Counting in Anonymous Dynamic Networks: An Experimental Perspective

G. A. Di Luna[†], S. Bonomi[†], I. Chatzigiannakis[‡], R. Baldoni[†]

[†] Dipartimento di Ingegneria Informatica, Automatica e Gestionale Antonio Ruberti
Universitá degli Studi di Roma La Sapienza
Via Ariosto, 25
I-00185 Roma, Italy
{baldoni, bonomi, diluna}@dis.uniroma1.it
[‡] Computer Technology Institute & Press "Diophantus" (CTI)
Patras, Greece
ichatz@cti.gr

Abstract. Counting is a fundamental problem of every distributed system as it represents a basic building block to implement high level abstractions. In anonymous dynamic networks, counting is far from being trivial as nodes have no identity and the knowledge about the network is limited to the local perception of the process itself. Moreover, nodes have to cope with continuous changes of the topology imposed by an external adversary. A relevant example of such kind of networks is represented by wireless sensor networks characterized by the dynamicity of the communication links due to possible collisions or to the presence of duty-cycles aimed at battery preservation. In a companion paper [14], two leader-based algorithms, namely $\mathcal{A}_{NoK}$ and $\mathcal{A}_{LCO}$, to count the number of processes in an anonymous dynamic network have been proposed. Such algorithms employ the notion of energy transfer to count the exact number of nodes by (i) having no knowledge on the network or (ii) having access to a *local counting oracle* reporting the exact number of neighbors at the beginning of a communication round. Let us notice that, while $\mathcal{A}_{LCO}$ has a well defined terminating condition, $\mathcal{A}_{NoK}$ only ensures that eventually the leader is able to count the exact number of processes but it is not able to identify when this happens. In this paper, we define a new algorithm $\mathcal{A}^*_{NoK}$ by augmenting $\mathcal{A}_{NoK}$ with a termination heuristics that allows the leader to guess when it should output the current count and we provide an experimental evaluation on different types of dynamic graphs for both $\mathcal{A}_{NoK}$ and $\mathcal{A}^*_{NoK}$. In addition, we also extended $\mathcal{A}_{LCO}$ by defining a new algorithm, namely $\mathcal{A}^*_{LCO}$, that is the basic $\mathcal{A}_{LCO}$ augmented with a symmetry breaking condition that helps to speed up the convergence time.

1 Introduction

Networks of tiny artifacts will play a fundamental role in the computational environments and applications of tomorrow. Networked embedded sensors and mobile devices will produce a constant flow of data between the real world and modern and traditional networks such as information, communication and social networks. Such hyperconnected dynamic environments create very challenging system models where what was trivially solvable in a static system, is now far from being trivial. What is becoming apparent is that in such environments, theory and models for static distributed systems do not capture anymore the new kind of applications that are emerging. As a result, over the last years, dynamic distributed systems have attracted a lot of interest from the relevant research community (see e.g., [7,8,23]).

A critical issue in designing such hyperconnected dynamic infrastructures is security and trust, especially when artifacts exchange crucial information that needs to be protected [10]. It is evident that contemporary networks have significant difficulties dealing with third-party tracking and monitoring online, much of it spurred by data aggregation, profiling, and selective targeting. Terms like information security, data confidentiality and integrity, entity authentication and identification need to be considered [25]. A promising approach for addressing these problems is to incorporate privacy in the design and models of such future systems by guaranteeing the *anonymity* of the artifacts.

In this paper, we consider the problem of counting the number of nodes in a network without revealing any information on the identity of nodes or providing information about the network state. Counting is among the most fundamental problems of distributed computation and it is a key function for network management and control, and the vast number of papers appearing in the relevant literature is a clear indication of its importance. A large part of these studies deals with causes of dynamicity such as failures and changes in the topology that eventually stabilize [15]. However, the low rate of topological changes that is usually assumed is unsuitable for reasoning about truly dynamic networks. We envision future networks with highly dynamic changes: connected artifacts may become immediately unreachable after they have been received a message from them. We consider recent theoretical models for dynamic networks in which the topology may change arbitrarily from round to round. In some models (e.g.,[17]), edges - representing communication among hyperconnected artifacts - are changed at each round by an adversary, that is constricted to modify edges in such a way that the network is always connected. In other models (e.g., [12,5]), edges appear by following a random distribution where certain properties of the dynamic network hold with high probability. Under these assumptions taken for granted, theoretical results indicate that we can design protocols for distributed tasks that are robust, scalable and that terminate.

In this work we remove fundamental assumptions made by previous theoretical models: (a) we avoid any assumption on the network knowledge: we look into cases where nodes do not know the size (or an upper bound) n of the network, or any other metric; (b) we also avoid any assumption on providing unique identities (ids) to the artifacts: nodes execute identical programs and in symmetric networks it is impossible to count the nodes unless a leader is not introduced; (c) we do not require the network to be connected at each time instance, or connected with high probability. We believe that the resulting mode of operation is more suitable for future hyperconnected environments, where privacy is incorporated in the model. Under this mode of operation, we propose a new distributed algorithm, namely $\mathcal{A}^*_{NoK}$, that employs a termination heuristic in order to provide estimates on the size of the anonymous network. $\mathcal{A}^*_{NoK}$ builds upon the no-knowledge algorithm (i.e. $\mathcal{A}_{NoK}$) introduced in a companion paper [14]. Both of them exploits an energy-transfer technique to count the exact number of nodes.

We follow a detailed experimental approach and investigate the performance of both $\mathcal{A}_{NoK}$ and $\mathcal{A}^*_{NoK}$ (Sections 4.1 and 4.2). We also consider an algorithm, namely $\mathcal{A}_{LCO}$, studied in [14] in order to compare its performance with $\mathcal{A}^*_{NoK}$. To do so we suitably modify it into a new algorithm $\mathcal{A}^*_{LCO}$ in order to operate under the new, more generic, mode of operation (Section 4.5). We look into different random evolving graph models in order to identify the error rate of the algorithm as well as the efficiency for terminating the computation. We also look into networks that are periodically disconnected as the artifacts duty-cycle (Section 4.4).

For the case of densely connected anonymous networks, $\mathcal{A}^*_{NoK}$ terminates always correctly. In cases where the network experiences regular partitions, $\mathcal{A}^*_{NoK}$ provides estimates on the size whose accuracy varies according to the degree of disconnection of the network (see Section 4.2). Longer periods of network disconnections bring to lower accuracy in counting. Let us finally remark that $\mathcal{A}^*_{NoK}$ is able to answer to predicates such as "*does the network contain more than T nodes?*" (i.e., $|V| \geq T$)in a number of rounds lesser than the one needed by the base $\mathcal{A}_{NoK}$ algorithm presented in [14] as shown in Section 4.3.

2 Preliminaries

System Model. A *dynamic network* is a network whose topology changes along time due to possible failures of nodes or communication links. We consider computations executed in discrete *synchronous* rounds, controlled by a fictional global clock accessible to all the nodes. Thus, all nodes have access to the current round number via a local variable that we usually denote by r. A dynamic network is modeled by a *dynamic graph* $G(r) = (V, E(r))$, where V is a set of n nodes (or processors) and $E : \mathbb{N} \to \mathcal{P}(E')$, where $E' = \{\{u, v\} : u, v \in V\}$, is a function mapping a round number $r \in \mathbb{N}$ to a set $E(r)$ of bidirectional links drawn from E' [17]. Intuitively, a dynamic graph G is an infinite sequence $G(1), G(2), \ldots$ of *instantaneous graphs*, whose edge sets are subsets of E' chosen by a *worst-case adversary*. The set V is assumed throughout this work to be *static*, that is it remains the same throughout the execution.

Nodes in V are *anonymous*, i.e. they have no identifier. At each round r, the local view of a node v, denoted as $l_v(r)$, is defined by the multi set containing all the states of processes that are neighbors of v at round r (i.e. all the local variables maintained by the neighbors of v at round r).

Nodes in the network communicate by sending and receiving messages via *anonymous broadcast*; in every round r, each node u generates a single message $m_u(r)$ to be delivered to all its current neighbors in $N_u(r) = \{v \mid \{u, v\} \in E(r)\}$.

Dynamic Graph Models. In order to model the dynamicity of the topology graph, we consider the following four models:

1. **$\mathbf{G(n, p)}$ graph [11]**: at the beginning of each round r the set of edges is emptied and then for any pair of processes $u, v \in V$, the edge uv is created according to a given probability p. Let us recall that in the $G(n, p)$ graph model, there exist a connectivity threshold t, depending on the number of nodes n, such that if probability p is above the threshold, then $G(n, p)$ is connected with very high probability.
2. **Edge-Markovian (EM) graph [12]** : at each round r, edges are modified according to the following rules:
   (a) For each edge $uv \in E(r-1)$, uv is removed from $E(r)$ with a probability p_d (i.e., *death probability*).
   (b) For each edge $uv \notin E(r-1)$, uv is created and inserted in $E(r)$ with a probability p_b (i.e., *birth probability*).

   Clearly, connectivity of the graph at each round depends on p_d and p_b.
3. **Random Connected graph**: at each round we sample a pair of nodes $(u, v) \in V$, and we create an edge obtaining the graph $G(V, E')$, we iterate the procedure until we obtain a connected graph. This model constructs evolving graphs that are the sparsest possible, still guaranteeing connectivity.
4. **Duty-cycle based graph**: at round r_0 the dynamic graph has a fixed, connected, topology. Each node follows a duty cycling phase during which, if at a given round r_i the node is awake it can receive and send messages according the topology of r_0 to any neighboring node that is also awake. While when at round r_j it is in sleep mode, all adjacent edges are removed from the graph. The presence of the duty cycle essentially brings some dynamicity in the graph since not all edges will be set at each round. This model constructs evolving graphs that reflect realistic deployments of resource constraint devices. Remark that this model does not guarantee that the graph will be connected at each round.

Energy-Transfer Technique. In [14], a new technique for counting the size of the network is introduced that overcomes the lack of identities and the constantly dynamic environment by a new and surprisingly simple concept of *energy-transfer*. Each node is assigned a fixed energy charge, and during each round it discharges itself by disseminating it around to its neighbors. The leader acts as a sink collecting energy (i.e., energy is not transferred by the leader to neighbors). The technique enforces, at each round, an invariant on the sum of energy among networks' nodes: energy is not created or destroyed. Considering the behavior of the nodes, the energy is eventually transferred to the leader and stored there. The leader measures the energy received to count the size of the network. Interestingly, this technique is very simple to implement and depends on very limited information about the attributes of a given network. The paper introduced a series of algorithms that apply the *energy-transfer technique* that either assume knowledge on certain aspects of the network (e.g., an upper bound on node degree) in order to terminate the computation (i.e. $\mathcal{A}_{LCO}$ algorithm), or do not make any additional assumption but do not terminate as the $\mathcal{A}_{NoK}$ algorithm (essentially the computation converges to the correct input, but the nodes are not able to detect when to terminate). The performance of the algorithms is rigorously studied and correctness and termination is formally proved.

We remark that the results in [14], especially those concerning the absence of any knowledge assumption, represent an interesting feasibility point, even if they cannot be used in practice since the leader is not able to verify any terminating condition and thus it is not able to provide an answer to the counting problem. In this paper, we present an algorithm, namely $\mathcal{A}^*_{NoK}$, obtained by the basic $\mathcal{A}_{NoK}$ one, in which we define a terminating condition based on the definition of an heuristic and we show that it enables an accurate count.

Related Work. The question concerning which problems can be solved by a distributed system when all processors use the same algorithm and start from the same state has a long story with its roots dating back to the seminal work of Angluin [3], who investigated the problem of establishing a "center". She was the first to realize the connection with the theory of graph coverings, which was going to provide, in particular with the work of Yamashita and Kameda [24], several characterizations for problems that are solvable under certain topological constraints. Other well-known studies on unknown networks have dealt with the problems of robot-exploration and map-drawing of an unknown graph [2,13,21] and on information dissemination [6]. Sakamoto [22] studied the "usefulness" of initial conditions for distributed algorithms (e.g. leader or knowing n) on anonymous networks by presenting a transformation algorithm from one initial condition to another. Fraigniaud *et al.* [16] assumed a unique leader in order to break symmetry and assign short labels as fast as possible. Recently, Chalopin *et al.* [9] have studied the problem of naming anonymous networks in the context of snapshot computation. Finally, Aspnes *et al.* [4] studied the relative powers of reliable anonymous distributed systems with different communication mechanisms: anonymous broadcast, read-write registers, or read-write registers plus additional shared-memory objects.

Distributed systems with worst-case dynamicity were first studied in [20] by introducing the 1-interval connectivity model. They studied flooding and routing problems in asynchronous communication and allowed nodes detect local neighborhood changes. Under the same model, [17] studied the problem of counting for networks where nodes have unique IDs and provided an algorithm that requires $O(n^2)$ rounds using $O(\log n)$ bits per message. In [18] studied the problem of anonymous counting in this worst-case dynamicity model and provided an algorithm where given that the nodes know an upper bound on the maximum degree that will ever appear, the nodes obtain an upper bound on the size of the network. In [19] the 1-interval connectivity assumption is replaced by other less restrictive *temporal connectivity* conditions that only require that *another causal influence occurs within every time-window of some given length*. They introduce several novel metrics for capturing the speed of information spreading in a dynamic network and provide terminating algorithms for fast propagation of information under continuous disconnectivity.

To the best of our knowledge, this is the first experimental study for distributed counting algorithms in anonymous dynamic networks that are possibly disconnected. We believe that our results provide strong evidence that efficient computation can be designed for such future networks.

3 Counting Algorithms for Anonymous Dynamic Networks

3.1 The No-Knowledge Algorithm $\mathcal{A}_{NoK}$

The No-Knowledge Algorithm ($\mathcal{A}_{NoK}$) presented in [14] works in the following way: each non-leader node v starts, at round r_0, with energy quantity $e_v = 1$ and it transfers half of its current energy to the neighbors.

However, v has no knowledge about the network and thus it cannot know the exact number of neighbors in r before receiving messages, but it can only guess such number. Thus, v supposes to have d neighbors and it broadcasts a quantity of energy $\frac{1}{2d}$ (as if there are really d neighbors). Then v starts to collect messages transmitted by its neighbors at the beginning of the round and it stores such messages in a local variable S_{msg}. At the end of the round, v updates its energy e_v to $\frac{1}{2} + (d - |S_{msg}|)\frac{1}{2d} + \sum_{\forall m \in S_{msg}} m$ to preserver the quantity of energy over all the network.

Notice that, if the real number of neighbors at round r is lower than the estimation (i.e., $|N_v(r)| \leq d$) then the global energy conserved among all the processes is still constant (this is due to the compensation done by v at the end of the round based on the effective number of received messages). On the contrary, if the number of neighbors is greater than the estimation (i.e., $|N_v(r)| > d$) then, there is the release of a *local surplus* of energy. As an example, consider the case where v has energy e_v the estimation of neighbors is $d = 2$ and the real number of neighbors is $N_v(r) = 8$. When v sends $\frac{e_v}{4}$ to each neighbors, the total amount of energy transferred is twice the energy stored by v (i.e., the energy transferred is $8 \times \frac{e_v}{4} = 2e_v$ while node v had only e_v residual energy). However, since v adjusts its local residual energy considering the number of received messages, it follows that its residual energy will become negative and globally the energy is still preserved.

The local surplus of positive/negative energy could create, in the leader, a temporary value of energy e that is greater than $|V|$ or negative. Moreover, the adversary could change, at each round, the degree of nodes in order to avoid the convergence of the leader. To overcome these issues each processes stores locally the highest number of neighbors it has ever seen and it uses such number as estimation of its degree d. In this way the surplus of local negative/positive energy that the adversary can create is upper bounded by a function $f(|V|)$: each node v can increase d at most $|V| - 1$ times, from 1 to $|V|$. This implies that worst case adversary cannot create an infinite surplus of local energy. Since the conservation of energy is not violated and the local surplus of energy is finite, it is straightforward to prove that the leader has to converge to the value $|V|$ and the adversary could delay this convergence only a finite number of times. Intuitively, the adversary cannot delay too much its moves, because when the energy stored in $V \setminus \{v_l\}$ is less than a certain value, the local surplus of energy that it could create, even in worst case, it is not enough to change the leader count. So, if at each round r the leader counts $\lceil e_{v_l} \rceil$, it is possible to prove that there exists a round r^* after which the leader will always count the correct value despite the move of the adversary [14].

Unfortunately, looking to the number of consecutive rounds in which the leader outputs always the same count is not sufficient to provide a terminating condition as such number can always be influenced by the adversary. As a consequence, the leader cannot detect convergence. In fact, let us suppose that the leader stops when the increment of energy at round r is below a threshold t. It is always possible to have a network of size $t + 1$ where each node have a residual charge of t. So each increment on the leader energy is below the termination threshold but the residual energy on the network is greater than 1, so if the leader terminates it will miss one node.

3.2 The The No-Knowledge Algorithm with Termination Heuristic $\mathcal{A}^*_{NoK}$

In this section, we will present the heuristic added to the basic $\mathcal{A}_{NoK}$ to obtain the new algorithm $\mathcal{A}^*_{NoK}$ working in an anonymous network with No Knowledge assumption and having a termination condition. The heuristic is used by the leader to decide at which time the current count can be considered as the final one. The heuristic is based on the assumption that the dynamicity of the graph is governed by a random process (i.e., a graph where links change according to a uniform probability distribution) and it considers the notion of *flow observed by the leader*.

At each round r, the leader v_l will receive a fraction of energy from all its neighbors. So the flow of energy to the leader at round r can be expressed as:

$$\varPhi^r(v_l) = \sum_{\forall v \in N_{v_l}(r)} \frac{e_v(r)}{2d_v^{max}(r)}$$

where $e_v(r)$ is the energy of v at round r and $d_v^{max}(r)$ is the maximum number of neighbors that node v has so far. After a sufficient number of rounds, the estimation of the flow observed by the leader is

$$\varPhi^r(v_l) = \sum_{\forall v \in N_{v_l}(r)} \frac{e_v(r)}{2d_v^{max}(r)} \simeq \frac{|N_{v_l}(r)|}{2d_{avg}^{max(r)}} \overline{e_v(r)}$$

where $2d_{avg}^{max}(r)$ is the average of the maximum degrees seen by nodes in G at round r and $\overline{e_v(r)}$ is the average of the energy kept by all non-leader nodes at round r.
Let us remark that, in the absence of the leader, the energy is always balanced among nodes in the network and let us recall that the leader is the only node absorbing energy. As a consequence, nodes being neighbors of the leader could have less energy than others as they transferred part of their energy to the leader without receiving nothing from it. Due to the assumption about the probabilistic nature of the edges creation process and considering the functioning of $\mathcal{A}_{NoK}$, those non-leader nodes will tend to have a similar quantity of energy as they will balance energy surplus. Thus, the leader can estimate $\overline{e_v(r)} \simeq \frac{|V|-e_{v_l}(r-1)}{|V|}$.

Due to the assumption about the probabilistic nature of the edges creation process, the leader will see almost the same maximum number of neighbors as the other nodes. Thus, $2d_{v_l}^{max}(r) \simeq 2d_{avg}^{max}(r)$. Thus, substituting we have

$$\varPhi^r(v_l) \simeq \frac{|N_{v_l}(r)|}{2d_{v_l}^{max}(r)} \frac{|V| - e_{v_l}(r-1)}{|V|}$$

from which we obtain

$$|\bar{V}(r)| \simeq \frac{\rho(r) e_{v_l}(r-1)}{\rho(r) - \varPhi^r(v_l)}$$

where $|\bar{V}(r)|$ is estimation of the number of processes in the network done by the leader at round r and $\rho(r) = \frac{|N_{v_l}(r)|}{2d_{v_l}^{max}(r)}$.

Let $k = \lceil e_{v_l}(r) \rceil$ be the number representing the count done by the leader at round r, and let $\varDelta(r) = |\bar{V}(r)| - e_{v_l}(r)$ be difference between the network size estimated with the energy flow and the energy currently stored at the leader. We can finally define a termination condition as follows: as long as $\lceil e_{v_l}(r) \rceil$ remains stable, the leader computes the average $\overline{\varDelta}$ of $\varDelta(r)$ over the last k rounds and if after k consecutive rounds the quantity $\lceil e_{v_l}(r+k) + \overline{h} \rceil$ is equal to k and $\lceil e_{v_l}(r+k) \rceil = k$ the counting procedure terminates and the leader outputs k.

3.3 The Local Counting Oracle-based Algorithm $\mathcal{A}_{LCO}$

To overcome the issue of the terminating condition, in [14] the notion of a *local counting oracle* (LCO) is introduced that reports, at each round r, the current number of neighbors. Assuming the existence of such an oracle, a counting algorithm, namely $\mathcal{A}_{LCO}$, has been presented that is able to count the exact number of processes in a finite number of rounds. In this paper, we first propose a new algorithm called $\mathcal{A}^*_{LCO}$ and obtained trough a simple modification of $\mathcal{A}_{LCO}$ in order to let it work in a practical context and then we compared its performances with respect to $\mathcal{A}^*_{NoK}$.

The basic idea of $\mathcal{A}_{LCO}$ is to color every node in the graph and to count how many processes are colored with a specific color. The computation is done in synchronous round. At round r_0, the leader colors itself with color c_0 while each other node has no color, or simply its color is $\perp$. At any round r a new color c_r is used and each node with color $\perp$ can be colored with c_r if and only if it has at least one neighbor with a color different from $\perp$. In addition, in any round r, every non-leader colored process propagates 1 energy unit together with its color c_r, with the multiset containing the information it has about its neighbors and the current round r. When the counting algorithm starts at round r_0, the leader knows exactly how many processes are colored with color c_0 as they are its neighbors, they are colored by the leader itself and their number is provided by the LCO. For any following round, the leader initializes a local variable acting as container to collect the energy propagated by colored nodes; such energy is transferred to the leader by using the same mechanism of $\mathcal{A}_{NoK}$ and the access to LCO ensures that negative energy can not be created.

The leader starts collecting energy from nodes with color c_0 and it waits until it collects energy form all of them. When the leader collected all the energy from nodes with color c_0, it can compute, using the multi sets of local view gathered by nodes with color c_0, a bound B_1 on the set of nodes with color c_1. Such bound is given by the number of $\perp$ multiplied for the multiplicity of the multi set (multiplicity obtained using the energy). The bound B_1 is used by the leader to check when it has collected enough energy to obtain the correct count C_1 of process with color c_1.

Each node w with color c_1 at round 2 will create an unitary quantity of energy that will be transferred to the leader, this energy is marked with the local view $l_w(1)$, the color c_1 and the round 1.

Collecting this energy, the leader can compute, for each node w, the multiplicity of neighbors with id 0. Using this information it can lower the bound B_1 till it obtains the correct count C_1 of nodes with id 1, this condition is reached when the energy collected from nodes with color c_1 is equal to the adjusted bound B_1. The leader uses the sum $C_{\leq 1} = C_0 + C_1$ to obtain the multi sets of neighbors of nodes with ids $\{0, 1\}$, if this multi set is empty the leader terminates otherwise it uses the same procedure to obtain nodes with color c_2 and so on.

3.4 The Local Counting Oracle-based Algorithm with Symmetry Breaking $\mathcal{A}^*_{LCO}$

$\mathcal{A}_{LCO}$ in [14] is able to provide an exact count in a finite time. However, it may take a large amount of time since the leader may not be able to count two nodes with the same color until it collects more than one unit of energy from them. Let us assume for example that there are y nodes that at round r have the same color and the same multi set l_v of neighbors. The leader has to collect at least $y - 1 + \epsilon$ energy to count the correct multiciplity y of l_v, this process could be extremely slow. However, in practice such two nodes may be identified if we consider the history of their local views, i.e., the union of all the multi-sets they saw from round r_0 until the current round.

Based on this intuition, we defined a new algorithm, namely $\mathcal{A}^*_{LCO}$, by breaking some symmetry in $\mathcal{A}_{LCO}$. Symmetry breaking is achieved by introducing an additional parameter obtained considering all the local views history and it is used to break the symmetry and to disambiguate processes having the same color. Essentially, each non-leader process with color $c_i \neq \perp$ computes, at each round r, a round id $rid_r = rid_{r-1} + l_v(r-1)$; in this way, two nodes v, u that at round r' have a different multi set of neighbors will always have for each round $r > r'$ a different rid_r. Such rid_r will be attached together with the other information to the energy created at round r.

With this modification the algorithm logically uses the concept of energy to count when the symmetry has not been broken by the dynamic topology, i.e., two nodes that at each rounds have the same neighborhood. Otherwise it can count fast, let us suppose that at round r all nodes with $c_{id} \neq \perp$ have a different rid_r the leader could collect information from all necessary nodes in at most V rounds. Considering the previous example if $\frac{y}{2}$ nodes have different rid_r the leader have to wait till it collects $\frac{y}{2} - 1 + \epsilon$ of energy that is in general faster.

4 Performance Evaluation

Simulator. In order to run our experiments, we developed a JAVA simulator using the Jung library [1] to keep track of the graph data structure. Each process v is seen as a node in the graph and it exposes an interface composed of two methods: the first one allowing to send a message for round r and the second one allowing to deliver messages for the round r. Moreover, each node has associated a queue q_v storing the messages that it has to receive. The simulation is done trough a set of threads; a thread T_j takes a node from a list l_m containing all of nodes to be examined in this round, removes it from the list and invokes the method send message. T_j also takes the message m generated by v, and adds it to the queues of $N_v(r)$. When l_m is empty, a different set of threads is activated to deliver messages. T_j takes a node v from a list l_d and manage the delivery of all messages in q_v that v received during the current round. When all the messages in the queues are delivered to all the processes, the round terminates and the topology can be modified according to the dynamicity model considered and a new round can start.
Metrics and parameters. We investigate three key performance metrics:

- **Convergence Time Distribution:** the convergence time is defined by the first round at which the algorithm outputs the correct value. In the following, we studied the probability distribution of the convergence time to show the average latency of the algorithms before reaching a correct count.
- **Flow Based Gain $\boldsymbol{\Delta}$**: such metrics represents the difference measured by the leader between the size estimated through the flow and the the size estimated trough the energy stored inside the leader (i.e., $\Delta(r) = |\bar{V}(r)| - e_{v_l}(r)$).
- **Error frequency** ρ: we measured the percentage of uncorrect termination obtained while adopting the heuristics-based termination condition defined in Section 3.2.

The above metrics have been evaluated by varying the following parameters:

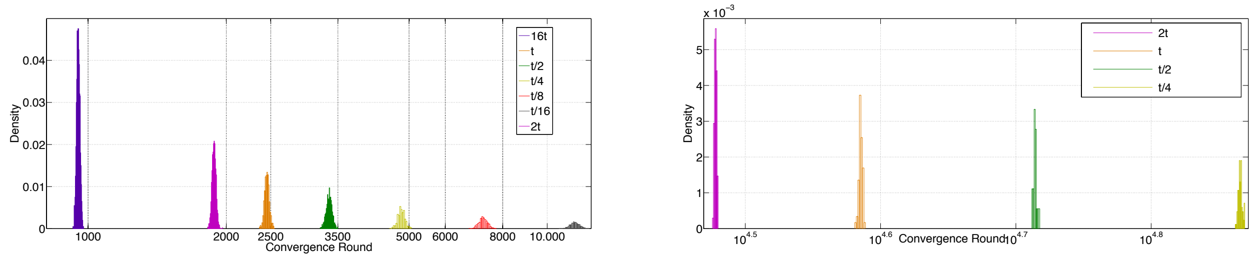
- **Dynamicity model:** we considered different types of dynamic graphs to evaluate the factors impacting every metrics (see Sec. 2 for a formal description).
- **Edges creation probability p:** such probability governs the graph dynamicity according to the specific model considered ($G_{(n,p)}$ or edge-Markovian).

We have evaluated the performance of the algorithms under different metrics in networks comprised of $\{10, 100, 1000\}$ nodes. When not explicitly stated, tests are the results of 1000 independent runs.

4.1 Evaluation of $\mathcal{A}_{NoK}$

Concerning the $\mathcal{A}_{NoK}$ algorithm, we implemented and tested it on both $G_{n,p}$, edge-Markovian and Duty-cycle-based graphs. Let us first consider the case of $G_{(n,p)}$ graphs and let us recall that the connectivity threshold t is defined according to the number of nodes in the graph (i.e., $t = \frac{ln(|V|)}{|V|}$). We evaluate our algorithm for several probability p. In particular, for any probability greater than $2t$, we consider only connected graph instances, i.e., at each step, we check the connectivity and in case of disconnected graph we sample a new random graph. For probabilities smaller than $2t$ we allow disconnected graph instances.

Figure 1 shows the convergence time distribution of the $\mathcal{A}_{NoK}$ algorithm running on $G_{(n,p)}$ graphs. As expected the convergence time becomes worse when we consider disconnected instances. However, it is



(a) $|V|$=100 (b) $|V|$=1000

Fig. 1: $\mathcal{A}_{NoK}$ Convergence time distribution for $G_{(n,p)}$ graphs with different edge creation probabilities p.

worth notice that the algorithm is able to converge to the correct count even in presence of disconnected instances. Moreover, the increment of convergence time is inversely proportional to p and there is an increment of the distribution variance due to the presence of disconnected instances.

When considering edge-Markovian graph, we set the probability of creating an edge as in the $G_{(n,p)}$ graphs and we fixed the probability of deleting an edge to 0.25 (i.e., $p_d = 0.25$ and $p_b = f(t)$).

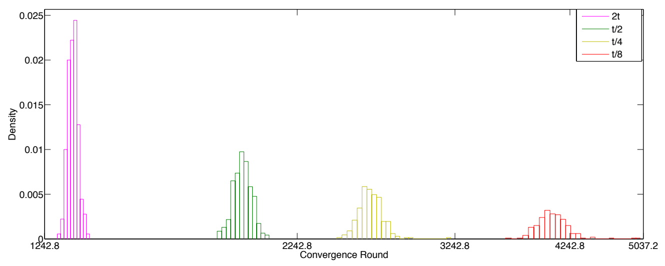


Fig. 2: $\mathcal{A}_{NoK}$ Convergence time distribution for edge-Markovian graphs with different edge creation probabilities p_b and $|V| = 100$.

Figure 2 shows the $\mathcal{A}_{NoK}$ convergence time distribution and we can see that it is comparable to the $G_{(n,p)}$ graph one. In addition, the persistence of edges across rounds (due to $p_d \leq 1$) mitigates the low values of edge creation probability. As a consequence, the convergence is faster than the pure $G_{(n,p)}$.

4.2 Evaluation of $\mathcal{A}^*_{NoK}$

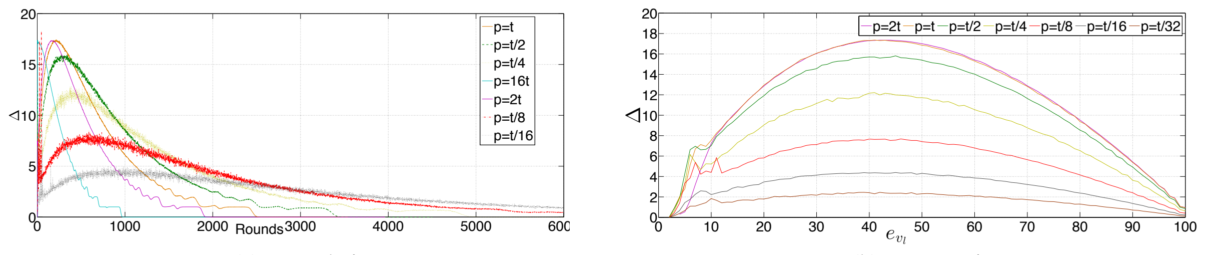
In the following tables we evaluate the $\mathcal{A}^*_{NoK}$ algorithm on both $G_{(n,p)}$ and Edge-Markovian graphs. In Figure 3 we show several measures related to the heuristic correctness. In particular, in addition to the error frequency ρ, we measured also the average error and maximum error done, by the heuristic, in terms of number of nodes missed with respect to the real number of nodes in the graph. We omit from the Figure some probabilities since they always terminate correctly ($p \geq \frac{t}{2}$ in case of $G_{(n,p)}$ graphs and$p_b \geq \frac{t}{4}$ for the Edge-Markovian). In case of disconnected topologies, i.e., $p \leq \frac{t}{4}$ for the $G_{(n,p)}$ or $p_b \leq \frac{t}{8}$ for the Edge-Markovian, we have that the percentage of counting instances terminating correctly is smaller that 100% and it becomes proportionally worse with the decrease of p. Moreover, it is possible to see a bimodal behavior of the heuristic when it fails: two cases are frequent in the experiments (i) the heuristic forces the termination in the first rounds of the counting process with the consequence of having the leader outputting a count much smaller than the real number of processes and (ii) the heuristic fails when the energy accumulated by the leader is close to the current network size. In all our experiments we have not found a case in which the heuristics forces the termination in a case different from this two. Moreover in the table we indicate the *Convergence Detection Time*, that is the number of rounds after the first convergence that the heuristics employs to correctly terminate the count. It is possible to see that in the majority of experiments, even on disconnected instances the heuristic converges in a time that is equal to the size of the network.

Model	$G_{(n,p)}$						Edge-Markovian $p_d = 0.25$		
p	$\frac{t}{4}$			$\frac{t}{8}$	$\frac{t}{16}$	$\frac{t}{32}$	$\frac{t}{8}$	$\frac{t}{16}$	$\frac{t}{32}$
$\|V\|$	10	100	1000	100	100	100	100	100	100
ρ	22%	3%	2%	19%	25%	84%	30%	68%	76%
Average Error	2,02	8,96	1	9	44,5	41,4	1	3,12	11,8
Max Error in Nodes	8	96	1	99	99	99	1	99	99
σ of Error	2,1166	27,4	0	27,4	48,3	48,8	1	14,23	29,73
Convergence Detection Time Average	10,2	100	1000	100	100	100	100	100	100
Convergence Detection Time Max	40	100	1000	100	100	100	100	100	100
Convergence Detection Time Min	10	100	1000	100	100	100	100	100	100

Fig. 3: Evaluation of the Termination Correctness ρ. The results are the outcome of 500 experiments

4.3 Comparison between $\mathcal{A}_{NoK}$ and $\mathcal{A}^*_{NoK}$

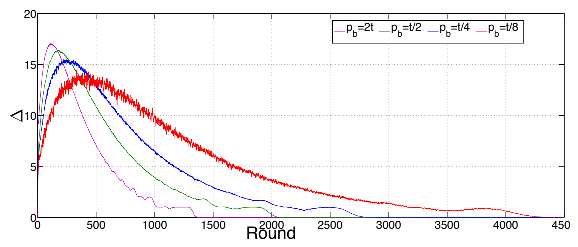
The flow could be used to estimate the size of $|V|$ obtaining a faster count. Figure 4 shows the evolution of Δ, i.e., difference measured by the leader between the size estimated through the flow and the the size estimated trough the energy stored inside the leader, both from a temporal perspective 4(a) and from the energy perspective 4(b).



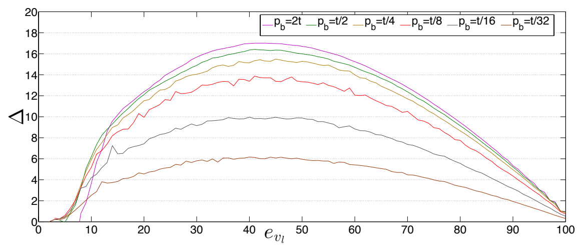
(a) Round view (b) Energy view

Fig. 4: Difference between the size estimated with the flow ($\mathcal{A}^*_{NoK}$) and the size estimated by looking to the energy stored at the leader ($\mathcal{A}_{NoK}$) in a $G_{n,p}$ network of $|V|$=100.

The value Δ reaches the maximum when the energy at the leader is approximately half of the network size; in this case, when the network is connected (i.e., $p \geq t$), the use of the heuristic allows the leader to predict, correctly, the presence of at least others 17 nodes.
So, on connected instances our approach could be useful to answer faster to predicates likes $|V| \geq t$. In addiction, the flow-based estimation continues to perform well on non-connected instances only until a certain threshold, then the gain obtained with the flow drops to one or two nodes more than the ones estimated by the energy.



(a) Round View



(b) Energy View

Fig. 5: Difference between the size estimated with the flow ($\mathcal{A}^*_{NoK}$) and the size estimated by looking to the energy stored at the leader ($\mathcal{A}_{NoK}$) for Edge Markovian network with $p_d = 0.25$ of $|V|$=100.

Moreover the figures show why the termination heuristics works bad on instances where $p \leq \frac{t}{4}$, we can see that $\varDelta$ falls behind the threshold of 1, both when the energy in the leader is low, and when the energy in the leader is approaching the value $|V|$ this could lead to two possible misbehavior, terminating after few rounds from the start, so with a value that could be sensibly distant from the value of $|V|$ or it could terminate near $|V|$, when $\varDelta$ falls again behind 1.

Figure 4(a) shows the behavior of $\varDelta$ along time. In particular,

– when the network is connected (i.e., $p \geq t$), the counting done by the leader fast approaches half of the network size (i.e., the maximum value for $\varDelta$). The energy-based count approaches the actual size with an exponential time; this is visible from the exponential decay of $\varDelta$. This behavior is present also when $p < t$, even tough there is a slower decay of $\varDelta$ that obviously reflects a slower approach to the actual size.
– for values of $p \leq t$ the curves show a high variance. This is due to the presence of disconnected topologies that introduce a variance in the convergence time for which the magnitude is proportional to the inverse of p. This high variance in convergence is due to the high variance of the flow that the leader will see during the execution.

The same behavior can be observed in Edge-Markovian graphs (cfr. Figure 5). The presence of more edges in the edge-markovian graph affects positively the $\varDelta$ measures since it is less prone to the value of p. It is possible to notice a slightly low maximum value for the edge-markovian process, 17 against 17.3 of the $G_{(n,p)}$ graph.

We run also tests with larger graphs ($|V| = 1000$) but we omit them here since curves exhibit the same behavior of those shown in Figures 4 and 5, notably in this case the maximum delta is about 170 nodes.

4.4 Duty Cycle

In order to test the adaptiveness of our heuristic, we run $\mathcal{A}^*_{NoK}$ on regular topologies: rings and chains. Over those topologies, we simulate a duty-cyle of 80%. Each node independently sleeps for 20% of the time and during this period links of sleeping nodes are deleted. Considering a ring topology with $|V| = 100$, the average convergence time is around 26986 rounds for 100 experiments, for the chain the convergence time is on average 70000 rounds . We also tested random $G_{(n,p)}$ topologies where $p = 2t$, in this case the average over 200 experiments shows a convergence time of 1059 rounds. The most noticeable phenomenon is that on graphs with duty-cyle both the termination heuristic and the size estimation perform really bad, on ring and chain the termination heuristic always fails, on the random graphs fails on the 23% of the instances.

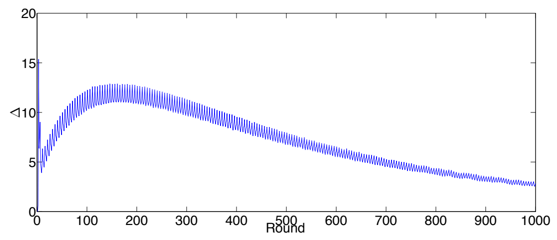


Fig. 6: Difference between the size estimated with the flow ($\mathcal{A}^*_{NoK}$) and the size estimated by looking to the energy stored at the leader ($\mathcal{A}_{NoK}$) of 200 runs for duty cycle and random graph with $|V|$=100.

4.5 Evaluation of $\mathcal{A}^*_{LCO}$

We evaluate the termination time of $\mathcal{A}^*_{LCO}$ over $G_{(n,p)}$, edge-markovian and Random Connected graphs. Let us recall that this algorithm only works on instances that are 1-interval connected. The basic $\mathcal{A}_{LCO}$ algorithm employs, on average, 15 round for $|V| = 10$, 393 for $|V| = 100$ and 7753 for $|V| = 1000$, in Table 2 we can see the performance for the symmetry breaking version (i.e., $\mathcal{A}^*_{LCO}$). As expected the symmetry breaking extension allows the algorithm to terminate faster, the termination time is close to the size of the network. Moreover we can see that the additional knowledge offered by the LCO allows the counting algorithm to count faster then the $\mathcal{A}_{NoK}$ or the $\mathcal{A}^*_{NoK}$ one.

Model	$G_{(n,p)}$ $p = 2t$			Edge-Markovian $p_b = 2t$			Random Connected Graph		
$\lvert V\rvert$	10	100	1000	10	100	1000	10	100	1000
Average Termination	7,6	107,6	1690,6	9,9	113,4	15543,2	10,14	113,8	1559
Max Termination	17	187	2117	19	222	2684	23	220	1899
Min Termination	5	96	1175	5	99	807	6	13	1263

Fig. 7: Termination performance of $\mathcal{A}^*_{LCO}$ on 1-interval connected instances.

5 Conclusion and Future Works

In this work, we presented two new practical algorithms, $\mathcal{A}^*_{NoK}$ and $\mathcal{A}^*_{LCO}$ based on the notion of energy transfer by enhancing the original ones $\mathcal{A}_{NoK}, \mathcal{A}_{LCO}$ presented in [14]. Such algorithms have been implemented, tested and compared among them. The experiments show that $\mathcal{A}^*_{NoK}$ terminates correctly on dense graphs and it has acceptable performances on disconnected instances; however, its error rate became high when we consider sparse and extremely disconnected instances or regular ones where the dynamicity is due to duty-cyclying. An interesting point revealed by the analysis is that the when the heuristic fails, it exhibits a bimodal behavior. This interesting feature has to be further investigated to understand if and how it is possible to design better heuristics. Thanks to the concept of energy-flow, $\mathcal{A}^*_{NoK}$ could answer faster to predicates like $|V| \geq T$, pushing towards a practical use of energy-transfer based algorithms. Moreover, the expected and somehow regular behavior of $\mathcal{A}_{NoK}$ slow convergence on sparse and disconnected instances could be exploited to design algorithms that want to estimate, in a distributed-way, the edge-density and the connectivity of dynamic anonymous graph with size knowledge. As a matter of fact, the convergence time seems to strictly depend from the probability threshold of edge creation p. As last contribution, we presented $\mathcal{A}^*_{LCO}$ that terminates fast on dynamic graphs where the dynamicity model is a random adversary, showing an interesting trade-off between the knowledge of the network (the use of LCO) and the counting time.

References

1. Jung. `http://jung.sourceforge.net/`, 2013.
2. S. Albers and M.R. Henzinger. Exploring unknown environments. *SIAM J. Comput.*, 29(4):1164–1188, 2000.
3. Dana Angluin. Local and global properties in networks of processors (extended abstract). In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, STOC '80, pages 82–93. ACM, 1980.
4. James Aspnes, Faith Ellen Fich, and Eric Ruppert. Relationships between broadcast and shared memory in reliable anonymous distributed systems. *Distributed Computing*, 18(3):209–219, February 2006.
5. 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 *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I*, ICALP '08, pages 121–132, Berlin, Heidelberg, 2008. Springer-Verlag.
6. B. Awerbuch, O. Goldreich, R. Vainish, and D. Peleg. A trade-off between information and communication in broadcast protocols. *Journal of the ACM (JACM)*, 37(2):238–256, 1990.
7. Roberto Baldoni, Marin Bertier, Michel Raynal, and Sara Tucci Piergiovanni. Looking for a definition of dynamic distributed systems. In Victor E. Malyshkin, editor, *PaCT*, volume 4671 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2007.
8. Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *CoRR*, abs/1012.0009, 2010.
9. Jérémie Chalopin, Yves Métivier, and Thomas Morsellino. On snapshots and stable properties detection in anonymous fully distributed systems (extended abstract). In Guy Even and Magnús Halldórsson, editors, *Structural Information and Communication Complexity*, volume 7355 of *Lecture Notes in Computer Science*, pages 207–218. Springer Berlin / Heidelberg, 2012. 10.1007/978-3-642-31104-8_18.
10. Jianmin Chen and Jie Wu. A survey on cryptography applied to secure mobile ad hoc networks and wireless sensor networks, 2010.
11. Andrea E. F. Clementi, Francesco Pasquale, Angelo Monti, and Riccardo Silvestri. Communication in dynamic radio networks. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, PODC '07, pages 205–214, New York, NY, USA, 2007. ACM.
12. Andrea E.F. Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Flooding time in edge-markovian dynamic graphs. In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing*, PODC '08, pages 213–222, New York, NY, USA, 2008. ACM.
13. X. Deng and C.H. Papadimitriou. Exploring an unknown graph. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 355–361. IEEE, 1990.
14. G. Di Luna, R. Baldoni, S. Bonomi, and I. Chatzigiannakis. Counting on Anonymous Dynamic Networks through Energy Transfer. Technical Report 1, 2013.
15. Shlomi Dolev. *Self-stabilization*. MIT Press, Cambridge, MA, USA, 2000.
16. P. Fraigniaud, A. Pelc, D. Peleg, and S. Pérennes. Assigning labels in unknown anonymous networks. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 101–111. ACM, 2000.
17. Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 513–522, New York, NY, USA, 2010. ACM.
18. Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Brief announcement: Naming and counting in anonymous unknown dynamic networks. In *DISC*, pages 437–438, 2012.
19. Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Causality, influence, and computation in possibly disconnected synchronous dynamic networks. In Roberto Baldoni, Paola Flocchini, and Ravindran Binoy, editors, *OPODIS*, volume 7702 of *Lecture Notes in Computer Science*, pages 269–283. Springer, 2012.
20. Regina O'Dell and Rogert Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of the 2005 joint workshop on Foundations of mobile computing*, DIALM-POMC '05, pages 104–110, New York, NY, USA, 2005. ACM.
21. P. Panaite and A. Pelc. Exploring unknown undirected graphs. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 316–322. Society for Industrial and Applied Mathematics, 1998.
22. Naoshi Sakamoto. Comparison of initial conditions for distributed algorithms on anonymous networks. In *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, PODC '99, pages 173–179. ACM, 1999.
23. Christian Scheideler. Models and techniques for communication in dynamic networks. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '02, pages 27–49, London, UK, UK, 2002. Springer-Verlag.
24. Masafumi Yamashita and Tiko Kameda. Computing on an anonymous network. In *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*, PODC '88, pages 117–130, New York, NY, USA, 1988. ACM.
25. Yun Zhou, Yuguang Fang, and Yanchao Zhang. Securing wireless sensor networks: a survey. *Communications Surveys Tutorials, IEEE*, 10(3):6 –28, 2008.