity*, pages 295–308, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[91] Yuichi Sudo, Fukuhito Ooshita, Taisuke Izumi, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Logarithmic expected-time leader election in population protocol model. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC, page 60–62, 2019.

[92] Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, Toshimitsu Masuzawa, Ajoy K. Datta, and Lawrence L. Larmore. Loosely-Stabilizing Leader Election with Polylogarithmic Convergence Time. In *22nd International Conference on Principles of Distributed Systems (OPODIS 2018)*, volume 125, pages 30:1–30:16, 2018.

[93] Vito Volterra. Variazioni e fluttuazioni del numero d'individui in specie animali conviventi. *Mem. Acad. Lincei Roma*, 2:31–113, 1926.