Polynomial Anonymous Dynamic Distributed Computing Without a Unique Leader

Dariusz R. Kowalski
Department of Computer Science, University of Liverpool, UK
SWPS University of Social Sciences and Humanities, Warsaw, Poland
D.Kowalski@liverpool.ac.uk

Miguel A. Mosteiro
Computer Science Department, Pace University, New York, NY, USA
mmosteiro@pace.edu

Abstract

Counting the number of nodes in Anonymous Dynamic Networks is enticing from an algorithmic perspective: an important computation in a restricted platform with promising applications. Starting with Michail, Chatzigiannakis, and Spirakis [18], a flurry of papers sped up the running time guarantees from doubly-exponential to polynomial [16]. There is a common theme across all those works: a distinguished node is assumed to be present, because Counting cannot be solved deterministically without at least one.

In the present work we study challenging questions that naturally follow: how to efficiently count with more than one distinguished node, or how to count without any distinguished node. More importantly, what is the minimal information needed about these distinguished nodes and what is the best we can aim for (count precision, stochastic guarantees, etc.) without any. We present negative and positive results to answer these questions. To the best of our knowledge, this is the first work that addresses them.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Anonymous Dynamic Networks, Counting, distributed algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.147

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Funding This work was supported by the National Science Center Poland (NCN) grant 2017/25/B/ST6/02553; the UK Royal Society International Exchanges 2017 Round 3 Grant #170293; and Pace University SR Grant and Kenan Fund.

1 Introduction

The recent excitement around the problem of *Counting*, i.e., computing the number of nodes of an Anonymous Dynamic Network (ADN) comes as no surprise. On one hand, knowing the number of processors is a fundamental requirement to decide termination in a myriad of distributed algorithms. On the other hand, ADN is a very restrictive scenario from an algorithmic perspective: network nodes do not have identifiers and communication links may change arbitrarily and continuously over time. The combination of an important problem with harsh computational conditions is any algorithmist's delight.

Node anonymity in ADNs is motivated by expected applications of such communication infrastructure. For instance, in ad-hoc networks embedded in the Internet of Things, nodes may have to be deployed in a massive scale, and having unique identifiers may simply be impractical or inconvenient. Moreover, low node-cost expectations may introduce uncertainty about the number of nodes that will effectively startup. Hence, the need of Counting.

Strikingly, the progress on deterministic Counting speed-ups over various works ranged over a broad spectrum: from unbounded [11, 12] to polynomial time [16], going through doubly-exponential [11] and exponential time [10, 7]. In all that fruitful work, the ADN model has been equipped with one distinguishable node[1]. The assumption is well motivated: in a seminal paper [18], it was shown that, without at least one such node, Counting cannot be solved deterministically. But what if we have more than one special node? Do these nodes really need to be *special*? In other words, is it enough to put one of two different programs on each of the nodes, that otherwise are all identical? Moreover, can we let the nodes choose at random which program to run and have no special nodes? To the best of our knowledge, this is the first work that considers these questions.

As the more general case where all nodes are identical, only differentiated by the program they run, let the set of n network nodes be formed by ℓ *black* nodes (the “special” ones) and $n - \ell$ *white* nodes (the “regular” ones). **Our first contribution** is negative results. On one hand, if ℓ is unknown, Counting cannot be solved deterministically. On the other hand, for randomized Counting algorithms, we show that if ℓ is unknown or zero, there exist executions when the algorithm does not stop.

Even knowing how many black nodes are in the network, straightforward application of previous ideas for Counting is not clear. Indeed, each black node may carry its own count, but how do we combine or compare final counts? Even passing messages among black nodes is challenging, because black nodes are also indistinguishable among them and communication links change arbitrarily. For instance, a black node is not able to tell whether a received message is even its own, coming back after being previously sent for dissemination.

Our second contribution is a deterministic Counting protocol that computes exactly the number of network nodes. Our protocol uses no information about the network, except the number of black nodes $\ell \geq 1$. After completing its execution, all nodes obtain the exact size of the network and stop. Moreover, they stop all at the same time, allowing the algorithm to be concatenated with other computations.

This protocol resembles our Methodical Counting protocol presented in [16]. So, we call it Methodical multi-Counting (MMC). However, it is not a simple combination of multiple instances of Methodical Counting to handle multiple black nodes. We overcome the challenge of how to combine the actions of multiple indistinguishable black nodes by careful design of a set of alarms, so that *all* black nodes can simultaneously detect when a running estimate of the size is correct. Moreover, the asymptotic performance of MMC is $O(n^{4+\epsilon}(\log^3 n)/\ell)$, for an arbitrarily small $\epsilon > 0$. This is a speed-up by a factor arbitrarily close to $n\ell/\log n$ with respect to Methodical Counting. That is, even for $\ell = 1$, MMC is faster than the best previous work.

In face of the impossibility of deterministic Counting without a distinguished node [18], an enticing question is what is possible introducing randomness. **Our third contribution** is a Counting protocol that computes exactly the number of nodes in an ADN where $\ell = 0$ (no special nodes). That is, we show that randomness breaks the impossibility result of [18], and for the first time it is possible to consider an ADN model where *all* nodes are identical, indistinguishable, and run the same program. Our protocol, called Leader-less Methodical Counting (LLMC), is Monte Carlo; i.e., there is a small probability $\epsilon > 0$ of obtaining the wrong size, and the running time of $O(n^{4+\epsilon} \log^3 n)$ holds with probability $1 - \epsilon$.

[1] Exactly one, usually called *leader*. We refrain from using the name leader to avoid confusion: in our model we may have more than one, and in our algorithm they are not going to select a single one among them. Moreover, they cannot even be local leaders, because due to ADN’s dynamicity they may all be connected being their own (local) leaders.

To the best of our knowledge, this is the first comprehensive study of Counting in ADNs where ℓ may be different than one.

2 Previous Work

A comprehensive overview of work related to ADNs can be found in a survey by Casteigts et al. [5] and references in the papers cited here.

With respect to lower bounds, it was proved in [9] that at least $\Omega(\log n)$ rounds are needed, even if D is constant. Also, $\Omega(\mathcal{D})$ is a lower bound since at least one node needs to hear about all other nodes to obtain the right count.

Counting and *Naming* was already studied in [18] for dynamic and static networks, showing that it is impossible to solve Counting without the presence of a distinguished node, even if nodes do not move. The Counting protocol requires knowledge of an upper bound on Δ, and obtains only an upper bound, which may be as bad as exponential.

Conscious Counting [11] computes the exact count, but it needs to start from an upper bound, and it takes exponential time only if the size upper bound is tight. In the same work and follow-up papers [12, 13], more challenging scenarios where Δ is unknown are studied, but protocols either do not terminate [11], or the protocol is terminated heuristically [13]. In experiments [13], such heuristic was found to perform well on dense topologies, but for other topologies the error rate was high. Another protocol in [12] is shown to terminate eventually, without running-time guarantees and under the assumption of having for each node an estimate of the number of neighbors in each round. In [18] it was conjectured that some knowledge of the network such as the latter would be necessary, but the conjecture was disproved later in [10]. On the other hand the protocol in [10] requires exponential space.

Incremental Counting, presented recently in [19], reduced exponentially the best-known running time guarantees. The protocol obtains the exact count, all nodes terminate simultaneously, the topology dynamics is only limited to 1-interval connectivity, it only requires polynomial space, and it only requires knowledge of the dynamic maximum degree Δ. The running time is still exponential, but reducing from doubly-exponential was an important step towards understanding the complexity of Counting.

In a follow-up paper [6], Incremental Counting was tested experimentally showing a promising polynomial behavior. The study was conducted on pessimistic inputs designed to slow the convergence, such as bounded-degree trees rooted at the leader uniformly chosen at random for each round, and a single path starting at the leader with non-leader nodes permuted uniformly at random for each round. The protocol was also tested on static versions of the inputs mentioned, classic random graphs, and networks where some disconnection is allowed. The results exposed important observations. Indeed, even for topologies that stretch the dynamic diameter, the running times obtained are below Δn^3. It was also observed that random graphs, as used in previous experimental studies [13], reduce the convergence time, and therefore are not a good choice to indicate worst-case behavior. These experiments showed good behavior even for networks that sometimes are disconnected, indicating that more relaxed models of dynamics, such as (α, β)-connectivity [14, 15], are worth to study.

All in all, the experiments in [6] showed that Incremental Counting behaves well in a variety of pessimistic inputs, but not having a proof of what a worst-case input looks like, and being the experiments restricted to a range of values of n far from the expected massive size of an ADN, a theoretical proof of polynomial time remained an open problem even from a practical perspective.

A polynomial Counting algorithm was presented in a manuscript [2], relying on the availability of an algorithm to compute average with polynomial convergence time. Such average computation is modeled as a Markov chain with underlying doubly-stochastic matrix, which requires topology information within two hops (cf. [21]). In the pure model of ADN, such information is not available, and gathering it may not be possible due to possible topology changes from round to round.

Recently, we presented the first polynomial-time deterministic Counting algorithm for ADNs in [16], called Methodical Counting. Unlike previous works, Methodical Counting does not require any knowledge of network characteristics, such as dynamic maximum degree or an upper bound on the size. That is, it works in the pure model of ADN. Like previous works, Methodical Counting requires the presence of a distinguished node. In the present work, we generalize that assumption assuming the presence of $\ell \geq 1$ distinguished nodes. As in [16], we leverage previous work on lazy random walks to analyze MMC, but the alarms to detect wrong computations had been completely re-designed to deal with multiple distinguished nodes. Moreover, with respect to Methodical Counting, MMC achieves a $\Omega((\log^2 n)/(n\ell))$ speed-up. That is, even for $\ell = 1$, MMC also provides a speed-up with respect to previous work.

In the same paper [16], we also presented extensions of Methodical Counting to compute more complex functions, such as the sum of input values held by nodes and other algebraic and Boolean functions.

With respect to randomized Counting, a linear Counting algorithm for dynamic networks was presented in [17]. The algorithm requires unique identifiers (i.e., it is not applicable to ADNs), knowledge of an upper bound on the size of the network, and only guarantees an approximation to the network size. To the best of our knowledge, no randomized Counting algorithms for ADNs have been studied before.

Other studies also dealing with the time complexity of information gathering exist [8, 3, 22, 4, 20, 23], but include in their model additional assumptions, such as the network having the same topology frequently enough or node identifiers.

3 Model, Problem, and Notation

We define the Counting problem as follows. An algorithm $\mathcal{A}$ solves the *Counting Problem* if, after completing its execution, all network nodes running $\mathcal{A}$ have obtained the size of the network and stop (not necessarily concurrently). Notice that we focus on *exact* Counting. That is, all nodes obtain the exact size of the network n, rather than an approximation as other works. We define now a class of Counting algorithms.

For a given algorithm $\mathcal{A}$, let an *execution* of $\mathcal{A}$ be a sequence of steps of $\mathcal{A}$ followed in one of the possible sequence of choices made by $\mathcal{A}$. Let $\mathcal{X}(\mathcal{A})$ be the set of all possible executions of $\mathcal{A}$. An algorithm $\mathcal{A}$ is called *eventually stopping* if, for all $X \in \mathcal{X}(\mathcal{A})$, X has finite length.

We will model worst case scenarios assuming the presence of an adversary that controls the topology of the network. In particular, we consider the following adversaries.

Let the sequence $\mathcal{E} = \langle E_1, E_2, \dots \rangle$ be the sets of communication links of an ADN for time slots $t_1, t_2, \dots$. Consider the execution of an algorithm $\mathcal{A}$. We say that an adversary is *oblivious* if it determines the sequence $\mathcal{E}$ completely before the execution of $\mathcal{A}$ begins. On the other hand, we say that an adversary is *adaptive* if it determines the sequence $\mathcal{E}$ during the execution of $\mathcal{A}$, according to the actions of $\mathcal{A}$.

Notice that the distinction between oblivious and adaptive makes sense only for randomized algorithms, given that for deterministic algorithms the actions of the algorithm are defined before the execution.

The following model is customary in the ADNs literature. We consider a network composed by a set V of $n > 1$ network *nodes* with processing and communication capabilities. It was shown in [18] that Counting cannot be solved in Anonymous Networks without the availability of at least one distinguished node in the network. Hence, all previous studies of Counting in ADNs included in the model the presence of such node called the *leader*. However, to the best of our knowledge, nothing is known about deterministic Counting in presence of *multiple* distinguished nodes. In this work, we generalize the ADN model assuming that the number of distinguished nodes is $\ell \geq 1$. Aside from the distinction between distinguished and not-distinguished, all nodes are indistinguishable within their group; we call them black nodes and white nodes resp. All black nodes execute exactly the same program, and all white nodes execute exactly the same program. That is, there are no identifiers that allow to distinguish one black (resp. white) node from another black (resp. white) node. (Although we label the nodes throughout the paper for the sake of presentation and analysis.)

Each pair of nodes that are able to communicate define a communication *link*, and the set of links is called the *topology* of the network. The nodes in a communication link are called *neighbors*. The event of sending a message to neighbors is called a *broadcast* or *transmission*. Nodes and links are reliable, in the sense that no communication or node failures occur. Hence, a broadcasted message is received by all neighbors. Moreover, links are *symmetric*, that is, if node a is able to send a message to node b, then b is able to send a message to a.

Without loss of generality, we discretize time in *rounds*. In any given round, a node may broadcast a message, receive all messages from broadcasting neighbors, and carry out some computations, in that order. Time needed for computations is assumed negligible and the size of messages is unbounded.

The set of links among nodes may change from round to round, and nodes have no way of knowing which were the neighbors they had before. These topology changes are arbitrary, limited only to maintain the network connected in each round. That is, at any given round the topology is such that there is a *path*, i.e., a sequence of links, between each pair of nodes, but the set of links may change arbitrarily from round to round. This adversarial model of dynamics is known as 1-*interval connectivity* in [17].

The following notation will be used. The maximum number of neighbors that any node may have at any given time, called the *dynamic maximum degree*, is denoted as Δ or $d_{\max}$ indistinctively. The maximum length of a path between any pair of nodes at any given time is called the *dynamic diameter* and it is denoted as D. The maximum length of an opportunistic path between any pair of nodes over many time slots is called the *chronopath* [14] and it is denoted as $\mathcal{D}$.

4 Impossibility of Counting

Note that the results of this section hold even for static anonymous networks. The proofs of the following lemmas are left to the full version of this paper for brevity.

▶ **Lemma 1.** *For every positive integer ℓ there are two networks of $\ell + 4$ and $2(\ell + 4)$ nodes, respectively, such that:*

(i) *no deterministic algorithm could successfully accomplish Counting on both of them in finite time, even if some ℓ nodes are black in the former and 2ℓ nodes are black in the latter network.*

(ii) *for any randomized algorithm there is an execution in which no node outputs a correct count in a finite time, if no node is initially black in any of them or even if some ℓ nodes are black in the former and 2ℓ nodes are black in the latter network.*

The following results follow immediately from the lemma.

▶ **Theorem 2.** *If the number of black nodes is not given as a parameter, then there is no deterministic algorithm accomplishing Counting in finite time in ADNs.*

▶ **Theorem 3.** *If the number of black nodes is not given as a parameter or if there is no black node, then for every randomized algorithm there is an execution in which no node outputs correct node count in finite time in some fixed anonymous networks.*

▶ **Corollary 4.** *If there is no black node or their number is not explicitly known to the nodes, there is no eventually stopping randomized algorithm accomplishing Counting in ADNs even with approximation smaller than $\sqrt{2}$ and even against an oblivious adversary.*

Since there is no eventually stopping PTAS algorithm for Counting if the exact number of black nodes is unknown or (in case of randomized algorithms) there is no black node, we pursue two directions. One is to design a polynomial-time deterministic algorithm for exact Counting if the exact number of black nodes is apriori known. The other direction is to design a randomized algorithm computing the exact number of nodes with an arbitrary probability $1-\epsilon$, for any $\epsilon \in (0,1)$, even in scenario when no node is black prior the computation. The latter algorithm is polynomial in the sense of expected number of rounds and also holds with high probability (i.e., probability polynomially close to 1 in terms of n).

Remark: the inapproximability result could be extended from $\sqrt{2}$ to *any* constant by considering networks with $c \cdot (\ell + 4)$ nodes, for an arbitrary constant c instead of $c = 2$ used in the above proofs for the ease of arguments.

5 Methodical multi-Counting

In this section we present MMC. For brevity, we give the intuition of the algorithm, leaving the pseudocode to the full version of this paper.

Initially, each of the ℓ black nodes is assigned a potential of 0 and each of the $n-\ell$ white nodes is assigned a potential of ℓ. Then, the algorithm is composed by epochs, each divided into phases composed by rounds of communication. Epoch k corresponds to a size estimate k that is iteratively updated from epoch to epoch until the correct value n is found. Each epoch is divided into p phases. The purpose of each phase is for the black nodes to collect as much potential as possible from white nodes in a mass-distribution fashion as follows.

Each phase is composed by r rounds of communication. In each round, each node broadcasts its potential and receives the potential of all its neighbors. Each node keeps only a fraction $1/d$ of the potentials received. The parameters p, r, and d are functions of k. The specific functions needed to guarantee correctness and saught efficiency are defined in Theorem 14. This varying way of distributing potential is different from previous approaches using mass distribution. After communication, each node updates its own potential accordingly. That is, it adds a fraction $1/d$ of the potentials received, and subtracts a fraction $1/d$ of the potential broadcasted times the number of potentials received. Then, a new round starts. At the end of each phase, each black node “consumes” its potential. That is, it increases an internal accumulator ρ with its current potential, which is zeroed for starting the next phase.

The correctness (or incorrectness) of the estimate is detected by various alarms as follows. A node stops the update of potential described, raises its potential to ℓ, and broadcasts an alarm status “low” in each round until the end of the epoch if any of the following happens: 1) at the end of the first phase its potential is above some threshold τ as defined in Theorem 14, 2) at any round it receives more than $d-1$ messages, or 3) at any round receives an alarm status "low" from one of its neighbors. Case 1) allows the black nodes to detect that the estimate is wrong when $k^{1+\epsilon} < n$ for some $\epsilon > 0$ (Lemmas 10 and 12), case 2) allows the black nodes to detect that d is too small and hence the estimate is low, and case 3) allows dissemination of these alarms. (In "low" status the potential is set to ℓ to facilitate the analysis, but it is not strictly needed by the algorithm.)

At the end of each epoch, each black node checks the value of ρ and updates its status accordingly. If it is within some range, call it Γ, the current estimate is correct and each black node changes its status to "done". Otherwise, the estimate is incorrect. If ρ is below (resp. above) Γ each black node changes its status to "high" (resp. "low") indicating that the estimate is too big (resp. "low"). The case when ρ is below (resp. above) Γ allows to detect when $k > n$ (resp. $k < n \leq k^{1+\epsilon}$) (c.f. Lemmas 7 and 13 respectively.).

After black nodes update their status, the network is flooded with it for k rounds. If $k \geq n$, those rounds are enough for all white nodes to receive the "done" or "high" status. If they receive "done", after completing the k rounds all nodes stop. Otherwise, after completing the k rounds all nodes update k according to status to start a new epoch. If k has not been detected to be greater than n since the computation started, it is doubled for the next epoch, otherwise it is updated as in binary search.

5.1 Analysis of Methodical multi-Counting

In this section we analyze MMC. We use standard notations $\mathbf{I}$ for the unit vector, and L_p for the norm of vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ as $||\mathbf{x}||_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$, for any $p \geq 1$. Only for the analysis, nodes are labeled as $0, 1, 2, \ldots, n-1$. The potential of a node i at the beginning of round s of phase t is denoted as $\Phi_{s,t}[i]$, the potential of all nodes is denoted as a vector $\mathbf{\Phi}_{s,t}$, and the aggregated potential is then $||\mathbf{\Phi}_{s,t}||_1$. The subindices s, t, or both are omitted sometimes for clarity. We will refer to the potential right after the last round of a phase as $\mathbf{\Phi}_{r+1}$. Such round does not exist in the algorithm, but we use this notation to distinguish between the potential right before black nodes consume their own potential and the potential at the beginning of the first round of the next phase.

First, we provide a broad description of our analysis of MMC. Consider the vector of potentials $\mathbf{\Phi}_i$ held by nodes at the beginning of any given phase i. The way that potentials are updated in each round is equivalent to the progression of a d-lazy random walk on the evolving graph underlying the network topology [1], where the initial vector of potentials is equivalent to an initial distribution $\mathbf{\Pi}_i$ on the overall potential $||\mathbf{\Phi}_i||_1$ and the probability of choosing a specific neighbor is $1/d$.

Note that MMC is not a simple "derandomization" of the lazy random walk on evolving graphs. First, in the Anonymous Dynamic Network model neighbors cannot be distinguished, and even their number is unknown at transmission time (only at receiving time the node learns the number of its neighbors). Second, due to unknown network parameters, it may happen in an execution of MMC that the total potential received could be bigger than 1. Third, our algorithm does not know a priori when to terminate and provide a result even with some reasonable accuracy, as the formulas on mixing and cover time of lazy random walks depend on the (a priori unknown) number of nodes n. Nevertheless, we can still use some results obtained in the context of analogous lazy random walks in order to prove useful properties of parts of MMC, namely, some parts in which parameters are temporarily fixed and the number of received messages does not exceed parameter d.

It was shown in [1] that random walks on d-regular explorable evolving graphs have a uniform stationary distribution, and bounds on the mixing and cover time were proved as well. Moreover, it was observed that those properties hold even if the graph is not regular and d is only an upper bound on the degree.[2]

Thus, for the cases where d is an upper bound on the number of neighboring nodes, we analyze the evolution of potentials within each phase leveraging previous work on random walks on evolving graphs. Specifically, we use the following result which is an extension of Corollary 14 in [1].

▶ **Theorem 5** (Corollary 14 in [1]). *After t rounds of a $d_{\max}$-lazy random walk on an evolving graph with n nodes, dynamic diameter D, upper bound on maximum degree $d_{\max}$, and initial distribution $\mathbf{\Pi}_0$, the following holds.*

$$\left|\left|\mathbf{\Pi}_t - \frac{\mathbf{I}}{n}\right|\right|_2^2 \leq \left(1 - \frac{1}{d_{\max} D n}\right)^t \left|\left|\mathbf{\Pi}_0 - \frac{\mathbf{I}}{n}\right|\right|_2^2$$

In between phases, black nodes "consume" their potential, effectively changing the distribution at that point. Then, a new phase starts.

In MMC, given that d is a function of the estimate k, if the estimate is low, there may be inputs for which d is not an upper bound on the number of neighbors. We show in our analysis that in those cases the black nodes detect the error and after some time all nodes increase the estimate.

Structure of the proof

The proof of correctness is structured in the following cases, depending on the relation between the size estimate k and n. For some γ and ϵ, after completing an epoch with size estimate k, we prove that ρ, the potential accumulated by a black node, must be within the following ranges.

$$\begin{aligned}
k = n &\quad\Rightarrow\quad (k-\ell)\left(1 - \frac{1}{k^\gamma}\right) \leq \rho \leq (k-\ell)\left(1 + \frac{1}{k^\gamma}\right) && \text{(Lemma 6)}\\
k > n &\quad\Rightarrow\quad \rho < (k-\ell)\left(1 - \frac{1}{k^\gamma}\right) && \text{(Lemma 13)}\\
k < n \leq k^{1+\epsilon} &\quad\Rightarrow\quad \rho > (k-\ell)\left(1 + \frac{1}{k^\gamma}\right) && \text{(Lemma 7)}
\end{aligned}$$

For the remaining case when $k^{1+\epsilon} < n$, we prove first the following relation between $\Phi_{r+1,1}$, the potential of any node at the end of the first phase, and a threshold τ.

$$\begin{aligned}
k^{1+\epsilon} < n &\quad\Rightarrow\quad \Phi_{r+1,1} > \tau \text{ for at least one node} && \text{(Lemma 10)}\\
k \geq n &\quad\Rightarrow\quad \Phi_{r+1,1} \leq \tau \text{ for all nodes} && \text{(Lemma 11)}
\end{aligned}$$

Thus, if $k^{1+\epsilon} < n$, and only if $k < n$, there is at least one node with potential above τ, which moves to a status "low" and spreads this alarm. We complete the proof with the following.

$$k^{1+\epsilon} < n \quad\Rightarrow\quad \text{all nodes receive an alarm "low" during phase 2} \qquad \text{(Lemma 12)}$$

[2] Their analysis relies on Lemma 12, which bounds the eigenvalues of the transition matrix as long as it is stochastic, connected, symmetric, and non-zero entries lower bounded by $1/d$. Those conditions hold for all the transition matrices, even if the evolving graph is not regular.

Analysis

We start the analysis considering the case $k = n$ in the following lemma. The proofs of the lemmata in this section are left to the full version of this paper for brevity.

▶ **Lemma 6.** *If $d \geq k = n$, for an ADN with $\ell < k$ black nodes, for any $\gamma > 0$ there is a $\alpha \geq \max\{2, 1 + \gamma + \log_k 3\}$ such that, after running the MMC protocol for $p \geq (2\gamma \ln k)/\left(\ell\left(\frac{1}{k} + \frac{1}{k^\alpha}\right)\right)$ phases, each of $r \geq 2\alpha d k^2 \ln k$ rounds, the potential ρ consumed by each of the ℓ black nodes is such that*

$$(k-\ell)\left(1 - \frac{1}{k^\gamma}\right) \leq \rho \leq (k-\ell)\left(1 + \frac{1}{k^\gamma}\right).$$

The previous lemma shows that, after running MMC enough time, if for some black node it is $\rho > (k-\ell)\left(1+\frac{1}{k^\gamma}\right)$ or $\rho < (k-\ell)\left(1-\frac{1}{k^\gamma}\right)$, for some $\gamma > 0$, we know that the estimate k is wrong. However, the complementary case, that is, $(k-\ell)\left(1-\frac{1}{k^\gamma}\right) \leq \rho \leq (k-\ell)\left(1+\frac{1}{k^\gamma}\right)$, may occur even if the estimate is $k \neq n$ and hence the error has to be detected by other means. To prove correctness in that case we further separate the range of k in three cases. The first one, when $k < n \leq k^{1+\epsilon}$, for some $\epsilon > 0$, in the following lemma, which is based on upper bounding the potential left in the system after running MMC long enough. To ensure that $d \geq \Delta + 1$, we restrict $d \geq k^{1+\epsilon}$.

▶ **Lemma 7.** *Under the following conditions $1 < k < n \leq k^{1+\epsilon} \leq d$, $\epsilon > 0$, after running the MMC protocol for $p \geq 2\delta(\ln k)/(\ell\left(1/n + 1/k^\beta\right))$ phases, each of $r \geq 2\beta d k^{2+2\epsilon} \ln k$ rounds, under the following conditions $\beta \geq \log_k(n(2k^\delta + 1))$, $\beta > 2$, $\delta > \log_k(nk^\gamma/(nk^\gamma - (n-1)(k^\gamma+1)))$, and $\gamma > \log_k(n-1)$. Then, the potential ρ consumed by any black node is $\rho > (k-\ell)(1 + 1/k^\gamma)$.*

We now consider the case $k^{1+\epsilon} < n$. First, we prove the following two claims that establish properties of the potential during the execution of MMC. (Recall that we use round $r+1$ to refer to potentials at the end of the phase right before black nodes consume their potential.)

▷ Claim 8. Given an ADN of n nodes running MMC with parameter d, for any round t of the first phase, such that $1 \leq t \leq r+1$, if d was larger than the number of neighbors of each node x for every round $t' < t$, then $||\mathbf{\Phi}_t||_1 = (n-\ell)\ell$.

▷ Claim 9. Given an ADN of n nodes running MMC, for any round t of any phase and any node x, it is $0 \leq \Phi_t[x] \leq \ell$.

To show that if $k^{1+\epsilon} < n$ MMC detects that the estimate is low, we focus on the first phase. We define a threshold τ and a number of rounds such that, after the first phase is completed, some nodes will have potential above τ and this can happen only if the estimate is low. Then we show that black nodes receive an alarm indicating that.

First, we show an upper bound of at most $k^{1+\epsilon}$ nodes with potential at most τ at the end of the first phase (Lemma 10). Thus, given that $k^{1+\epsilon} < n$, we know that there is at least one node with potential above τ at the end of the first phase. Second, we show that if the estimate is not low, that is $k \geq n$, then all nodes have potential at most τ at the end of the first phase (Lemma 11). That is, a potential above τ can only happen when indeed the estimate is low. Finally, we show that if $k^{1+\epsilon} < n$ an alarm "low" initiated by nodes with potential above τ must be received after $k^{1+\epsilon}$ further rounds of communication (Lemma 12).

▶ **Lemma 10.** *For $\epsilon > 0$, after running the first phase of the MMC protocol, there are at most $k^{1+\epsilon}$ nodes that have potential at most $\tau = \ell(1 - \ell/k^{1+\epsilon})$.*

▶ **Lemma 11.** *If $k \geq n$, $r \geq (4 + 2\epsilon - 2\ln(k^{\epsilon} - 1)/\ln k)dk^2 \ln k$, and $\epsilon > 0$, given that $k > \ell \geq 1$, at the end of the first phase no individual node should have potential larger than $\tau = \ell(1 - \ell/k^{1+\epsilon})$.*

The previous lemma shows that, if the estimate is "not-low" ($k \geq n$), at the end of the first phase all nodes must have "low" potential ($\Phi_{1,r+1} \leq \tau$). (Notice the inverse relation between estimate and potential.) In the following lemma we show that if $k^{1+\epsilon} < n$ (i.e. low estimate) there are some nodes with $\Phi_{1,r+1} > \tau$ (i.e. high potential), and that all the other nodes will know this within the following phase.

▶ **Lemma 12.** *If $k^{1+\epsilon} < n$, $0 < \epsilon \leq 1 + \log_k 4d$, and $r \geq (4 + 2\epsilon - 2\ln(k^{\epsilon} - 1)/\ln k)dk^2 \ln k$, within the following $k^{1+\epsilon}$ rounds after the first phase of the MMC protocol, all black nodes have received an alarm status "low".*

Finally, in the following lemma we show that if $k > n$, black nodes detect that the potential consumed is too low for the estimate k to be correct.

▶ **Lemma 13.** *Under the following conditions $d > k > n > \ell > 0$, for $\beta \geq \log_k(n(2k^{\delta} - 1))$, $\delta > \log_k(k^{\gamma}(n - \ell)/(k^{\gamma} - (n - \ell) - 1))$, and $\gamma > \log_k(n - \ell + 1)$, after running the MMC protocol for p phases and r rounds such that $p \leq 2\delta \ln k(1 - \ell\left(\frac{1}{n} - \frac{1}{k^{\beta}}\right))/(\ell\left(\frac{1}{n} - \frac{1}{k^{\beta}}\right))$, and $r \geq 2\beta dk^2 \ln k$, the potential ρ consumed by any black node is $\rho < (k - \ell)\left(1 - \frac{1}{k^{\gamma}}\right)$.*

Based on the above lemmata, we establish the correctness and running time of MMC:

▶ **Theorem 14.** *Given an ADN with n nodes, which include ℓ black nodes such that $n > \ell \geq 1$ black nodes, after running MMC for each estimate $k = \ell+1, \ell+2, \ell+3, \ldots, n$ with parameters: $d = k^{1+\epsilon}$, $p = \left\lceil \frac{2\ln k}{\ell} \max\left\{\frac{\gamma}{1/k+1/k^{\alpha}}, \frac{\delta}{1/d+1/k^{\beta}}\right\}\right\rceil$,
$r = \left\lceil 2dk^2(\ln k)\max\left\{\alpha, \beta k^{2\epsilon}, 2 + \epsilon - \frac{\ln(k^{\epsilon}-1)}{\ln k}\right\}\right\rceil$, and $\tau = \ell\left(1 - \frac{\ell}{k^{1+\epsilon}}\right)$, under the following conditions: $\epsilon > 0$, $\alpha \geq 1 + \gamma + \log_k 3$, $\beta \geq \log_k(d(2k^{\delta} + 1))$, $\gamma > \log_k(d - 1)$, $\delta > \log_k \frac{dk^{\gamma}}{k^{\gamma}+1-d}$. Then, all nodes stop after at most $\sum_{k \in E \cup B}(pr + d)$ rounds of communication and output n, for $E = \{2^i(\ell + 1) : i = 0, 1, \ldots, \log\lceil n/(\ell + 1)\rceil\}$, and $B = \{(2^{\log\lceil n/(\ell+1)\rceil} - 2^i)(\ell + 1) : i = 0, 1, \ldots, \log\lceil n/(\ell + 1)\rceil - 2\}$.*

▶ **Corollary 15.** *The time complexity of MMC on an ADN with ℓ black nodes and $n - \ell$ white nodes is $O\left(\frac{n^{4+\epsilon}}{\ell}\log^3 n\right)$, for any $\epsilon > 0$.*

6 Leader-less Methodical Counting

The main idea of LLMC (cf. Algorithm 1) is to consider consecutive powers of 2 as values of K, and for each such K to select black nodes locally with probability corresponding to the inverse of K and run MMC with $\ell = 1$. If $K \geq n$ and there is indeed one black node, MMC guarantees that all nodes find the proper count n of nodes and stop. There are however two problems: how to recognize if $K \geq n$ and what to do when there is no black node or at lest two black nodes. Algorithm LLMC overcomes these two issues implicitly, by combining the executions of MMC for consecutive powers of 2 with two other techniques:

- introducing parallel threads and carefully counting the number of threads with no black nodes recorded (locally) and requiring that their ratio is bigger than half (this is to recognize whether K is close to n with sufficiently high probability),
- making use of the fact that having more than one black node in the execution of MMC for $\ell = 1$ (as a subroutine of LLMC) cannot return an estimate bigger than as if it was run with one black node; therefore, taking the maximum of returned estimates over threads could mitigate the potential problem of more than one black node.

The main control parameter used in LLMC is K, which is an upper bound for estimates considered in one execution of the While loop (which we call epoch K). We start from sufficiently large K, to assure that starting from this value of K the chance of getting incorrect output or behavior of nodes (e.g., stopping at different times) is smaller than $\epsilon/2$ Within one While loop, we initiate $f(K)$ parallel threads and for each of them we select black nodes for the whole thread – trying to make sure that the chance of getting one black node is sufficiently large (we need it to argue about correct stopping), especially for $K \geq n$ (recall that we could not recognize for sure whether $K \geq n$ or not). Then in each thread independently we run MMC for $\ell = 1$ as a subroutine (hoping that we selected exactly one black node in the beginning of the loop), which checks all possible values of k from 1 to K to find a good estimate of n (i.e., we have to trim the execution of MMC to estimates $k \leq K$).

This approach does not work for $K < n$, as then the probability of getting more than one black node could be bigger than the one for one black node; however, we could eliminate such cases by monitoring the number of threads with no black node, which in case of $K < n$ should be compared with a large threshold (intuitively, we want LLMC to reach this threshold with high probability when K will be close to n). Note that a no-black-node thread will not output any estimate. If a thread identifies a good estimate, it puts it to the set *Count* and we do not enter the next iteration of While loop but stop, returning the maximum value in *Count* - this is to make preference to threads with one black node over those with more than one black node (we will argue that they could return values but not bigger than ones by threads with one black node).

Algorithm 1 LLMC algorithm. $\epsilon \in (0, 1)$.

```
 1: procedure
 2:     $K \leftarrow \lceil\lceil 12/(\epsilon) \rceil\rceil$        // $\lceil\lceil x \rceil\rceil$: the smallest power of 2 bigger than $x$
 3:     $Count \leftarrow \emptyset$    // set of potentially "good" estimates computed in threads
 4:     $EmptyThreads \leftarrow 0$        // number of threads with no black node detected
 5:     while $Count = \emptyset$ or $EmptyThreads \leq f(K)/2$ do
 6:         $Count \leftarrow \emptyset$, $EmptyThreads \leftarrow 0$
 7:         $K \leftarrow 2K$
 8:         Initiate $f(K) = 64\frac{\log(K/\epsilon)}{\log(e/(e-2))}$ parallel threads
              // parallel computation and messages sharing same resources/medium
 9:         for each thread do
10:             for each node do
11:                 Select to be a black node with probability $1/g(K)$, where $g(K) = K/2$
12:             end for
13:             $k \leftarrow MMC(K, 1)$                             // refer to Algorithm 2
14:             if $k > 0$ then
15:                 $Count \leftarrow Count \cup \{k\}$
16:             end if
17:             if no black node detected then
18:                 Increase $EmptyThreads$ by 1
19:             end if
20:         end for
21:     end while
22:     return max($Count$)  // Output the maximum number in $Count$ as the size $n$.
23: end procedure
```

Algorithm 2 Subroutine of LLMC.

```
    Input: number of black nodes $\ell$, max size estimate $K$.
 1: procedure MMC ($K$,$\ell$)
 2:     Run MMC modified as follows:
 3:         – Stop iterations when size estimate $k > K$
 4:         – If estimate $k < K$, remain idle until end of phase $K$ // for synchronization
 5:         – Include a Boolean $p_j$ in each node $j$ as follows:
 6:             – Initially:
 7:                 if node $j$ is black then $p_j \leftarrow true$ else $p_j \leftarrow false$
 8:             – In each iteration:
 9:                 Broadcast and Receive messages including $p_j$
10:                 if $p_i = true$ received from some neighbor $i$ then $p_j \leftarrow true$
11:     Upon completion:
12:         if $status = done$ return $k$ else return 0
13: end procedure
```

6.1 Analysis of Leader-less Methodical Counting

We may assume that $n > 1$, otherwise the algorithm would easily recognize $n = 1$ by no received communication. Throughout the whole analysis we consider an adaptive adversary, as we allow network changes to be done online when viewing the whole history of the computation up to the current round.

We call an execution of the While loop for a fixed parameter K *epoch* K. Recall that K is a power of 2 bigger than $12/(\epsilon)$; by $\lceil\lceil x \rceil\rceil$ we denote the smallest power of 2 bigger than x.

Observe from the structure of the algorithm that nodes could only stop at the end of an epoch. In the analysis below, we will often prove some properties under condition that the stopping times are synchronized, i.e., either all nodes stop or none, and remove this assumption at the end of the analysis. That is, we will show that in fact all nodes synchronize their stopping time and output the correct value of n with desired probability at least $1 - \epsilon$. We also conjecture that all arbitrary constants in the algorithm, i.e., in the definition of starting value of K, functions $f(K)$ and $g(K)$, could be substantially lowered, as we set them high to avoid too many cases in the analysis (so making it focused on main arguments).

The proofs of the following lemmata are left to the full version of this paper for brevity.

▶ **Lemma 16.** *The probability that for some epoch K, where $K < n$, the value of EmptyThreads is bigger than $f(K)/2$ at any node is smaller than $1 - \epsilon/2$. Consequently, with probability at least $1 - \epsilon/2$: no node stops during epochs $K < n$.*

Recall from the description of the algorithm that nodes could only stop at the end of an epoch. We prove the following two structural properties of LLMC.

▶ **Proposition 17.** *Consider an epoch $K \geq n$ and assume that all nodes are active in the beginning of this epoch. If some nodes stop while some other do not at the end of epoch K, then there are more than $f(K)/2$ threads with no black node selected and no thread with exactly one black node.*

▶ **Proposition 18.** *Consider an epoch $K \geq n$ and assume that all nodes are active in the beginning of this epoch. If all nodes stop but some of them output incorrect value, then there are more than $f(K)/2$ threads with no black node and no thread with exactly one black node.*

We use these structural properties to estimate the probabilities of the two following events.

▶ **Lemma 19.** *Consider an epoch $K \geq n$ and assume that all nodes are active in the beginning of this epoch. The probability of event: in epoch K there will be some nodes that stop and some other that do not, or all nodes stop with incorrect output, is at most ϵ/K.*

▶ **Lemma 20.** *The probability of some nodes stopping at different times or outputting incorrect value is at most $\epsilon/2$.*

We now analyze good events of correct stopping by all nodes, moreover, simultaneously. We start from stating a structural property and follow with estimating of the probability of correct simultaneous stopping.

▶ **Proposition 21.** *For any epoch $K \geq n$, if all nodes are active in the beginning, the number of threads with no black node is bigger than $f(K)/2$ and there is at least one thread with one black node, then all threads with one black node store the same value in set Count and it is the biggest value stored in Count in this epoch.*

▶ **Lemma 22.** *Consider an epoch $K \geq n$ and assume that all nodes are active in the beginning of this epoch. The probability of event: in epoch K all nodes stop with correct value, is at least $1 - \exp(-f(K)/64) - \exp\left(-\frac{nf(K)}{g(K)e}\right)$.*

▶ **Corollary 23.** *Consider epoch $K = \lceil\lceil 8n \rceil\rceil$ and assume that all nodes are active in the beginning of this epoch. The probability of event: in epoch K all nodes stop with correct value, is at least $1 - \epsilon/n$.*

▶ **Theorem 24.** *For any given $\epsilon > 0$, LLMC simultaneously stops at all nodes returning correct count n of nodes in $O(n^4 \log^4 n)$ rounds, with probability at least $1 - \epsilon$, even when running against an adaptive adversary.*

7 Conclusions

This paper expanded the knowledge about feasibility of polynomial computations in anonymous dynamic environments/networks. In particular, counting-type computations can be done deterministically if symmetry is broken by existing a partition of nodes where the size of one subset is known. It is also feasible without any distinction between the nodes with an arbitrary probability. Natural open directions include a study of the message complexity of MMC and LLMC, randomized approximation solutions and extensions to other related models and problems, as well as deriving tighter upper and lower bounds in the considered setting.

References

1 Chen Avin, Michal Koucký, and Zvi Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Automata, languages and programming*, pages 121–132. Springer, 2008.

2 Roberto Baldoni and Giuseppe A Di Luna. Counting on Anonymous Dynamic Networks: Bounds and Algorithms. manuscript, 2016.

3 Siddhartha Banerjee, Aditya Gopalan, Abhik Kumar Das, and Sanjay Shakkottai. Epidemic spreading with external agents. *IEEE Transactions on Information Theory*, 60(7):4125–4138, 2014.

4 Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

5 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

6 Maitri Chakraborty, Alessia Milani, and Miguel A. Mosteiro. Counting in Practical Anonymous Dynamic Networks is Polynomial. In *Proceedings of the 4th International Conference on Networked Systems*, volume 9944 of *Lecture Notes in Computer Science*, pages 131–136, 2016.

7 Maitri Chakraborty, Alessia Milani, and Miguel A. Mosteiro. A Faster Exact-Counting Protocol for Anonymous Dynamic Networks. *Algorithmica*, 80(11):3023–3049, 2018. `doi:10.1007/s00453-017-0367-4`.

8 Yuxin Chen, Sanjay Shakkottai, and Jeffrey G Andrews. On the role of mobility for multimessage gossip. *IEEE Transactions on Information Theory*, 59(6):3953–3970, 2013.

9 Giuseppe Antonio Di Luna and Roberto Baldoni. Investigating the Cost of Anonymity on Dynamic Networks. *CoRR*, abs/1505.03509, 2015. `arXiv:1505.03509`.

10 Giuseppe Antonio Di Luna and Roberto Baldoni. Non Trivial Computations in Anonymous Dynamic Networks. In *Proceedings of the 19th International Conference on Principles of Distributed Systems*, Leibniz International Proceedings in Informatics, 2015. To appear.

11 Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Conscious and Unconscious Counting on Anonymous Dynamic Networks. In *Proceedings of the 15th International Conference on Distributed Computing and Networking*, volume 8314 of *Lecture Notes in Computer Science*, pages 257–271. Springer Berlin Heidelberg, 2014.

12 Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Counting in Anonymous Dynamic Networks under Worst-Case Adversary. In *Proceedings of the 34th International Conference on Distributed Computing Systems*, pages 338–347. IEEE, 2014.

13 Giuseppe Antonio Di Luna, Silvia Bonomi, Ioannis Chatzigiannakis, and Roberto Baldoni. Counting in Anonymous Dynamic Networks: An Experimental Perspective. In *Proceedings of the 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, volume 8243 of *Lecture Notes in Computer Science*, pages 139–154. Springer Berlin Heidelberg, 2014.

14 M. Farach-Colton, A. Fernández Anta, A. Milani, M. A. Mosteiro, and S. Zaks. Opportunistic Information Dissemination in Mobile Ad-hoc Networks: adaptiveness vs. obliviousness and randomization vs. determinism. In *Proc. of the 10th Latin American Theoretical Informatics Symposium*, volume 7256 of *Lecture Notes in Computer Science*, pages 303–314. Springer-Verlag, Berlin, 2012.

15 Antonio Fernández Anta, Alessia Milani, Miguel A. Mosteiro, and Shmuel Zaks. Opportunistic information dissemination in mobile ad-hoc networks: the profit of global synchrony. *Distributed Computing*, 25(4):279–296, 2012. `doi:10.1007/s00446-012-0165-9`.

16 Dariusz R. Kowalski and Miguel A. Mosteiro. Polynomial Counting in Anonymous Dynamic Networks with Applications to Anonymous Dynamic Algebraic Computations. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 156:1–156:14, 2018. `doi:10.4230/LIPIcs.ICALP.2018.156`.

17 Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed Computation in Dynamic Networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 513–522. ACM, 2010.

18 Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. Naming and counting in anonymous unknown dynamic networks. In *Stabilization, Safety, and Security of Distributed Systems*, pages 281–295. Springer, 2013.

19 Alessia Milani and Miguel A. Mosteiro. A Faster Counting Protocol for Anonymous Dynamic Networks. In *Proceedings of the 19th International Conference on Principles of Distributed Systems*, volume 46 of *Leibniz International Proceedings in Informatics*, pages 1–13, 2015.

20 Damon Mosk-Aoyama and Devavrat Shah. Fast distributed algorithms for computing separable functions. *IEEE Transactions on Information Theory*, 54(7):2997–3007, 2008.

21 Angelia Nedic, Alex Olshevsky, Asuman Ozdaglar, and John N Tsitsiklis. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

22 Sujay Sanghavi, Bruce Hajek, and Laurent Massoulié. Gossiping with multiple messages. *IEEE Transactions on Information Theory*, 53(12):4640–4654, 2007.

23 Atish Das Sarma, Anisur Rahaman Molla, and Gopal Pandurangan. Distributed computation in dynamic networks via random walks. *Theoretical Computer Science*, 581:45–66, 2015.