Computing in Dynamic Networks[⋆]

Othon Michail[1], Ioannis Chatzigiannakis[1], and Paul G. Spirakis[1,2]

[1] Computer Technology Institute & Press "Diophantus" (CTI), Patras, Greece
[2] Department of Computer Science, University of Liverpool, UK
Email: michailo@cti.gr, ichatz@cti.gr, P.Spirakis@liverpool.ac.uk

1 Introduction

1.1 Motivation-State of the art

Distributed computing systems are more and more becoming dynamic. The static and relatively stable models of computation can no longer represent the plethora of recently established and rapidly emerging information and communication technologies. In recent years, we have seen a tremendous increase in the number of new mobile computing devices. Most of these devices are equipped with some sort of communication, sensing, and mobility capabilities. Even the Internet has become mobile. The design is now focused on complex collections of heterogeneous devices that should be robust, adaptive, and self-organizing, possibly moving around and serving requests that vary with time. Delay-tolerant networks are highly-dynamic, infrastructure-less networks whose essential characteristic is a possible absence of end-to-end communication routes at any instant. Mobility may be *active*, when the devices control and plan their mobility pattern (e.g. mobile robots), or *passive*, in opportunistic-mobility networks, where mobility stems from the mobility of the carries of the devices (e.g. humans carrying cell phones) or a combination of both (e.g. the devices have partial control over the mobility pattern, like for example when GPS devices provide route instructions to their carriers). Thus, it can vary from being completely predictable to being completely unpredictable. Gossip-based communication mechanisms, e-mail exchanges, peer-to-peer networks, and many other contemporary communication networks all assume or induce some sort of highly-dynamic communication network.

The formal study of dynamic communication networks is hardly a new area of research. There is a huge amount of work in distributed computing that deals with causes of dynamicity such as failures and changes in the topology that are rather slow and usually eventually stabilize (like, for example, in self-stabilizing systems [Dol00]). However the low rate of topological changes that is usually assumed there is unsuitable for reasoning about truly dynamic networks. Even graph-theoretic techniques need to be revisited: the suitable graph model is now that of a *dynamic graph* (a.k.a. *temporal graph* or *time-varying graph*) (see e.g. [MMCS13, KKK00, CFQS12, HS12]), in which each edge has an associated set of time-labels indicating availability times. Though static graphs have been extensively studied, for their temporal generalization we are still far from having a concrete set of structural and algorithmic principles. Additionally, it is not yet clear how is the complexity of combinatorial optimization problems affected by introducing to them a notion of time. In an early but serious attempt to answer this question, Orlin [Orl81] observed that many dynamic languages derived from **NP**-complete languages can be shown to be **PSPACE**-complete.

[⋆] Supported in part by the project "Foundations of Dynamic Distributed Computing Systems" (FOCUS) which is implemented under the "ARISTEIA" Action of the Operational Programme "Education and Lifelong Learning" and is co-funded by the European Union (European Social Fund) and Greek National Resources.

Among the other few things that we do know, is that the max-flow min-cut theorem holds with unit capacities for time-respecting paths [Ber96]. However, there are fundamental properties of classical graphs that do not easily carry over to their temporal counterparts. For example, Kempe, Kleinberg, and Kumar [KKK00] found out that there is no analogue of Menger's theorem [3] for arbitrary temporal networks with one label on every edge, which additionally renders the computation of the number of node-disjoint s-t paths **NP**-complete. In a very recent work [MMCS13], the authors achieved a reformulation of Menger's theorem which is valid for all temporal graphs and introduced several interesting cost minimization parameters for optimal temporal network design. One is the *temporality* of a graph G, in which the goal is to create a temporal version of G minimizing the maximum number of labels of an edge, and the other is the *temporal cost* of G, in which the goal is to minimize the total number of labels used. Optimization of these parameters is performed subject to some *connectivity constraint.* They proved several upper and lower bounds for the temporality of some very basic graph families such as rings, directed acyclic graphs, and trees, as well as a trade-off between the temporality and the maximum label of rings. Furthermore, they gave a *generic method* for computing a lower bound of the temporality of an arbitrary graph w.r.t. the constraint of preserving a time-respecting analogue of every simple path of G. Finally, they proved that computing the temporal cost w.r.t. the constraint of preserving at least one time-respecting path from u to v whenever v is reachable from u in G, is **APX**-hard. Even the standard network diameter metric is no more suitable and has to be replaced by a dynamic/temporal version. In a dynamic star graph in which all leaf-nodes but one go to the center one after the other in a modular way, any message from the node that enters last the center to the node that never enters the center needs $n-1$ steps to be delivered, where n is the size (number of nodes) of the network; that is the *dynamic diameter* is $n-1$ while, one the other hand, the classical diameter is just 2 [AKL08] (see also [KO11]).

Distributed systems with worst-case dynamicity were first studied in [OW05]. Their outstanding novelty was to assume a communication network that may change arbitrarily from time to time subject to the condition that each instance of the network is connected. They studied asynchronous communication and considered nodes that can detect local neighborhood changes; these changes cannot happen faster than it takes for a message to transmit. They studied *flooding* (in which one node wants to disseminate one piece of information to all nodes) and *routing* (in which the information need only reach a particular destination node t) in this setting. They described a uniform protocol for flooding that terminates in $O(Tn^2)$ rounds using $O(\log n)$ bit storage and message overhead, where T is the maximum time it takes to transmit a message. They conjectured that without identifiers (IDs) flooding is impossible to solve within the above resources. Finally, a uniform routing algorithm was provided that delivers to the destination in $O(Tn)$ rounds using $O(\log n)$ bit storage and message overhead.

Computation under worst-case dynamicity was further studied in a series of works by Kuhn *et al.* in the synchronous case. In [KLO10], the network was assumed to be T*-interval connected* meaning that any time-window of length T has a static connected spanning subgraph (persisting throughout the window). Among others, *counting* (in which nodes must determine the size of the network) and *all-to-all token dissemination* (in which n different pieces of information, called tokens, are handed out to the n nodes of the network, each node being assigned one token, and all nodes must

[3] *Menger's theorem*, which is the analogue of the max-flow min-cut theorem for udirected graphs, states that the maximum number of node-disjoint s-t paths is equal to the minimum number of nodes needed to separate s from t (see e.g. [Bol98]).

collect all n tokens) were solved in $O(n^2/T)$ rounds using $O(\log n)$ bits per message, almost-linear-time randomized approximate counting was established for $T = 1$, and two lower bounds on token dissemination were given. Dutta *et al.* [DPR+13] improved one of these lower bounds and presented offline centralized algorithms for the token dissemination problem. Several variants of *coordinated consensus* in 1-interval connected networks were studied in [KMO11]. Two interesting findings were that in the absence of a good initial upper bound on n, eventual consensus is as hard as computing deterministic functions of the input and that *simultaneous consensus* can never be achieved in less than $n-1$ rounds in any execution. [Hae11] is a recent work that presents information spreading algorithms in worst-case dynamic networks based on *network coding*. An *open* setting (modeled as high churn) in which nodes constantly join and leave has very recently been considered in [APRU12]. For an excellent introduction to distributed computation under worst-case dynamicity see [KO11]. Some very thorough surveys on dynamic networks are [Sch02, CFQS12, HS12].

Here we are mostly concerned with: (i) (Section 4) [MCS12, MCS13] that studied the fundamental naming and counting problems (and some variations) in dynamic networks that are anonymous, unknown, and possibly dynamic. In *counting*, nodes must determine the size of the network n and in *naming* they must end up with unique identities. Networks are *anonymous* because all nodes begin from identical states apart possibly from a unique leader node and *unknown* because nodes have no a priori knowledge of the network (apart from some minimal knowledge when necessary) including ignorance of n. The network dynamicity model in these papers was the one of [KLO10]. (ii) (Section 5) The worst-case distributed model of [MCS14], in which the requirement for continuous connectivity was first dropped. That work proposed a set of metrics for capturing the speed of information spreading in a dynamic network that may be disconnected at every instant and efficient algorithms were developed.

Another notable model for dynamic distributed computing systems is the *population protocol* (PP) model [AAD+06]. In that model, the computational agents are passively mobile, interact in ordered pairs, and the connectivity assumption is a *strong global fairness condition* according to which all events that may always occur, occur infinitely often. These assumptions give rise to some sort of structureless interacting automata model. The usually assumed *anonymity* and *uniformity* (i.e. n is not known) of protocols only allow for commutative computations that eventually stabilize to a desired configuration. Several computability issues in this area have already been established. Constant-state nodes on a complete interaction network (and several variations) compute the *semilinear predicates* [AAER07]. Semilinearity persists up to $o(\log \log n)$ local space but not more than this [CMN+11]. If constant-state nodes can additionally leave and update fixed-length pairwise marks then the computational power dramatically increases to the commutative subclass of $\mathbf{NSPACE}(n^2)$ [MCS11a]. Interestingly, when operating under a uniform random scheduler, population protocols are formally equivalent to *chemical reaction networks* (CRNs), which model chemistry in a *well-mixed solution* and are widely used to describe information processing occurring in natural cellular regulatory networks [Dot14]. However, CRNs and population protocols can only capture the dynamics of molecular counts and not of structure formation. Then [MS14] studied the fundamental problem of *network construction* by a distributed computing system. It initiated this study by proposing and studying a very *simple*, yet sufficiently generic, model for distributed network construction. To this end, the authors assumed (as in [AAD+06, MCS11a]) the computationally weakest type of processes, i.e. finite automata, and also a very minimal adversarial communication model. The model of [MS14] may be viewed as an extension of population protocols and CRNs aiming to capture the stable structures that may occur in a well-mixed solution. They gave protocols (optimal in some cases) and lower bounds for several basic network construction problems such as *spanning line*, *spanning ring*, *spanning star*, and *regular network* and they proved several *universality* results by presenting generic protocols that are capable of simulating a Turing Machine (TM) and exploiting it in order to construct a large class of networks. For introductory texts to this area see [AR07, MCS11b].

1.2 Structure of the Chapter

In this chapter, our focus is on computational network analysis from a theoretical point of view. In particular, we study the *propagation of influence and computation in dynamic distributed computing systems.* We focus on a *synchronous message passing* communication model with bidirectional links. Our network dynamicity assumption is a *worst-case dynamicity* controlled by an adversary scheduler, which has received much attention recently. Section 2 formally defines the dynamic network models under consideration and the problems studied throughout. Section 3 discusses the central notion of causal influence and the 1-interval connectivity model. In Section 4, we study the fundamental *naming* and *counting* problems (and some variations) in networks that are *anonymous*, *unknown*, and possibly dynamic. Network dynamicity is modeled here by the *1-interval connectivity model* [KLO10], in which communication is synchronous and a (worst-case) adversary chooses the edges of every round subject to the condition that each instance is connected. Then, in Section 5 we replace the assumption that the network is connected at every instant by minimal *temporal connectivity* conditions. These conditions only require that *another causal influence occurs within every time-window of some given length.* Based on this basic idea we define several novel metrics for capturing the speed of information spreading in a dynamic network. We present several results that correlate these metrics. Moreover, we investigate *termination criteria* in networks in which an upper bound on any of these metrics is known. We exploit these termination criteria to provide efficient (and optimal in some cases) protocols that solve the fundamental *counting* and *all-to-all token dissemination* (or *gossip*) problems. In Section 6, we propose another model of worst-case temporal connectivity, called *local communication windows*, that assumes a fixed underlying communication network and restricts the adversary to allow communication between local neighborhoods in every time-window of some fixed length. We prove some basic properties and provide a protocol for counting in this model. Finally, in Section 7 we conclude and discuss some interesting future research directions.

2 Preliminaries

2.1 The Dynamic Network Model

A *dynamic network* is modeled by a *dynamic graph* $G = (V, E)$, where V is a set of n nodes (or processors) and $E : \mathbb{N} \to \mathcal{P}(E')$ (wherever we use $\mathbb{N}$ we mean $\mathbb{N}_{\geq 1}$) is a function mapping a round number $r \in \mathbb{N}$ to a set $E(r)$ of bidirectional links drawn from $E' = \{\{u, v\} : u, v \in V\}$. [4] 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.* A *static network* is just a special case of a dynamic network in which $E(i + 1) = E(i)$ for all $i \in \mathbb{N}$. The set V is assumed throughout this section to be *static*, that is it remains the same throughout the execution.

[4] By $\mathcal{P}(S)$ we denote the *powerset* of the set S, that is the set of all subsets of S.

A dynamic graph/network $G = (V, E)$ is said to be 1-*interval connected*, if, for all $r \in \mathbb{N}$, the static graph $G(r)$ is connected [KLO10]. Note that this allows the network to change arbitrarily from round to round always subject to the condition that it remains connected. In Section 4, we focus on 1-interval connected dynamic networks which also implies that we deal with connected networks in the static-network case.

In Section 4, we assume that nodes in V are *anonymous*, by which we mean they do not initially have any ids and also we assuem that they do not know the topology or the size of the network, apart from some minimal knowledge when necessary (i.e. we say that *the network is unknown*). In several cases, and in order to break symmetry, we may assume a unique *leader node* (or *source*) l. If this is the case, then we assume that l starts from a unique initial state l_0 (e.g. 0) while all other nodes start from the same initial state q_0 (e.g. $\perp$). All nodes but the leader execute identical programs. In Section 5, we assume that nodes in V have unique identities (ids) drawn from some namespace $\mathcal{U}$ (we assume that ids are represented using $O(\log n)$ bits) and again that they do not know the topology or the size of the network, apart from some minimal necessary knowledge to allow for terminating computations (usually an upper bound on the time it takes for information to make some sort of progress). Any such assumed knowledge will be clearly stated. In all cases, nodes have unlimited local storage (though they usually use a reasonable portion of it).

Communication is *synchronous message passing* [Lyn96, AW04], meaning that it is executed in discrete steps controlled by a global clock that is available to the nodes and that nodes communicate by sending and receiving messages (usually of length that is some reasonable function of n, like e.g. $\log n$). Thus all nodes have access to the current round number via a local variable that we usually denote by r. We use the terms *round*, *time*, and *step* interchangeably to refer to the discrete steps of the system. Naturally, real rounds begin to count from 1 (e.g. "first round") and we reserve time 0 to refer to the initial state of the system. We consider two different models of message transmission. One is *anonymous broadcast*, in which, in every round r, each node u generates a single message $m_u(r)$ to be delivered to all its current neighbors in $N_u(r) = \{v : \{u, v\} \in E(r)\}$. The other is *one-to-each* in which a different message $m_{(u,i)}(r)$, $1 \leq i \leq d_u(r)$, where $d_u(r) := |N_u(r)|$ is the degree of u in round r, may be generated for each neighbor v_i.

In every round, the adversary first chooses the edges for the round; for this choice it can see the internal states of the nodes at the beginning of the round. In the one-to-each message transmission model we additionally assume that the adversary also reveals to each node u a set of locally unique edge-labels $1, 2, \ldots, d_u(r)$, one for each of the edges currently incident to it. Note that these labels can be reselected arbitrarily in each round so that a node cannot infer what the internal state of a neighbor is based solely on the corresponding local edge-name. Then each node transitions to a new state based on its internal state (containing the messages received in the previous round) and generates its messages for the current round: in anonymous broadcast a single message is generated and in one-to-each a different message is generated for each neighbor of a node. Note that, in both models, a node does not have any information about the internal state of its neighbors when generating its messages. Deterministic algorithms are only based on the current internal state to generate messages. This implies that the adversary can infer the messages that will be generated in the current round before choosing the edges. Messages are then delivered to the corresponding neighbors. In one-to-each, we assume that each message m_i received by some node u is accompanied with u's local label i of the corresponding edge, so that a node can associate a message sent through edge i with a message received from edge i. These messages will be processed by the nodes in the subsequent round so we typically begin rounds with a "receive" command referring to the messages received in the previous round. Then the next round begins.

2.2 Problem Definitions

We investigate the computability of the following fundamental problems for distributed computing in the context of dynamic networks.

k-labeling. An algorithm is said to solve the k-labeling problem if whenever it is executed on a network comprising n nodes each node u eventually terminates and outputs a *label* (or *name* or *id*) id_u so that $|\{id_u : u \in V\}| \geq k$.

Naming. The naming problem is a special case of the k-labeling problem in which it must additionally hold that $k = n$. This, in turn, implies that $id_u \neq id_v$ for all distinct $u, v \in V$ (so, unique labels are required for the nodes).

Minimal (Consecutive) Naming. It is a special case of naming in which it must additionally hold that the n nodes output the labels $\{0, 1, \ldots, n-1\}$.

Counting Upper Bound. Nodes must determine an upper bound k on the network size n.

Counting. A special case of counting upper bound in which it must hold that $k = n$.

All-to-all Token Dissemination (or Gossip). There is a token assignment function $I : V \to \mathcal{T}$ that assigns to each node $u \in V$ a single token $I(u)$ from some domain $\mathcal{T}$ s.t. $I(u) \neq I(v)$ for all $u \neq v$. An algorithm solves all-to-all token dissemination if for all instances (V, I), when the algorithm is executed in any dynamic graph $G = (V, E)$, all nodes eventually terminate and output $\bigcup_{u \in V} I(u)$. We assume that each token in the nodes' input is represented using $O(\log n)$ bits. The nodes know that each node starts with a unique token but they do not know n.

3 Spread of Influence in Dynamic Graphs (Causal Influence)

Probably the most important notion associated with a dynamic network/graph is the *causal influence*, which formalizes the notion of one node "influencing" another through a chain of messages originating at the former node and ending at the latter (possibly going through other nodes in between). Recall that we denote by (u, t) the state of node u at time t and usually call it the *t-state of u*. The pair (u, t) is also called a *time-node*. We again use $(u, r) \rightsquigarrow (v, r')$ to denote the fact that node u's state in round r influences node v's state in round r'. Formally:

Definition 1 ([Lam78]). *Given a dynamic graph $G = (V, E)$ we define an order $\to \subseteq (V \times \mathbb{N}_{\geq 0})^2$, where $(u, r) \to (v, r+1)$ iff $u = v$ or $\{u, v\} \in E(r+1)$. The* causal order $\rightsquigarrow \subseteq (V \times \mathbb{N}_{\geq 0})^2$ *is defined to be the reflexive and transitive closure of $\to$.*

Obviously, for a dynamic distributed system to operate as a whole there must exist some upper bound on the time needed for information to spread through the network. This is a very weak guarantee as without it global computation is in principle impossible. An abstract way to talk about information spreading is via the notion of the *dynamic diameter*. The *dynamic diameter* (also called *flooding time*, e.g., in [CMM$^+$08, BCF09]) of a dynamic graph, is an upper bound on the time required for each node to causally influence (or, equivalently, to be causally influenced by) every other node; formally, the dynamic diameter is the minimum $D \in \mathbb{N}$ s.t. for all times $t \geq 0$ and all $u, v \in V$ it holds that $(u, t) \rightsquigarrow (v, t + D)$. A small dynamic diameter allows for fast dissemination of information. Throughout, we do not allow nodes to know the dynamic diameter of the network. We only allow some minimal knowledge (that will be explained every time) based on which nodes may infer bounds on the dynamic diameter.

A class of dynamic graphs with small dynamic diameter is that of T*-interval connected* graphs. As already stated, T-interval connectivity was proposed in [KLO10] as an elegant way to capture a special class of dynamic networks, namely *those that are connected at every instant.* Intuitively, the parameter T represents the rate of connectivity changes. Formally, a dynamic graph $G = (V, E)$ is said to be T*-interval connected*, for $T \geq 1$, if, for all $r \in \mathbb{N}$, the static graph $G_{r,T} := (V, \bigcap_{i=r}^{r+T-1} E(r))$ is connected [KLO10]; that is, in every time-window of length T, a connected spanning subgraph is preserved. In one extreme, if $T = 1$ then the underlying connected spanning subgraph may change arbitrarily from round to round and in the other extreme if T is ∞ then a connected spanning subgraph must be preserved forever. Recall that T-interval connected networks have the very nice feature to allow for constant propagation of information. For example, 1-interval connectivity guarantees that the state of a node causally influences the state of another uninfluenced node in every round (if one exists). To get an intuitive feeling of this fact, consider a partitioning of the set of nodes V to a subset V_1 of nodes that know the r-state of some node u and to a subset $V_2 = V \backslash V_1$ of nodes that do not know it. Connectivity asserts that there is always an edge in the cut between V_1 and V_2, consequently, if nodes that know the r-state of u broadcast it in every round, then in every round at least one node moves from V_2 to V_1. This is formally captured by the following lemma from [KLO10]:

Lemma 1 ([KLO10]). *For any node $u \in V$ and time $r \geq 0$, in a 1-interval connected network, we have*

1. $|\{v \in V : (u, 0) \rightsquigarrow (v, r)\}| \geq \min\{r + 1, n\}$,
2. $|\{v \in V : (v, 0) \rightsquigarrow (u, r)\}| \geq \min\{r + 1, n\}$.

Before proving the lemma, let us define two very useful sets. For all times $0 \leq t \leq t'$, we define by $\text{past}_{(u,t')}(t) := \{v \in V : (v, t) \rightsquigarrow (u, t')\}$ [KMO11] the *past set of a time-node (u, t') from time t* and by $\text{future}_{(u,t)}(t') := \{v \in V : (u, t) \rightsquigarrow (v, t')\}$ the *future set of a time-node (u, t) at time t'*. In words, $\text{past}_{(u,t')}(t)$ is the set of nodes whose t-state (i.e. their state at time t) has causally influenced the t'-state of u and $\text{future}_{(u,t)}(t')$ is the set of nodes whose t'-state has been causally influenced by the t-state of u. If $v \in \text{future}_{(u,t)}(t')$ we say that at time t' node v *has heard of/from* the t-state of node u. If it happens that $t = 0$ we say simply that v has heard of u. Note that $v \in \text{past}_{(u,t')}(t)$ iff $u \in \text{future}_{(v,t)}(t')$.

Proof. If $t = 0$, we have $\text{future}_{(u,0)}(0) = \text{past}_{(u,0)}(0) = \{u\} \Rightarrow |\text{future}_{(u,0)}(0)| = |\text{past}_{(u,0)}(0)| = 1 \geq \min\{0 + 1, n\} = 1$ and both statements hold in the base case.

1. Assume that $|future_{(u,0)}(i)| \geq \min\{i + 1, n\}$ for some $i > 0$. If $\min\{i + 1, n\} = n$ then clearly $\min\{i + 2, n\} = n$ and the statement also holds for time $i + 1$. If $\min\{i + 1, n\} < n$ then the set $T = V \backslash future_{(u,0)}(i)$ is non-empty. Connectivity in round $i + 1$ implies that there is some edge $\{v, w\}$ in the cut between $\text{future}_{(u,0)}(i)$ and T (such an edge joins the set of nodes whose state at

time i has been influenced by the initial state of u and those that has not). This in turn implies that w is influenced during round $i+1$ so that $|\text{future}_{(u,0)}(i+1)| \geq |\text{future}_{(u,0)}(i)|+1$ (increases by at least 1). So the statement remains true for time $i+1$. Informally, the set of nodes that have been influenced by the initial state of u increases by at least 1 in each round (while this set is smaller than V) due to connectivity and clearly in $n-1$ rounds all nodes must have been influenced by the initial state of any other node.

2. This one is a little more subtle. The reason is that here $|\text{past}_{(u,r)}(0)|$ does not necessarily increase as r increases (may remain the same in some rounds). For an example consider a 1st round in which a node u is connected to $n-2$ nodes $\{v_1, v_2, ..., v_{n-2}\}$ and a node w is connected only to v_{n-2}. At time 1 we have $|\text{past}_{(u,1)}(0)| = n-1$. From now on, the graph maintains the following static structure: the graph is the hamiltonian path $u, v_1, v_2, ..., v_{n-2}, w$. Clearly, the initial state of w must travel across the path to influence u, so no new influence occurs at u for $n-3$ rounds.

Let us now prove this statement. Assume that $|\text{past}_{(u,i)}(0)| \geq \min\{i+1, n\}$ for some $i > 0$. The only interesting case is when $|\text{past}_{(u,i)}(0)| = i+1 < n$ (if it is $> i+1$ then the statement trivially holds for the next round). Again $T = V \backslash past_{(u,i)}(0)$ is non-empty. Due to case 1, the initial configuration of the set T needs $V \backslash T = i+1$ rounds to influence all nodes in $\text{past}_{(u,i)}(0)$. Thus, again the initial state of some node in T influences u during round $i+1$ and the statement follows. □

For a distributed system to be able to perform global computation, nodes need to be able to determine for all times $0 \leq t \leq t'$ whether $\text{past}_{(u,t')}(t) = V$. If nodes know n, then a node can easily determine at time t' whether $\text{past}_{(u,t')}(t) = V$ by counting all different t-states that it has heard of so far (provided that every node broadcasts at every round all information it knows). If it has heard the t-states of all nodes then the equality is satisfied. If n is not known then various techniques may be applied (which is the subject of this section). By *termination criterion* we mean any locally verifiable property that can be used to determine whether $\text{past}_{(u,t')}(t) = V$.

Remark 1. Note that any protocol that allows nodes to determine whether $\text{past}_{(u,t')}(t) = V$ can be used to solve the counting and all-to-all token dissemination problems. The reason is that if a node knows at round r that it has been causally influenced by the initial states of all other nodes, then it can solve counting by writing $|\text{past}_{(u,r)}(0)|$ on its output and all-to-all dissemination by writing $\text{past}_{(u,r)}(0)$ (provided that all nodes send their initial states and all nodes constantly broadcast all initial states that they have heard of so far).

4 Naming and Counting in Anonymous Unknown Dynamic Networks

In this section, we study the fundamental naming and counting problems (and some variations) in networks that are anonymous, unknown, and possibly dynamic. In *counting*, nodes must determine the size of the network n and in *naming* they must end up with unique identities. By *anonymous* we mean that all nodes begin from identical states apart possibly from a unique leader node and by *unknown* that nodes have no a priori knowledge of the network (apart from some minimal knowledge when necessary) including ignorance of n. Network dynamicity is modeled by the 1-interval connectivity model [KLO10], in which communication is synchronous and a worst-case adversary chooses the edges of every round subject to the condition that each instance is connected. We first focus on static networks with broadcast where we show that a unique leader suffices in order to solve counting in linear time. Then we focus on dynamic networks with broadcast. We conjecture that dynamicity renders nontrivial computation impossible. In view of this, we let the nodes know an upper bound on the maximum degree that will ever appear and show that in this case the nodes can obtain an upper bound on n. Finally, we replace broadcast with *one-to-each*, in which a node may send a different message to each of its neighbors. Interestingly, this natural variation gives us the ability to state a correct naming protocol for this kind of dynamic distributed systems.

4.1 Further 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 [Ang80], who investigated the problem of establishing a "center". Further investigation led to the classification of computable functions [YK96, ASW88]. [BV99] removed the, until then, standard assumption of knowing the network size n and provided characterizations of the relations that can be computed with arbitrary knowledge. Other well-known studies on unknown networks have dealt with the problems of robot-exploration and map-drawing of an unknown graph [DP90, AH00] and on information dissemination [AGVP90]. Fraigniaud *et al.* [FPPP00] assumed a unique leader in order to break symmetry and assign short labels as fast as possible. To circumvent the further symmetry introduced by broadcast message transmission they also studied other natural message transmission models as sending only one message to a single neighbor. Recently, and independently of our work, Chalopin *et al.* [CMM12] have studied the problem of naming anonymous networks in the context of snapshot computation. Finally, Aspnes *et al.* [AFR06] 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.

4.2 Static Networks with Broadcast

We here assume that the network is described by a static graph $G = (V, E)$, where $E \subseteq \{\{u, v\} : u, v \in V\}$. Moreover, the message transmission model is broadcast, that is, in every round, each node u generates a single message to be delivered to all its neighbors. Note that any impossibility result established for static networks is also valid for dynamic networks as a static network is a special case of a dynamic network.

First of all, note that if all nodes start from the same initial state then, if we restrict ourselves to deterministic algorithms, naming is impossible to solve in general static networks, even if nodes know n. The reason is that in the worst-case they may be arranged in a ring (in which each node has precisely 2 neighbors) and it is a well-known fact [Ang80, Lyn96, AW04] that, in this case, in every round r, all nodes are in identical states.

We show now that impossibility persists even if we allow a unique leader and even if nodes have complete knowledge of the network.

Theorem 1 ([MCS13]). *Naming is impossible to solve by deterministic algorithms in general anonymous (static) networks with broadcast even in the presence of a leader and even if nodes have complete knowledge of the network.*

Proof. Imagine a star graph in which the leader has $n - 1$ neighbors (it is the *center*) and every other node has only the leader as its unique neighbor (they are the *leaves*). All leaf-nodes are in the same initial state and receive the same first message m_1 from the center. So they all transition to the same new state and generate the same outgoing message. It is straightforward to verify, by induction on the number of rounds, that in every round r all leaf-nodes are in identical states. In fact, in any network in which some node is connected to at least two terminal nodes, that is nodes with no further neighbors, those terminal nodes will forever be in identical states. □

An obvious generalization is that, under the same assumptions as in the statement of the theorem, it is impossible to solve k-labeling for any $k \geq 3$.

We now turn our attention to the simpler counting problem. First we establish the necessity of assuming a unique leader.

Theorem 2 ([MCS13]). *Without a leader, counting is impossible to solve by deterministic algorithms in general anonymous networks with broadcast.*

Proof. For the sake of contradiction, assume that an algorithm A solves it. Then it solves it on a static ring R_1 of size n with the first node terminating in $k \geq n$ rounds. Now consider a ring R_2 of size $k+1$. All nodes in both rings are initially in the same identical initial state $\perp$. Thus, any node in R_2 has the same k-neighborhood (states of nodes in distance at most k) as any node in R_1 which implies that after k rounds these two nodes will be in the same state (see e.g. Lemma 3.1 in [ASW88]). Thus a node in R_2 terminates after k rounds and outputs n which is a contradiction. □

In view of Theorem 2, we assume again a unique leader in order to solve counting. Recall that the *eccentricity* of a node u is defined as the greatest geodesic distance between u and v, over all $v \in V \backslash \{u\}$, where "distance" is equivalent to "shortest path". We first describe a protocol *Leader_Eccentricity* (inspired by the $Wake\&Label$ set of algorithms of [FPPP00]) that assigns to every node a label equal to its distance from the leader and then we exploit this to solve counting. We assume that all nodes have access to the current round number via a variable r.

Protocol *Leader_Eccentricity*. The leader begins with *label* $\leftarrow 0$ and *max_asgned* $\leftarrow 0$ and all other nodes with *label* $\leftarrow \perp$. In the first round, the leader broadcasts an *assign* (1) message. Upon reception of an *assign* (i) message, a node that has *label* $= \perp$ sets *label* $\leftarrow i$ and broadcasts to its neighbors an *assign* $(i+1)$ message and an *ack* (i) message. Upon reception of an *ack* (i) message, a node with *label* $\neq \perp$ and *label* $< i$ broadcasts it. Upon reception of an *ack* (i) message, the leader sets *max_asgned* $\leftarrow i$ and if $r > 2 \cdot (max\_asgned + 1)$ then it broadcasts a *halt* message, outputs its label, and halts. Upon reception of a *halt* message, a node broadcasts *halt*, outputs its label, and halts.

Theorem 3 ([MCS13]). *In Leader_Eccentricity nodes output $\epsilon + 1$ distinct labels where ϵ is the eccentricity of the leader. In particular, every node outputs its distance from the leader.*

Proof. At time 2, nodes at distance 1 from the leader receive *assign* (1) and set their label to 1. By induction on distance, nodes at distance i get label i at round $i+1$. In the same round, they send an ack that must arrive at the leader at round $2i+1$. If not then there is no node at distance i. □

We now use *Leader_Eccentricity* to solve counting in anonymous unknown static networks with a leader. We additionally assume that at the end of the *Leader_Eccentricity* process each node u knows the number of neighbors $up(u) = |\{\{v, u\} \in E : label(v) = label(u) - 1\}|$ it has to its upper level (it can store this during the *Leader_Eccentricity* process by counting the number of *assign* messages that arrived at it from its upper level neighbors). Moreover, we assume that all nodes know the leader's eccentricity ϵ (just have the leader include *max_asgned* in its *halt* message). Finally, let, for simplicity, the first round just after the completion of the above process be round $r = 1$. For this, we just need all nodes to end concurrently the *Leader_Eccentricity* process. This is done by having node with label i that receives or creates (this is true for the leader) a *halt* message in round r halt in round $(r + max\_asgned - i)$. Then the nodes just reset their round counters.

Protocol *Anonymous_Counting.* Nodes first execute the modified *Leader_Eccentricity*. When $\epsilon - r + 1 = label(u)$, a non-leader node u receives a possibly empty (in case of no lower-level neighbors) set of $partial\_count_i$ $(rval_i)$ messages and broadcasts a $partial\_count$ $((1 + \sum_i rval_i)/up(u))$ message. When $r = \epsilon + 1$, the leader receives a set of $partial\_count_i$ $(rval_i)$ messages, sets $count \leftarrow 1 + \sum_i rval_i$, broadcasts a *halt* (*count*) message, outputs *count*, and halts. When a non-leader u receives a *halt* (*count*) message, it outputs *count* and halts.

For a given round r denote by $rval_i(u)$ the ith message received by node u.

Theorem 4 ([MCS13]). *Anonymous_Counting solves the counting problem in anonymous static networks with broadcast under the assumption of a unique leader. All nodes terminate in $O(n)$ rounds and use messages of size $O(\log n)$.*

Proof. By induction on the round number r, in the beginning of round $r \geq 2$, it holds that $\sum_{u:label(u)=\epsilon-r+1} (1 + \sum_i rval_i(u)) = |\{u : label(u) \geq \epsilon - r + 1\}|$. Clearly, in round $\epsilon + 1$ it holds that $count = 1 + \sum_i rval_i(leader) = |\{u : label(u) \geq 0\}| = n$. $\square$

4.3 Dynamic Networks with Broadcast

We now turn our attention to the more general case of 1-interval connected dynamic networks with broadcast. We begin with a conjecture stating that dynamicity renders nontrivial computation impossible (see also [OW05] for a similar conjecture in a quite different setting). Then we naturally strengthen the model to allow some computation.

Conjecture 1 ([MCS13]). It is impossible to compute (even with a leader) the predicate $N_a \geq 1$, that is "exists an a in the input", in general anonymous unknown dynamic networks with broadcast.

The conjecture is essentially based on the following fact. Even in a dynamic network, it can be the case that two nodes that are initially in the same state a can for any number of rounds T have the same T-neighborhood, which means that the whole history of received messages is the same in both nodes and thus they always transition to identical states. This is, for example, true in a symmetric tree rooted at the leader (e.g. a tree with k identical lines leaving the root) in which the two nodes are in each round in equal distance from the root (even if this distance changes from round to round by moving the 2 nodes back and forth). In dynamic networks, it is also the case that for a node u to causally influence the leader with its t-state, all nodes that receive the t-state of u should continuously broadcast it at least until the leader receives it (then they could probably stop by receiving an ack or by using some known upper bound on the delivery time). Potentially, $O(n)$ nodes can receive the t-state of u before it is delivered to the leader. It seems that if the leader could at some point decide that the received messages originate from two distinct nodes that are forever in identical states then it would also decide the same on a dynamic network containing only one of these nodes, as in both cases the whole network could be full of messages of the same kind. So, it seems impossible for the leader to determine whether the network contains at least two

as and such a process is necessary for the leader to count the size of the network. To determine whether there are no as at all, in the absence of as, the leader should somehow determine that it has been causally influenced by the whole network, which in turn requires counting.

In view of Theorem 1, which establishes that we cannot name the nodes of a static, and thus also of a dynamic, network if broadcast communication is assumed, and of the above conjecture, implying that in dynamic networks we cannot count even with a leader [5], we start strengthening our initial model.

Let us now assume that there is a unique leader l that knows an upper bound d on maximum degree ever to appear in the dynamic network, that is $d \geq \max_{u \in V, r \in \mathbb{N}}\{d_u(r)\}$. We keep the broadcast message transmission.

Note first that impossibility of naming persists. However, we show that obtaining an upper bound on the size of the network now becomes possible, though exponential in the worst case.

Protocol *Degree_Counting.* The leader stores in d the maximum degree that will ever appear and begins with $label \leftarrow 0$, $count \leftarrow 1$, $latest\_event \leftarrow 0$, $max\_label \leftarrow 0$, and $r \leftarrow 0$ while all other nodes begin with $label \leftarrow \perp$, $count \leftarrow 0$, and $r \leftarrow 0$. In the beginning of each round each node increments by one its round counter r. The leader in each round r broadcasts *assign* (r). Upon reception of an *assign* (r_label) message, a node with $label = \perp$ sets $label \leftarrow r\_label$ and from now in each round r broadcasts *assign* (r) and *my_label* $(label)$. A node with $label = \perp$ that did not receive an *assign* message sends an *unassigned* (r) message. All nodes continuously broadcast the maximum *my_label* and *unassigned* messages that they have received so far. Upon reception of an *unassigned* (i) message, the leader, if $i > latest\_event$, it sets $count \leftarrow 1$ and, for $k = 1, \ldots, i$, $count \leftarrow count + d \cdot count$, $max\_label \leftarrow i$, and $latest\_event \leftarrow r$ and upon reception of a *my_label* (j) message, if $j > max\_label$, it sets $count \leftarrow 1$ and, for $k = 1, \ldots, j$, $count \leftarrow count + d \cdot count$, $latest\_event \leftarrow r$, and $max\_label \leftarrow j$ (if receives both i, j it does it for $\max\{i, j\}$). When it holds that $r > count + latest\_event - 1$ (which must eventually occur) then the leader broadcasts a *halt* $(count)$ message for *count* rounds and then outputs *count* and halts. Each node that receives a *halt* (r_count) message, sets $count \leftarrow r\_count$, broadcasts a *halt* $(count)$ message for *count* rounds and then outputs *count* and halts.

Theorem 5 ([MCS13]). *Degree_Counting solves the counting upper bound problem in anonymous dynamic networks with broadcast under the assumption of a unique leader. The obtained upper bound is $O(d^n)$ (in the worst case).*

Proof. In the first round, the leader assigns the label 1 to its neighbors and obtains an *unassigned* (1) message from each one of them. So, it sets $count \leftarrow (d + 1)$ (in fact, note that in the first step it can simply set $count \leftarrow d_u(1) + 1$, but this is minor), $latest\_event \leftarrow 1$, and $max\_label \leftarrow 1$. Now, if there are further nodes, at most by round $count + latest\_event - 1$ it must have received an *unassigned* (i) message with $i > latest\_event$ or a *my_label* (j) with $j > max\_label$. Note that the reception of an *unassigned* (i) message implies that at least $i + 1$ distinct labels have been assigned because as long as there are unlabeled nodes one new label is assigned in each round to at least one node (this is implied by Lemma 1 and the fact that all nodes with labels constantly assign new labels). Initially, one node (the leader) assigned to at most d nodes label 1. Then the $d + 1$ labeled nodes assigned to at most $(d + 1)d$ unlabeled nodes the label 2, totalling $(d + 1) + (d + 1)d$, and so on.

[5] This is implied because if we could count we could have a node wait at most $n - 1$ rounds until it hears of an a (provided that all nodes that have heard of an a forward it) and if no reject.

In the worst-case, each label in $\{0, 1, \ldots, n-1\}$ is assigned to precisely one node (e.g., consider a static line with the leader in the one endpoint). In this case the nodes count $O(d^n)$. $\square$

We point out that if nodes have access to more drastic initial knowledge such as an upper bound e on the *maximum expansion*, defined as $\max_{u,r,r'}\{|\text{future}_{u,r}(r'+1)| - |\text{future}_{u,r}(r')|\}$ (maximum number of concurrent new influences ever occuring), where $\text{future}_{(u,r)}(r') := \{v \in V : (u,r) \rightsquigarrow (v,r')\}$, for $r \leq r'$, then essentially the same protocol as above provides an $O(n \cdot e)$ upper bound.

4.4 Dynamic Networks with One-to-Each

The result of Theorem 1, in the light of (a) Conjecture 1, and (b) the assumption of a broadcast message transmission model, indicates that nontrivial computations in anonymous unknown dynamic networks are impossible even under the assumption of a unique leader. We now relax our assumptions so that we can state a correct naming protocol. We start by relaxing the assumption of a broadcast message transmission medium by offering to nodes access to a *one-to-each message transmission mechanism.* We also assume a unique leader, as without it, even under a one-to-each model, naming is impossible to solve.

1st Version - Protocol *Fair* We first present a protocol, that we call *Fair*, in which the unique leader assigns distinct labels to each node of the network. The labels assigned are tuples (r, h, i), where r is the round during which the label was assigned, h is the label of the leader node and i is a unique number assigned by the leader. The labels can be uniquely ordered first by r, then by h and finally by i (in ascending order).

Each node maintains the following local variables: *clock*, for counting the rounds of execution of the protocol (implemented due to synchronous communication, see Section 2.1), *label*, for storing the label assigned by the leader, *state*, for storing the local state that can be set to $\{anonymous, named, leader\}$, and *counter*, for storing the number of labels generated. All nodes are initialized to $clock \leftarrow 0$, $id \leftarrow (0, \perp, \perp)$, $state \leftarrow anonymous$, and $counter \leftarrow 0$ except from the leader that is initialized to $clock \leftarrow 0$, $id \leftarrow (0, 1, 1)$, $state \leftarrow leader$, and $counter \leftarrow 1$.

Each turn the leader u consults the one-to-each transmission mechanism and identifies a set of locally unique edge-labels $1, 2, \ldots, d(u)$, one for each of the edges incident to it. [6] The leader iterates the edge-label set and transmits to each neighboring node a different message m_i, $1 \leq i \leq d(u)$ that contains the unique label $(clock, label, counter + i)$. When the transmission is complete, it increases the variable *counter* by $d(u)$. All the other nodes of the network do not transmit any messages (or transmit a null message if message transmission is compulsory).

All nodes under $state = anonymous$, upon receiving a (non-null) message set the local *label* to the contents of the message and change *state* to *named*. All the other nodes of the network simply ignore all the messages received.

At the end of the turn all nodes do $clock{+}{+}$ (where '++' is interpreted as "increment by one").

Recall that a naming assignment is correct if *all nodes* are assigned *unique* labels. It is clear that *Fair* is a non-terminating correct protocol, given the following *fairness assumption*: the leader node at some point has become directly connected with each other node of the network (i.e., eventually meets all nodes).

[6] Recall from Section 4.1 that these edge-labels can be reselected arbitrarily in each round (even if the neighbors remain the same) by the adversary so that a node cannot infer what the internal state of a neighbor is, based solely on the corresponding local edge-name.

Lemma 2. *With one-to-each transmission, under the fairness assumption, and in the presence of a unique leader, protocol $Fair$ eventually computes a unique assignment for all the nodes in any anonymous unknown dynamic network.*

2nd Version - Protocol *Delegate* We now proceed by presenting a stronger protocol *Delegate* (based on $Fair$) that is correct even without the fairness assumption. To achieve correctness the leader node delegates the role of assignment of labels to all the nodes that it encounters. Thus, without loss of generality, even if the leader does not encounter all other nodes of the network, due to the *connectivity property*, all nodes will eventually hear from the leader. Therefore, all nodes will either receive a unique label from the leader or from another labeled node. The uniqueness among the labels generated is guaranteed since each label can be traced back to the node that issued it using the h parameter.

In *Delegate* the nodes maintain the same variables as in $Fair$. Each turn the leader performs the same actions as in $Fair$. Also similarly to $Fair$, each node that is in $state = anonymous$ does not transmit any message (or transmits a null message if message transmission is compulsory). Each node u that is in $state = named$ performs similar actions as the leader node and transmits to each edge-label i a message containing the unique label $(clock_u, label_u, counter_u + i)$ and then increases the variable $counter_u$ by $d(u)$. All nodes under $state = anonymous$, upon receiving one or more (non-null) messages that contain a label, select the message that contains the lowest label (i.e., the one with the lowest h parameter) and set the local *label* to the contents of the message and change *state* to *named*. At the end of the turn all nodes do $clock++$.

Lemma 3 ([MCS13]). *With one-to-each transmission, and in the presence of a unique leader, protocol Delegate correctly computes a unique assignment for all the nodes in any anonymous unknown dynamic network.*

3rd Version - Protocol *Dynamic_Naming* (terminating) Protocol $Fair$ computes a correct naming assignment (based on different assumptions) but does not terminate. Essentially the nodes continue to transmit labels for ever. We now describe a protocol that we call $Dynamic\_Naming$ that manages to terminate. $Dynamic\_Naming$ is an $O(n)$-time protocol that assigns unique ids to the nodes and informs them of n. As usual, there is a unique leader l with id 0 while all other nodes have id $\perp$.

The idea here is as follows. All nodes that have obtained an id assign ids and these ids are guaranteed to be unique. Additionally, we have nodes that have obtained an id to acknowledge their id to the leader. Thus, all nodes send their ids and all nodes continuously forward the received ids so that they eventually arrive at the leader (simple flooding mechanism). So, at some round r, the leader knows a set of assigned ids $K(r)$. We describe now the termination criterion. If $|K(r)| \neq |V|$ then in at most $|K(r)|$ additional rounds the leader must hear (be causally influenced) from a node outside $K(r)$ (to see why, see Lemma 1). Such a node, either has an id that the leader first hears of, or has no id yet. In the first case, the leader updates $K(r)$ and in the second waits until it hears of a new id (which is guaranteed to appear in the future). On the other hand, if $|K(r)| = |V|$ no new info will ever arrive at the leader in the future and the leader may terminate after the $|K(r)|$-round waiting period ellapses. This protocol solves the naming problem in anonymous unknown dynamic networks under the assumptions of one-to-each message transmission and of a unique leader. All nodes terminate in $O(n)$ rounds and use messages of size $\Theta(n^2)$.

Protocol *Dynamic_Naming*. Initially, every node has three variables $count \leftarrow 0$, $acks \leftarrow \emptyset$, and $latest\_unassigned \leftarrow 0$ and the leader additionally has $latest\_new \leftarrow 0$, $time\_bound \leftarrow 1$, and $known\_ids \leftarrow \{0\}$. A node with $id \neq \perp$ for $1 \leq i \leq k$ sends *assign* $(id, count + i)$ message to its ith neighbor and sets $count \leftarrow count + k$. In the first round, the leader additionally sets $known\_ids \leftarrow \{0, (0, 1), (0, 2), \ldots, (0, k)\}$, $latest\_new \leftarrow 1$, and $time\_bound \leftarrow 1 + |known\_ids|$. Upon receipt of l *assign* messages (rid_j), a node with $id = \perp$ sets $id \leftarrow \min_j\{rid_j\}$ (in number of bits), $acks \leftarrow acks \cup id$, sends an *ack* $(acks)$ message to all its k current neighbors, for $1 \leq i \leq k$ sends *assign* $(id, count + i)$ message to its ith neighbor, and sets $count \leftarrow count + k$. Upon receipt of l *ack* messages $(acks_j)$, a nonleader sets $acks \leftarrow acks \cup (\bigcup_j acks_j)$ and sends *ack* $(acks)$. A node with $id = \perp$ sends *unassigned* $(current\_round)$. Upon receipt of $l \geq 0$ *unassigned* messages (val_j), a node with $id \notin \{0, \perp\}$ sets $latest\_unassigned \leftarrow \max\{latest\_unassigned, \max_j\{val_j\}\}$ and sends *unassigned* $(latest\_unassigned)$. Upon receipt of l *ack* messages $(acks_j)$, the leader if $(\bigcup_j acks_j) \backslash known\_ids \neq \emptyset$ sets $known\_ids \leftarrow known\_ids \cup (\bigcup_j acks_j)$, $latest\_new \leftarrow current\_round$ and $time\_bound \leftarrow current\_round + |known\_ids|$ and upon receipt of l *unassigned* messages (val_j), it sets $latest\_unassigned \leftarrow \max\{latest\_unassigned, \max_j\{val_j\}\}$. If, at some round r, it holds at the leader that $r > time\_bound$ and $latest\_unassigned < latest\_new$, the leader sends a *halt* $(|known\_ids|)$ message for $|known\_ids| - 1$ rounds and then outputs id and halts. Any node that receives a *halt* (n) message, sends *halt* (n) for $n - 2$ rounds and then outputs id and halts.

Denote by $S(r) = \{v \in V : (l, 0) \rightsquigarrow (v, r)\}$ the set of nodes that have obtained an id at round r and by $K(r)$ those nodes in $S(r)$ whose id is known by the leader at round r, that is $K(r) = \{u \in V : \exists r' \text{ s.t. } u \in S(r') \text{ and } (u, r') \rightsquigarrow (l, r)\}$.

Theorem 6 ([MCS13]). *Dynamic_Naming solves the naming problem in anonymous unknown dynamic networks under the assumptions of one-to-each message transmission and of a unique leader. All nodes terminate in $O(n)$ rounds and use messages of size $\Theta(n^2)$.*

Proof. Unique names are guaranteed as in *Delegate*. Termination is as follows. Clearly, if $V \backslash K(r) \neq \emptyset$, either $|K(r + |K(r)|)| \geq |K(r)| + 1$ or $(u, r) \rightsquigarrow (l, r + |K(r)|)$ for some $u \in V \backslash S(r)$. The former is recognized by the leader by the arrival of a new id and the latter by the arrival of an *unassigned* $(timestamp)$ message, where $timestamp \geq r$. On the other hand, if $K(r) = V$ then $|K(r + |K(r)|)| = |K(r)|$ and $\nexists u \in V \backslash S(r)$ s.t. $(u, r) \rightsquigarrow (l, r + |K(r)|)$ as $V \backslash S(r) = \emptyset$. Finally, note that connectivity implies that $|S(r + 1)| \geq \min\{|S(r)| + 1, n\}$ which in turn implies $O(n)$ rounds until unique ids are assigned. Then another $O(n)$ rounds are required until nodes terminate. □

Clearly, by executing a simple $O(n)$-time process after *Dynamic_Naming* we can easily reassign minimal (consecutive) names to the nodes. The leader just floods a list of (old_id, new_id) pairs, one for each node in the network.

4th Version - Protocol *Individual_Conversations* (logarithmic messages) Though *Dynamic_Naming* is a correct and time-efficient terminating protocol for the naming problem it still has an important drawback. The messages sent may be of size $\Omega(n^2)$. We now refine *Dynamic_Naming* to arrive at a more involved construction that reduces the message size to $\Theta(\log n)$ by paying a small increase in termination time. We call this 4th version of our naming protocols *Individual_Conversations*. We only give that main idea here.

Protocol *Individual_Conversations* [Main Idea]. To reduce the size of the messages (i) the assigned names are now of the form $k \cdot d + id$, where id is the id of the node, d is the number of *unique consecutive* ids that the leader knows so far, and $k \geq 1$ is a name counter (ii) Any time that the leader wants to communicate to a remote node that has a unique id it sends a message with the id of that node and a timestamp equal to the current round. The timestamp allows all nodes to prefer this message from previous ones so that the gain is twofold: the message is delivered and no node ever issues a message containing more than one id. The remote node then can reply in the same way. For the assignment formula to work, nodes that obtain ids are not allowed to further assign ids until the leader freezes all named nodes and reassigns to them unique consecutive ids. During freezing, the leader is informed of any new assignments by the named nodes and terminates if all report that no further assignments were performed.

Theorem 7 ([MCS13]). *Individual_Conversations solves the (minimal) naming problem in* $O(n^3)$ *rounds using messages of size* $\Theta(\log n)$.

Proof. Though *Dynamic_Naming* is a correct and time-efficient terminating protocol for the naming problem it still has an important drawback. The messages sent may be of size $\Omega(n^2)$. There are two reasons for this increased message size. One is the method of assigning ids, in which the id of a node is essentially set to a pair containing the id of its fisrt parent and a counter. By induction on assignments, in which the leader assigns to a single node, that node assigns to another node, the third node to a fourth one, and so on, it is easy to see that ids may become n-tuples and thus have size $O(n)$. The other reason is that, for a node to acknowledge to the leader its assigned id, that node and all nodes that receive it must continuously broadcast it until the leader receives it (otherwise, delivery is not guaranteed by our dynamic network model). As $O(n)$ nodes may want to acknowledge at the same time, it follows that some node may need to continuously broadcast $O(n)$ ids each of size $O(n)$, thus $O(n^2)$. We now refine *Dynamic_Naming* to arrive at a more involved construction that reduces the message size to $\Theta(\log n)$ by paying a small increase in termination time. We call this protocol *Individual_Conversations*. Due to the many low-level details of the protocol we adopt a high-level but at the same time precise and clear verbal presentation.

One refinement concerns the method of assigning ids. We notice that if some d nodes have the unique consecutive ids $D = \{0, 1, 2, \ldots, k-1\}$, then we can have node with id $j \in D$ assign ids $k \cdot d + j$, for all $k \geq 1$. For example, if we have nodes $\{0, 1, 2, 3\}$, then node 0 assigns ids $\{4, 8, 12, \ldots\}$, node 1 assigns $\{5, 9, 13, \ldots\}$, node 2 assigns $\{6, 10, 14, \ldots\}$, and node 3 assigns $\{7, 11, 15, \ldots\}$. Clearly, the assignments are unique and in the worst case $k, d, j = O(n)$, which implies that the maximum assigned id is $O(n^2)$ thus its binary representation is $\Theta(\log n)$. So, if we could keep the assigning nodes to have unique consecutive ids while knowing the maximum existing id (so as to evaluate the id-generation formula), we could get logarithmic ids.

Even if we could implement the above assignment method, if nodes continued to constantly forward all ids that they ever hear of then we would not do better than message sizes $O(n \log n)$ (a node forwards $O(n)$ ids each of size $O(\log n)$). Clearly, another necessary improvement is to guarantee communication between the leader and some node with unique id j that the leader knows of, i.e. a pairwise conversation. It is important that a conversation is initiated by the leader so that we do not get multiple nodes trying to initiate a conversation with the leader, as this would increase the communication complexity. The leader sends a $request(rem\_id, current\_round)$ message, where rem_id is the id of the remote node and $current\_round$ is a timestamp indicating the time in which the request for conversation was initiated. Upon receipt of a $request(r\_id, timestamp)$ message all nodes such that $id \neq r\_id$ forward the message if it is the one with the largest timestamp that they have ever heard of. All nodes keep forwarding the message with the largest timestamp. When the remote node receives the message it replies with $report(id, current\_round)$, where id is its own id. Now all nodes will forward the report as it is the one with the largest timestamp and the report will eventually reach the leader who can reply with another request, and so on. Note that a node that participates in a conversation need not know how much time it will take for the other node to reply. It only needs to have a guarantee that the reply will eventually arrive. Then it can recognize that this is the correct reply by the type, the id-component, and the timestamp of the received message. A nice property of 1-interval connected graphs is that it guarantees any such reply to arrive in $O(n)$ rounds if all nodes that receive it keep broadcasting it (which is the case here, due to the timestamps). So, in order to keep the message sizes low, we must implement the above communication method in such a way that the leader always participates in a single conversation, so that a single message ever floods the whole network (in particular, the most recently created one).

Now, let us further develop our id-assignment method. Clearly, in the 1st round the leader can keep id 0 for itself and assign the unique consecutive ids $\{1, 2, \ldots, d_l(1)\}$ to its $|d_l(1)|$ neighbors in round 1. Clearly, each node with id j in $K(1) = \{0, 1, \ldots, |d_l(1)|\}$ can further assign the unique ids $k \cdot |K(1)| + j$, for $k \geq 1$. As before we can have a node stick to the smallest id that it hears from its neighbors but we additionally need that node to remember those ids that it rejected in a *rejected* list. However, note that, if nodes outside $K(1)$ that obtain a unique id are not allowed to further assign ids, then we do not guarantee that all nodes will eventually obtain an id. The reason is that the adversary can forever hide the set $K(1)$ from the rest of the graph via nodes that have obtained an id and do not further assign ids (that is, all nodes in $K(1)$ may communicate only to nodes in $K(1)$ and to nodes that have obtained an id but do not assign and all nodes that do not have an id may communicate only to nodes that do not have an id and to nodes that have obtained an id but do not assign, which is some sort of deadlock). So we must somehow also allow to nodes that obtain ids to further assign ids. The only way to do this while keeping our assignment formula is to restructure the new assignments so that they are still unique and additionally consecutive. So, for example, if nodes in $K(1)$ have at some point assigned a set of ids T, then the leader should somehow reassign to nodes in T the ids $\{|K(1)|, |K(1)| + 1, \ldots, |K(1)| + |T| - 1\}$.

So, at this point, it must be clear that the leader must first allow to the nodes that have unique consecutive ids (including itself) to perform some assignments. Then at some point it should freeze the assigning nodes and ask them one after the other to report the assignments that they have performed so far. Then, assuming that it has learned all the newly assigned unique ids, it should communicate with that nodes to reassign to them the next available unique consecutive ids and also it should inform all nodes with id of the maximum consecutive id that has been assigned so far. Now that all nodes with id have unique consecutive ids and know the maximum assigned, they can all safely use the id-assignment formula. In this manner, we have managed to also allow to the new nodes to safely assign unique ids. Finally, the leader unfreezes the nodes with ids one after the other, alows them to assign some new ids and at some point freezes them again to repeat the above process which we may call a cycle.

A very important point that we should make clear at this point is that, in 1-interval connected graphs, a new assignment is only guaranteed if at least for one round all nodes that have ids send assignment messages to all their incident edges. As if some node with id selected to issue no assignment message to some of its edges then the adversary could make that edge be the only edge that connects nodes with ids to nodes without ids and it could do the same any time an edge is not used. Forunately, this is trivially guaranteed in the solution we have develped so far. When the leader unfreezes the last node with id, even if it chooses to start freezing the nodes in the subsequent round, provided that at least for that round it does not freezes itself, then in that round all nodes including itself are not frozen, thus all take an assignment step in that round (sending assignment messages to all their incident edges). This guarantees that for at least one round all assign at the same time which in turn guarantees at least one new delivery, provided that there are still nodes without ids.

Another point that is still blur is the following. When the leader gets all reports from all nodes that were assigning ids during this cycle it cannot know which ids have been assigned but only which ids have been possibly assigned. The reason is that when a node u assigns some ids then it is not guaranteed that in the next round it will have the same neighbors. So it can be the case that some of its neighbors chooses to stick to a smaller id sent by some other node and u never notices it. So we have each node that assigns ids to remember the ids that have possibly been assigned and each node that is assigned an id to remember those ids that it rejected. Note that when a node u tries to assigns an id by sending it via a local edge, then, in the next round when it receives from that local edge, it can tell whether that id was possibly assigned by simply having all nodes send their id in every round. If the received id from that edge was $\perp$ then the corresponding neighbor did not have an id thus it must have been assigned some id even if that was not the one sent by u. In any case, the id sent by u will either be assigned or stored in the *rejected* list of that node. On the other hand, if the received id was not equal to $\perp$ then the neighbor already had an id, u knows that its assignment was for sure unsuccessful and may reuse this id in future assignments. The problem now is that, if the leader tries to initiate a conversation with an arbitrary id from those that have been possibly assigned, it can very well be the case that this id was not assigned and the leader may have to wait for a reply forever. Fortunately, this can be solved easily by having the unique node that has stored this id in its *rejected* list to reply not only when it gets a *request* message containing its own id but also when it gets a message containing an id that is also in its *rejected* list. Another way is the following. As the leader has first collected all possibly delivered ids, it can order them increasingly and start seeking that smallest id. As nodes stick to the smallest they hear, the smallest of all possibly assigned was for sure selected by some node. Then that node may inform the leader of some rejected ids which the leader will remove from its ordering and then the leader may proceed to seek for the next id that has remained in its ordered list. It is not hard to see that this method guarantees that the leader always seeks for existing ids.

Finally, the termination criterion is more or less the same as in $Dynamic\_Naming$. The leader knows that, if it allows all nodes with ids a common assignment step, then, provided that there are nodes without ids, at least one new assignment must take place. Clearly, if all nodes report that they performed no assignments, then the leader can terminate (and tell others to terminate) knowing that all nodes must have obtained an id. In the termination phase, it can reassign for a last time unique consecutive ids and inform all nodes of n. □

4.5 Higher Dynamicity

Given some high-dynamicity assumption (some sort of fairness), naming can be solved under broadcast communication. Intuitively, to break the symmetry that is responsible for the impossibility of Conjecture 1, we require that, given sufficient time, a node has influenced every other node in different rounds. Formally, there must exist k (not necessarily known to the nodes) s.t $(\text{arrival}_{(u,r)}(v), \text{arrival}_{(u,r+1)}(v), \ldots, \text{arrival}_{(u,r+k-1)}(v)) \neq (\text{arrival}_{(u,r)}(w), \text{arrival}_{(u,r+1)}(w), \ldots, \text{arrival}_{(u,r+k-1)}(w))$, $\forall u \in V, r \geq 0, v, w \in V \backslash \{u\}$, where $\text{arrival}_{(u,r)}(v)$

$:= \min\{r' > r : (u,r) \rightsquigarrow (v,r')\}$ (first time that v is causally influenced by the r-state of u). We also allow nodes to have time to acknowledge to their neighbors (formally, we may duplicate each instance of the dynamic graph, i.e. make it persist for two rounds).

The idea is to have the leader name its first $d_l(1)$ neighbors say with id 1. What the leader can exploit is that it knows the number of 1s in the network as it knows its degree in round 1. Now every node v named 1 counts $\text{arrival}_{(l,i)}(v)$ for all $i \geq 2$. This is achieved by having the leader continuously send an $(l, \mathit{current\_round})$ pair, unnamed nodes constantly forward it, and having every node named 1 set $\text{arrival}_{(l,i)}(v)$ to the round in which an (l,i) pair was first delivered. It is clear that, due to the above high-dynamicity assumption, the vector $s(v) = (1, \text{arrival}_{(l,2)}(v), \text{arrival}_{(u,3)}(v), \ldots, \text{arrival}_{(u,k+2)}(v))$ (in k rounds) will be a unique *id*. As the named nodes do not know k, we have them continuously send $(s, \mathit{current\_round})$ pairs, where s is the above vector, and all other nodes continuously forward these pairs. At some point, the leader must hear from $d_l(1)$ different s vectors with equal timestamps and then it knows that the 1s have obtained unique ids. Now the leader can stop them from further changing their ids. Then it allows them (including itself) to concurrently assign id 2 for at least one step. Assigning nodes count the number of assignments that they perform (in a variable *count* initially 0). This is done by having a node u that was assigned id 2 in round r to respond to its neighbors the number l of nodes that tried to assigned 2 to it. Then each of the assigning 1s sets $\mathit{count} \leftarrow \mathit{count} + 1/l$. When the leader freezes the 1s, they report their *count* variable and by summing them the leader learns the number, j, of 2s assigned. Then the leader sends again $(l, \mathit{current\_round})$ pairs and waits to receive j different s vectors with equal timestamps. The process continues in such cycles until at some point all existing unique ids report that they didn't manage to assign the current id being assigned.

5 Causality, Influence, and Computation in Possibly Disconnected Synchronous Dynamic Networks

In this section, we study the *propagation of influence and computation in dynamic distributed computing systems that are possibly disconnected at every instant*. We focus on a *synchronous message passing* communication model with *broadcast* and bidirectional links. Our network dynamicity assumption is again a *worst-case dynamicity* controlled by an adversary scheduler. However, we replace the usual (in worst-case dynamic networks) assumption that the network is connected at every instant by minimal *temporal connectivity* conditions. Our conditions only require that *another causal influence occurs within every time-window of some given length*. Based on this basic idea we define several novel metrics for capturing the speed of information spreading in a dynamic network. Moreover, we investigate *termination criteria* in networks in which an upper bound on any of these metrics is known. We exploit our termination criteria to give protocols that solve the fundamental *counting* and *all-to-all token dissemination* (or *gossip*) problems.

5.1 Our Metrics

As already stated, in this section we aim to deal with dynamic networks that are allowed to have disconnected instances. To this end, we define some novel generic metrics that are particularly suitable for capturing the speed of information propagation in such networks.

5.1.1 The Influence Time Recall that the guarantee on propagation of information resulting from instantaneous connectivity ensures that any time-node (u,t) influences another node *in each* *step* (if an uninfluenced one exists). From this fact, we extract two novel generic influence metrics that capture the maximal time until another influence (outgoing or incoming) of a time-node occurs.

We now formalize our first influence metric.

Definition 2 (Outgoing Influence Time). *We define the* outgoing influence time *(oit) as the minimum $k \in \mathbb{N}$ s.t. for all $u \in V$ and all times $t, t' \geq 0$ s.t. $t' \geq t$ it holds that*

$$|\text{future}_{(u,t)}(t'+k)| \geq \min\{|\text{future}_{(u,t)}(t')| + 1, n\}.$$

Intuitively, the oit is the maximal time until the t-state of a node influences the state of another node (if an uninfluenced one exists) and captures the speed of information spreading.

Our second metric is similarly defined as follows.

Definition 3 (Incoming Influence Time). *We define the* incoming influence time *(iit) as the minimum $k \in \mathbb{N}$ s.t. for all $u \in V$ and all times $t, t' \geq 0$ s.t. $t' \geq t$ it holds that*

$$|\text{past}_{(u,t'+k)}(t)| \geq \min\{|\text{past}_{(u,t')}(t)| + 1, n\}.$$

We can now say that the oit of a T-interval connected graph is 1 and that the iit can be up to $n-2$. However, is it necessary for a dynamic graph to be T-interval connected in order to achieve unit oit? First, let us make a simple but useful observation:

Proposition 1 ([MCS14]). *If a dynamic graph $G = (V, E)$ has oit (or iit) 1 then every instance has at least $\lceil n/2 \rceil$ edges.*

Proof. $\forall u \in V$ and $\forall t \geq 1$ it must hold that $\{u, v\} \in E(t)$ for some v. In words, at any time t each node must have at least one neighbor since otherwise it influences (or is influenced by) no node during round t. A minimal way to achieve this is by a perfect matching in the even-order case and by a matching between $n-3$ nodes and a linear graph between the remaining 3 nodes in the odd-order case.

Proposition 1 is easily generalized as: if a dynamic graph $G = (V, E)$ has oit (or iit) k then for all times t it holds that $|\bigcup_{i=t}^{t+k-1} E(i)| \geq \lceil n/2 \rceil$. The reason is that now any node must have a neighbor in any k-window of the dynamic graph (and not necessarily in every round).

Now, inspired by Proposition 1, we define a minimal dynamic graph that at the same time satisfies oit 1 and always disconnected instances:

The Alternating Matchings Dynamic Graph. Take a ring of an even number of nodes $n = 2l$, partition the edges into 2 disjoint perfect matchings A and B (each consisting of l edges) and alternate round after round between the edge sets A and B (see Figure 1).

Proposition 2 ([MCS14]). *The Alternating Matchings dynamic graph has oit 1 and any node needs precisely $n/2$ rounds to influence all other nodes.*

Proof. Take any node u. In the first round, u influences its left or its right neighbor on the ring depending on which of its two adjacent edges becomes available first. Thus, including itself, it has influenced 2 nodes forming a line of length 1. In the next round, the two edges that join the endpoints of the line with the rest of the ring become available and two more nodes become influenced; the one is the neighbor on the left of the line and the other is the neighbor on the right. By induction on the number of rounds, it is not hard to see that the existing line always expands from its endpoints to the two neighboring nodes of the ring (one on the left and the other on the right). Thus, we get exactly 2 new influences per round, which gives oit 1 and $n/2$ rounds to influence all nodes.

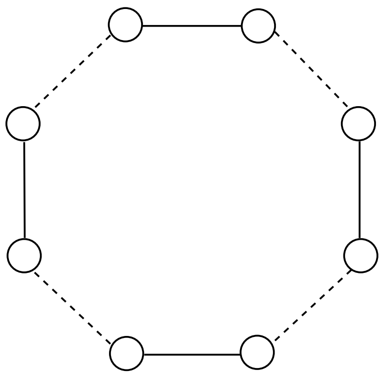

Fig. 1. The Alternating Matchings dynamic graph for $n = 8$. The solid lines appear every odd round $(1, 3, 5, \ldots)$ while the dashed lines every even round $(2, 4, 6, \ldots)$.

In the alternating matchings construction any edge reappears every second step but not faster than this. We now formalize the notion of the *fastest edge reappearance* (fer) of a dynamic graph.

Definition 4. *The* fastest edge reappearance *(fer) of a dynamic graph $G = (V, E)$ is defined as the minimum $p \in \mathbb{N}$ s.t., $\exists e \in \{\{u, v\} : u, v \in V\}$ and $\exists t \in \mathbb{N}$, $e \in E(t) \cap E(t + p)$.*

Clearly, the fer of the alternating matchings dynamic graph described above is 2, because no edge ever reappears in 1 step and some, at some point, (in fact, all and always) reappears in 2 steps. In Section 5.2, by invoking a geometric edge-coloring method, we generalize this basic construction to a more involved dynamic graph with oit 1, always disconnected instances, and fer equal to $n - 1$. [7]

We next give a proposition associating dynamic graphs with oit (or iit) upper bounded by K to dynamic graphs with connected instances.

Proposition 3 ([MCS14]). *Assume that the oit or the iit of a dynamic graph, $G = (V, E)$, is upper bounded by K. Then for all times $t \in \mathbb{N}$ the graph $(V, \bigcup_{i=t}^{t+K\lfloor n/2\rfloor-1} E(i))$ is connected.*

Proof. It suffices to show that for any partitioning (V_1, V_2) of V there is an edge in the cut labeled from $\{t, \ldots, t + K\lfloor n/2\rfloor - 1\}$. W.l.o.g. let V_1 be the smaller one, thus $|V_1| \leq \lfloor n/2\rfloor$. Take any $u \in V_1$. By definition of oit, $|\mathrm{future}_{(u,t)}(t + K\lfloor n/2\rfloor - 1)| \geq |\mathrm{future}_{(u,t)}(t + K|V_1| - 1)| \geq |V_1| + 1$ implying that some edge in the cut has transferred u's t-state out of V_1 at some time in the interval $[t, t + K\lfloor n/2\rfloor - 1]$. The proof for the iit is similar.

5.1.2 The Moi (Concurrent Progress) Consider now the following influence metric:

Definition 5. *Define the* maximum outgoing influence *(moi) of a dynamic graph $G = (V, E)$ as the maximum k for which $\exists u \in V$ and $\exists t, t' \in \mathbb{N}$, $t' \geq t$, s.t. $|\mathrm{future}_{(u,t)}(t'+1)| - |\mathrm{future}_{(u,t)}(t')| = k$.*

[7] It is interesting to note that in dynamic graphs with a static set of nodes (that is V does not change), if at least one change happens each time, then every instance $G(t)$ will eventually reappear after at most $\sum_{k=0}^{\binom{n}{2}} \binom{\binom{n}{2}}{k}$ steps. This counts all possible different graphs of n vertices with k edges and sums for all $k \geq 0$. Thus the fer is bounded from above by a function of n.

In words, the moi of a dynamic graph is the maximum number of nodes that are ever concurrently influenced by a time-node.

Here we show that one cannot guarantee at the same time unit oit and at most one outgoing influence per node per step. In fact, we conjecture that unit oit implies that some node disseminates in $\lfloor n/2 \rfloor$ steps.

We now prove an interesting theorem stating that if one tries to guarantee unit oit then she must necessarily accept that at some steps more than one outgoing influences of the same time-node will occur leading to faster dissemination than $n-1$ for this particular node.

Theorem 8 ([MCS14]). *The moi of any dynamic graph with $n \geq 3$ and unit oit is at least 2.*

Proof. For $n = 3$, just notice that unit oit implies that, at any time t, some node has necessarily 2 neighbors. We therefore focus on $n \geq 4$. For the sake of contradiction, assume that the statement is not true. Then at any time t any node u is connected to exactly one other node v (at least one neighbor is required for oit 1 - see Proposition 1 - and at most one is implied by our assumption). Unit oit implies that, at time $t+1$, at least one of u, v must be connected to some $w \in V \backslash \{u, v\}$, let it be v. Proposition 1 requires that also u must have an edge labeled $t+1$ incident to it. If that edge arrives at v, then v has 2 edges labeled $t+1$. If it arrives at w, then w has 2 edges labeled $t+1$. So it must arrive at some $z \in V \backslash \{u, v, w\}$. Note now that, in this case, the $(t-1)$-state of u first influences both w, z at time $t+1$ which is contradictory, consequently the moi must be at least 2.

In fact, notice that the above theorem proves something stronger: Every second step at least half of the nodes influence at least 2 new nodes each. This, together with the fact that it seems to hold for some basic cases, makes us suspect that the following conjecture might be true:

Conjecture 2 ([MCS14]). If the oit of a dynamic graph is 1 then $\forall t \in \mathbb{N}$, $\exists u \in V$ s.t. $|\text{future}_{(u,t)}(t + \lfloor n/2 \rfloor)| = n$.

That is, if the oit is 1 then, in every $\lfloor n/2 \rfloor$-window, some node influences all other nodes (e.g. influencing 2 new nodes per step).

5.1.3 The Connectivity Time

We now propose another natural and practical metric for capturing the temporal connectivity of a possibly disconnected dynamic network that we call the *connectivity time* (ct).

Definition 6 (Connectivity Time). *We define the* connectivity time *(ct) of a dynamic network $G = (V, E)$ as the minimum $k \in \mathbb{N}$ s.t. for all times $t \in \mathbb{N}$ the static graph $(V, \bigcup_{i=t}^{t+k-1} E(i))$ is connected.*

In words, the ct of a dynamic network is the maximal time of keeping the two parts of any cut of the network disconnected. That is to say, in every ct-window of the network an edge appears in every (V_1, V_2)-cut. Note that, in the extreme case in which the ct is 1, every instance of the dynamic graph is connected and we thus obtain a 1-interval connected graph. On the other hand, greater ct allows for different cuts to be connected at different times in the ct-round interval and the resulting dynamic graph can very well have disconnected instances. For an illustrating example, consider again the alternating matchings graph from Section 5.1.1. Draw a line that crosses two edges belonging to matching A partitioning the ring into two parts. Clearly, these two parts communicate every second round (as they only communicate when matching A becomes available), thus the ct is 2 and every instance is disconnected. We now provide a result associating the ct of a dynamic graph with its oit.

Proposition 4 ([MCS14]). *(i) $\mathsf{oit} \leq \mathsf{ct}$ but (ii) there is a dynamic graph with oit 1 and $\mathsf{ct} = \Omega(n)$.*

Proof. (i) We show that for all $u \in V$ and all times $t, t' \in \mathbb{N}$ s.t. $t' \geq t$ it holds that $|\text{future}_{(u,t)}(t' + \mathsf{ct})| \geq \min\{|\text{future}_{(u,t)}(t')| + 1, n\}$. Assume $V \backslash \text{future}_{(u,t)}(t') \neq \emptyset$ (as the other case is trivial). In at most ct rounds at least one edge joins $\text{future}_{(u,t)}(t')$ to $V \backslash \text{future}_{(u,t)}(t')$. Thus, in at most ct rounds $\text{future}_{(u,t)}(t')$ increases by at least one.

(ii) Recall the alternating matchings on a ring dynamic graph from Section 5.1.1. Now take any set V of a number of nodes that is a multiple of 4 (this is just for simplicity and is not necessary) and partition it into two sets V_1, V_2 s.t. $|V_1| = |V_2| = n/2$. If each part is an alternating matchings graph for $|V_1|/2$ rounds then every u say in V_1 influences 2 new nodes in each round and similarly for V_2. Clearly we can keep V_1 disconnected from V_2 for $n/4$ rounds without violating $\mathsf{oit} = 1$.

The following is a comparison of the ct of a dynamic graph with its dynamic diameter D.

Proposition 5 ([MCS14]). *$\mathsf{ct} \leq D \leq (n-1)\mathsf{ct}$.*

Proof. $\mathsf{ct} \leq D$ follows from the fact that in time equal to the dynamic diameter every node causally influences every other node and thus, in that time, there must have been an edge in every cut (if not, then the two partitions forming the cut could not have communicated with one another). $D \leq (n-1)\mathsf{ct}$ holds as follows. Take any node u and add it to a set S. In ct rounds u influences some node from $V \backslash S$ which is then added to S. In $(n-1)\mathsf{ct}$ rounds S must have become equal to V, thus this amount of time is sufficient for every node to influence every other node. Finally, we point out that these bounds cannot be improved in general as for each of $\mathsf{ct} = D$ and $D = (n-1)\mathsf{ct}$ there is a dynamic graph realizing it. $\mathsf{ct} = D$ is given by the dynamic graph that has no edge for $\mathsf{ct} - 1$ rounds and then becomes the complete graph while $D = (n-1)\mathsf{ct}$ is given by a line in which every edge appears at times $\mathsf{ct}, 2\mathsf{ct}, 3\mathsf{ct}, \ldots$.

Note that the ct metric has been defined as an underapproximation of the dynamic diameter. Its main advantage is that it is much easier to compute than the dynamic diameter since it is defined on the union of the footprints and not on the dynamic adjacency itself.

5.2 Fast Propagation of Information Under Continuous Disconnectivity

In Section 5.1.1, we presented a simple example of an always-disconnected dynamic graph, namely, the alternating matchings dynamic graph, with optimal oit (i.e. unit oit). Note that the alternating matchings dynamic graph may be conceived as simple as it has small fer (equal to 2). We pose now an interesting question: Is there an always-disconnected dynamic graph with unit oit and fer as big as $n-1$? Note that this is harder to achieve as it allows of no edge to ever reappear in less than $n-1$ steps. Here, by invoking a geometric edge-coloring method, we arrive at an always-disconnected graph with unit oit and maximal fer; in particular, no edge reappears in less than $n-1$ steps.

To answer the above question, we define a very useful dynamic graph coming from the area of edge-coloring.

Definition 7. *We define the following dynamic graph S based on an edge-coloring method due to Soifer [Soi09]: $V(S) = \{u_1, u_2, \ldots, u_n\}$ where $n = 2l$, $l \geq 2$. Place u_n on the center and $u_1, \ldots, u_{n-1}$ on the vertices of a $(n-1)$-sided polygon. For each time $t \geq 1$ make available only the edges $\{u_n, u_{m_t(0)}\}$ for $m_t(j) := (t - 1 + j \bmod n - 1) + 1$ and $\{u_{m_t(-i)}, u_{m_t(i)}\}$ for $i = 1, \ldots, n/2 - 1$; that is make available one edge joining the center to a polygon-vertex and all edges perpendicular to it. (e.g. see Figure 2 for $n = 8$ and $t = 1, \ldots, 7$).*

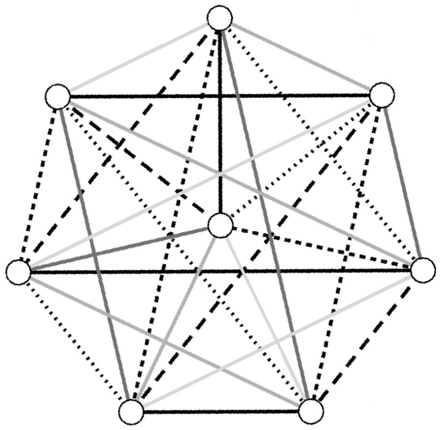

Fig. 2. Soifer's dynamic graph for $n = 8$ and $t = 1, \ldots, 7$. In particular, in round 1 the graph consists of the black solid edges, then in round 2 the center becomes connected via a dotted edge to the next peripheral node clockwise and all edges perpendicular to it (the remaining dotted ones) become available, and so on, always moving clockwise.

In Soifer's dynamic graph, denote by $N_u(t) := i : \{u, u_i\} \in E(t)$, that is the index of the unique neighbor of u at time t. The following lemma states that the next neighbor of a node is in almost all cases (apart from some trivial ones) the one that lies two positions clockwise from its current neighbor.

Lemma 4 ([MCS14]). *For all times $t \in \{1, 2, \ldots, n-2\}$ and all u_k, $k \in \{1, 2, \ldots, n-1\}$ it holds that $N_{u_k}(t+1) = n$ if $N_{u_k}(t) = (k - 3 \bmod n - 1) + 1$ else $N_{u_k}(t+1) = (k + 1 \bmod n - 1) + 1$ if $N_{u_k}(t) = n$ and $N_{u_k}(t+1) = (N_{u_k}(t) + 1 \bmod n - 1) + 1$ otherwise.*

Proof. Since $k \notin \{n, t, t+1\}$ it easily follows that $k, N_k(t), N_k(t+1) \neq n$ thus both $N_k(t)$ and $N_k(t+1)$ are determined by $\{u_{m_t(-i)}, u_{m_t(i)}\}$ where $m_t(j) := (t - 1 + j \bmod n - 1) + 1$ and $k = m_t(-i)$. The latter implies $(t-1-i \bmod n-1)+1 = k \Rightarrow (t-1+i \bmod n-1)+1+(-2i \bmod n-1) = k \Rightarrow m_t(i) = k - (-2i \bmod n - 1)$; thus, $N_k(t) = k - (-2i \bmod n - 1)$. Now let us see how the i that corresponds to some node changes as t increases. When t increases by 1, we have that $(t - 1 + i \bmod n - 1) + 1 = (t + i' \bmod n - 1) + 1 \Rightarrow i' = i - 1$, i.e. as t increases i decreases. Consequently, for $t+1$ we have $N_k(t+1) = k - [-2(i-1) \bmod n-1] = (N_{u_k}(t)+1 \bmod n - 1)+1$.

Theorem 9 ([MCS14]). *For all $n = 2l$, $l \geq 2$, there is a dynamic graph of order n, with* oit *equal to 1,* fer *equal to $n - 1$, and in which every instance is a perfect matching.*

Proof. The dynamic graph is the one of Definition 7. It is straightforward to observe that every instance is a perfect matching. We prove now that the oit of this dynamic graph is 1. We focus on the set $\mathrm{future}_{(u_n,0)}(t)$, that is the outgoing influence of the initial state of the node at the center. Note that symmetry guarantees that the same holds for all time-nodes (it can be verified that any node can be moved to the center without altering the graph). u_n at time 1 meets u_1 and thus $\mathrm{future}_{(u_n,0)}(1) = \{u_1\}$. Then, at time 2, u_n meets u_2 and, by Lemma 4, u_1 meets u_3 via the edge than is perpendicular to $\{u_n, u_2\}$, thus $\mathrm{future}_{(u_n,0)}(2) = \{u_1, u_2, u_3\}$. We show that for all times t it holds that $\mathrm{future}_{(u_n,0)}(t) = \{u_1, \ldots, u_{2t-1}\}$. The base case is true since $\mathrm{future}_{(u_n,0)}(1) = \{u_1\}$. It is not hard to see that, for $t \geq 2$, $N_{u_2}(t) = 2t-2$, $N_{u_1}(t) = 2t-1$, and for all $u_i \in \mathrm{future}_{(u_n,0)}(t)\backslash\{u_1, u_2\}$, $1 \leq N_{u_i}(t) \leq 2t-2$. Now consider time $t+1$. Lemma 4 guarantees now that for all $u_i \in \mathrm{future}_{(u_n,0)}(t)$ we have that $N_{u_i}(t+1) = N_{u_i}(t) + 2$. Thus, the only new influences at step $t+1$ are by u_1 and u_2 implying that $\mathrm{future}_{(u_n,0)}(t+1) = \{u_1, \ldots, u_{2(t+1)-1}\}$. Consequently, the oit is 1.

The fer is $n-1$ because the edges leaving the center appear one after the other in a clockwise fashion, thus taking $n-1$ steps to any such edge to reappear, and, by construction, any other edge appears only when its unique perpendicular that is incident to the center appears (thus, again every $n-1$ steps).

Note that Theorem 9 is optimal w.r.t. fer as it is impossible to achieve at the same time unit oit and fer strictly greater than $n-1$. To see this, notice that if no edge is allowed to reappear in less than n steps then any node must have no neighbors once every n steps.

5.3 Termination and Computation

We now turn our attention to termination criteria that we exploit to solve the fundamental counting and all-to-all token dissemination problems. First observe that if nodes know an upper bound H on the iit then there is a straightforward optimal termination criterion taking time $D+H$, where D is the dynamic diameter. In every round, all nodes forward all ids that they have heard of so far. If a node does not hear of a new id for H rounds then it must have already heard from all nodes. Keep in mind that nodes have no *a priori* knowledge of the size of the network.

5.3.1 Nodes Know an Upper Bound on the ct: An Optimal Termination Criterion We here assume that all nodes know some upper bound T on the ct. We will give an optimal condition that allows a node to determine whether it has heard from all nodes in the graph. This condition results in an algorithm for counting and all-to-all token dissemination which is optimal, requiring $D+T$ rounds in any dynamic network with dynamic diameter D. The core idea is to have each node keep track of its past sets from time 0 and from time T and terminate as soon as these two sets become equal. This technique is inspired by [KMO11], where a comparison between the past sets from time 0 and time 1 was used to obtain an optimal termination criterion in 1-interval connected networks.

Theorem 10 ([MCS14]). *[Repeated Past] Node u knows at time t that* $\mathrm{past}_{(u,t)}(0) = V$ *iff* $\mathrm{past}_{(u,t)}(0) = \mathrm{past}_{(u,t)}(T)$.

Proof. If $\mathrm{past}_{(u,t)}(0) = \mathrm{past}_{(u,t)}(T)$ then we have that $\mathrm{past}_{(u,t)}(T) = V$. The reason is that $|\mathrm{past}_{(u,t)}(0)| \geq \min\{|\mathrm{past}_{(u,t)}(T)| + 1, n\}$. To see this, assume that $V\backslash\mathrm{past}_{(u,t)}(T) \neq \emptyset$. At most by round T there is some edge joining some $w \in V\backslash\mathrm{past}_{(u,t)}(T)$ to some $v \in \mathrm{past}_{(u,t)}(T)$. Thus, $(w,0) \rightsquigarrow (v,T) \rightsquigarrow (u,t) \Rightarrow w \in \mathrm{past}_{(u,t)}(0)$. In words, all nodes in $\mathrm{past}_{(u,t)}(T)$ belong to $\mathrm{past}_{(u,t)}(0)$ and at least one node not in $\mathrm{past}_{(u,t)}(T)$ (if one exists) must belong to $\mathrm{past}_{(u,t)}(0)$ (see also Figure 3).

For the other direction, assume that there exists $v \in \text{past}_{(u,t)}(0) \backslash \text{past}_{(u,t)}(T)$. This does not imply that $\text{past}_{(u,t)}(0) \neq V$ but it does imply that even if $\text{past}_{(u,t)}(0) = V$ node u cannot know it has heard from everyone. Note that u heard from v at some time $T' < T$ but has not heard from v since then. It can be the case that arbitrarily many nodes were connected to no node until time $T-1$ and from time T onwards were connected only to node v (v in some sense conceals these nodes from u). As u has not heard from the T-state of v it can be the case that it has not heard at all from arbitrarily many nodes, thus it cannot decide on the count.

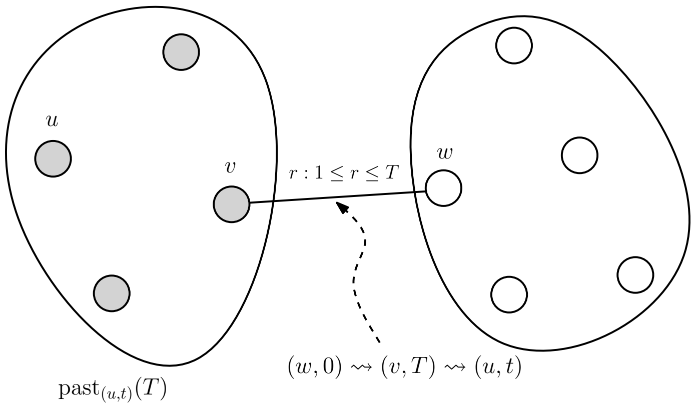

Fig. 3. A partitioning of V into two sets. The left set is $\text{past}_{(u,t)}(T)$, i.e. the set of nodes whose T-state has influenced u by time t. All nodes in $\text{past}_{(u,t)}(T)$ also belong to $\text{past}_{(u,t)}(0)$. Looking back in time at the interval $[1, T]$, there should be an edge from some v in the left set to some w in the right set. This implies that v has heard from w by time T and as u has heard from the T-state of v it has also heard from the initial state of w. This implies that $\text{past}_{(u,t)}(0)$ is a strict superset of $\text{past}_{(u,t)}(T)$ as long as the right set is not empty.

We now give a time-optimal algorithm for counting and all-to-all token dissemination that is based on Theorem 10.

Protocol *A*. All nodes constantly forward all 0-states and T-states of nodes that they have heard of so far (in this protocol, these are just the ids of the nodes accompanied with 0 and T timestamps, respectively) and a node halts as soon as $\text{past}_{(u,t)}(0) = \text{past}_{(u,t)}(T)$ and outputs $|\text{past}_{(u,t)}(0)|$ for counting or $\text{past}_{(u,t)}(0)$ for all-to-all dissemination.

For the time-complexity notice that any state of a node needs D rounds to causally influence all nodes, where D is the dynamic diameter. Clearly, by time $D + T$, u must have heard of the 0-state and T-state of all nodes, and at that time $\text{past}_{(u,t)}(0) = \text{past}_{(u,t)}(T)$ is satisfied. It follows that all nodes terminate in at most $D + T$ rounds. Optimality follows from the fact that this protocol terminates as long as $\text{past}_{(u,t)}(0) = \text{past}_{(u,t)}(T)$ which by the "only if" part of the statement of Theorem 10 is a necessary condition for correctness (any protocol terminating before this may terminate without having heard from all nodes).

5.3.2 Known Upper Bound on the oit Now we assume that all nodes know some upper bound K on the oit. Then one can show that if a node u has at some point heard of l nodes, then u hears of another node in $O(Kl^2)$ rounds (if an unknown one exists).

Theorem 11 ([MCS14]). *In any given dynamic graph with* oit *upper bounded by K, take a node u and a time t and denote $|\text{past}_{(u,t)}(0)|$ by l. It holds that $|\{v : (v, 0) \rightsquigarrow (u, t + Kl(l + 1)/2)\}| \geq \min\{l + 1, n\}$.*

Proof. Consider a node u and a time t and define $A_u(t) := \text{past}_{(u,t)}(0)$ (we only prove it for the initial states of nodes but easily generalizes to any time), $I_u(t') := \{v \in A_u(t) : A_v(t')\backslash A_u(t) \neq \emptyset\}$, $t' \geq t$, that is $I_u(t')$ contains all nodes in $A_u(t)$ whose t'-states have been influence by nodes not in $A_u(t)$ (these nodes know new info for u), $B_u(t') := A_u(t)\backslash I_u(t')$, that is all nodes in $A_u(t)$ that do not know new info, and $l := |A_u(t)|$. The only interesting case is for $V\backslash A_u(t) \neq \emptyset$. Since the oit is at most K we have that at most by round $t+Kl$, (u, t) influences some node in $V\backslash B_u(t)$ say via some $u_2 \in B_u(t)$. By that time, u_2 leaves B_u. Next consider $(u, t + Kl + 1)$. In $K(l - 1)$ steps it must influence some node in $V\backslash B_u$ since now u_2 is not in B_u. Thus, at most by round $t + Kl + K(l - 1)$ another node, say e.g. u_3, leaves B_u. In general, it holds that $B_u(t' + K|B_u(t')|) \leq \max\{|B_u(t')| - 1, 0\}$. It is not hard to see that at most by round $j = t + K(\sum_{1\leq i\leq l} i)$, B_u becomes empty, which by definition implies that u has been influenced by the initial state of a new node. In summary, u is influenced by another initial state in at most $K(\sum_{1\leq i\leq l} i) = kl(l + 1)/2$ steps.

The good thing about the upper bound of Theorem 11 is that it associates the time for a new incoming influence to arrive at a node only with an upper bound on the oit, which is known, and the number of existing incoming influences which is also known, and thus the bound is locally computable at any time. So, there is a straightforward translation of this bound to a termination criterion and, consequently, to an algorithm for counting and all-to-all dissemination based on it.

Protocol B. All nodes constantly broadcast all ids that they have heard of so far. Each node u keeps a set $A_u(r)$ containing the ids it knows at round r and a termination bound $H_u(r)$ initially equal to K. If, at round r, u hears of new nodes, it inserts them in $A_u(r)$ and sets $H_u(r) \leftarrow r + Kl(l+1)/2$, where $l = |A_u(r)|$. If it ever holds that $r > H_u(r)$, u halts and outputs $|A_u(r)|$ for counting or $A_u(r)$ for all-to-all dissemination.

In the worst case, u needs $O(Kn)$ rounds to hear from all nodes and then another $Kn(n+1)/2 = O(Kn^2)$ rounds to realize that it has heard from all. So, the time complexity is $O(Kn^2)$.

Note that the upper bound of Theorem 11 is loose. The reason is that if a dynamic graph has oit upper bounded by K then in $O(Kn)$ rounds all nodes have causally influenced all other nodes and clearly the iit can be at most $O(Kn)$. We now show that there is indeed a dynamic graph that achieves this worst possible gap between the iit and the oit.

Theorem 12 ([MCS14]). *There is a dynamic graph with* oit *k but* iit *$k(n - 3)$.*

Proof. Consider the dynamic graph $G = (V, E)$ s.t. $V = \{u_1, u_2, \ldots, u_n\}$ and u_i, for $i \in \{1, n - 1\}$, is connected to u_{i+1} via edges labeled jk for $j \in \mathbb{N}_{\geq 1}$, u_i, for $i \in \{2, 3, \ldots, n - 2\}$, is connected to u_{i+1} via edges labeled jk for $j \in \mathbb{N}_{\geq 2}$. and u_2 is connected to u_i, for $i \in \{3, \ldots, n - 1\}$ via edges labeled k. In words, at step k, u_1 is only connected to u_2, u_2 is connected to all nodes except from u_n and u_n is connected to u_{n-1}. Then every multiple of k there is a single linear graph starting from u_1 and ending at u_n. At step k, u_2 is influenced by the initial states of nodes $\{u_3, \ldots, u_{n-1}\}$. Then at step $2k$ it forwards these influences to u_1. Since there are no further shortcuts, u_n's state needs $k(n - 1)$ steps to arrive at u_1, thus there is an incoming-influence-gap of $k(n - 2)$ steps at u_1. To see that oit is indeed k we argue as follows. Node u_1 cannot use the shortcuts, thus by using just the linear graph it influences a new node every k steps. u_2 influences all nodes apart from u_n

at time k and then at time $2k$ it also influences u_n. All other nodes do a shortcut to u_2 at time k and then for all multiples of k their influences propagate to both directions from two sources, themselves and u_2, influencing 1 to 4 new nodes every k steps.

Next we show that the $Kl(l+1)/2$ ($l := |\mathrm{past}_{(u,t)}(0)|$) upper bound (of Theorem 11), on the time for another incoming influence to arrive, is optimal in the following sense: a node cannot obtain a better upper bound based solely on K and l. We establish this by showing that it is possible that a new incoming influence needs $\Theta(Kl^2)$ rounds to arrive, which excludes the possibility of a $o(Kl^2)$-bound to be correct as a protocol based on it may have nodes terminate without having heard of arbitrarily many other nodes. This, additionally, constitutes a tight example for the bound of Theorem 11.

Theorem 13 ([MCS14]). *For all n, l, K s.t. $n = \Omega(Kl^2)$, there is a dynamic graph with oit upper bounded by K and a round r such that, a node that has heard of l nodes by round r does not hear of another node for $\Theta(Kl^2)$ rounds.*

Proof. Consider the set $\mathrm{past}_{(u,t)}(0)$ and denote its cardinality by l. Take any dynamic graph on $\mathrm{past}_{(u,t)}(0)$, disconnected from the rest of the nodes, that satisfies $\mathsf{oit} \leq K$ and that all nodes in $\mathrm{past}_{(u,t)}(0)$ need $\Theta(Kl)$ rounds to causally influence all other nodes in $\mathrm{past}_{(u,t)}(0)$; this could, for example, be the alternating matchings graph from Section 5.1.1 with one matching appearing in rounds that are odd multiples of K and the other in even. In $\Theta(Kl)$ rounds, say in round j, some *intermediary* node $v \in \mathrm{past}_{(u,t)}(0)$ must get the outgoing influences of nodes in $\mathrm{past}_{(u,t)}(0)$ outside $\mathrm{past}_{(u,t)}(0)$ so that they continue to influence new nodes. Assume that in round $j-1$ the adversary directly connects all nodes in $\mathrm{past}_{(u,t)}(0)\backslash\{v\}$ to v. In this way, at time j, v forwards outside $\mathrm{past}_{(u,t)}(0)$ the $(j-2)$-states (and all previous ones) of all nodes in $\mathrm{past}_{(u,t)}(0)$. Provided that $V\backslash\mathrm{past}_{(u,t)}(0)$ is sufficiently big (see below) the adversary can now keep $S = \mathrm{past}_{(u,t)}(0)\backslash\{v\}$ disconnected from the rest of the nodes for another $\Theta(Kl)$ rounds (in fact, one round less this time) without violating $\mathsf{oit} \leq K$ as the new influences of the $(j-2)$-states of nodes in S may keep occurring outside S. The same process repeats by a new *intermediary* $v_2 \in S$ playing the role of v this time. Each time the process repeats, in $\Theta(|S|)$ rounds the intermediary gets all outgoing influences outside S and is then removed from S. It is straightforward to observe that a new incoming influence needs $\Theta(Kl^2)$ rounds to arrive at u in such a dynamic network. Moreover, note that $V\backslash\mathrm{past}_{(u,t)}(0)$ should also satisfy $\mathsf{oit} \leq K$ but this is easy to achieve by e.g. another alternating matchings dynamic graph on $V\backslash\mathrm{past}_{(u,t)}(0)$ this time. Also, $n - l = |V\backslash\mathrm{past}_{(u,t)}(0)|$ should satisfy $n - l = \Omega(Kl^2) \Rightarrow n = \Omega(Kl^2)$ so that the time needed for a $w \in V\backslash\mathrm{past}_{(u,t)}(0)$ (in an alternating matchings dynamic graph on $V\backslash\mathrm{past}_{(u,t)}(0)$) to influence all nodes in $V\backslash\mathrm{past}_{(u,t)}(0)$ and start influencing nodes in $\mathrm{past}_{(u,t)}(0)$ is asymptotically greater than the time needed for S to extinct. To appreciate this, observe that if $V\backslash\mathrm{past}_{(u,t)}(0)$ was too small then the outgoing influences of some $w \in V\backslash\mathrm{past}_{(u,t)}(0)$ that occur every K rounds would reach u before the $\Theta(Kl^2)$ bound was achieved. Finally, we note that whenever the number of nodes in $V\backslash S$ becomes odd we keep the previous alternating matchings dynamic graph and the new node becomes connected every K rounds to an arbitrary node (the same in every round). When $|V\backslash S|$ becomes even again we return to a standard alternating matchings dynamic graph.

We now show that even the criterion of Theorem 10, that is optimal if an upper bound on the ct is known, does not work in dynamic graphs with known an upper bound K on the oit. In particular, we show that for all times $t' < K(n/4)$ there is a dynamic graph with oit upper bounded by K, a node u, and a time $t \in \mathbb{N}$ s.t. $\text{past}_{(u,t)}(0) = \text{past}_{(u,t)}(t')$ while $\text{past}_{(u,t)}(0) \neq V$. In words, for any such t' it can be the case that while u has not been yet causally influenced by all initial states its past set from time 0 may become equal to its past set from time t', which violates the termination criterion of Theorem 10.

Theorem 14 ([MCS14]). *For all n, K and all times $t' < K(n/4)$ there is a dynamic graph with* oit *upper bounded by K, a node u, and a time $t > t'$ s.t. $\text{past}_{(u,t)}(0) = \text{past}_{(u,t)}(t')$ while $\text{past}_{(u,t)}(0) \neq V$.*

Proof. For simplicity assume that n is a multiple of 4. As in Proposition 4 (ii), by an alternating matchings dynamic graph, we can keep two parts V_1, V_2 of the network, of size $n/2$ each, disconnected up to time $K(n/4)$. Let $u \in V_1$. At any time t, s.t. $t' < t \leq K(n/4)$, the adversary directly connects $u \in V_1$ to all $w \in V_1$. Clearly, at that time, u learns the t'-states (and thus also the 0-states) of all nodes in V_1 and, due to the disconnectivity of V_1 and V_2 up to time $K(n/4)$, u hears (and has heard up to then) of no node from V_2. It follows that $\text{past}_{(u,t)}(0) = \text{past}_{(u,t)}(t')$ and $|\text{past}_{(u,t)}(0)| = n/2 \Rightarrow \text{past}_{(u,t)}(0) \neq V$ as required.

5.3.3 Hearing the Future In contrast to the previous negative results, we now present an optimal protocol for counting and all-to-all dissemination in dynamic networks with known an upper bound K on the oit, that is based on the following termination criterion. By definition of oit, if $\text{future}_{(u,0)}(t) = \text{future}_{(u,0)}(t + K)$ then $\text{future}_{(u,0)}(t) = V$. The reason is that if there exist uninfluenced nodes, then at least one such node must be influenced in at most K rounds, otherwise no such node exists and $(u, 0)$ must have already influenced all nodes (see also Figure 4). So, a fundamental goal is to allow a node to know its future set. Note that this criterion has a very basic difference from all termination criteria that have so far been applied to worst-case dynamic networks: instead of keeping track of its past set(s) and waiting for new incoming influences a node now directly keeps track of its future set and is informed by other nodes of its progress. We assume, for simplicity, a unique leader ℓ in the initial configuration of the system (this is not a necessary assumption and we will soon show how it can be dropped).

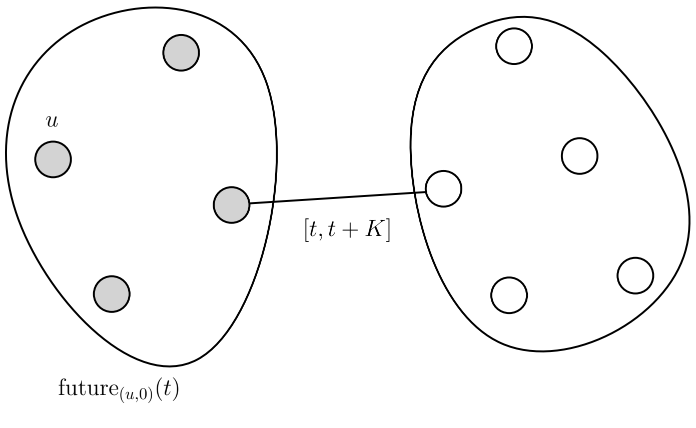

Fig. 4. If there are still nodes that have not heard from u, then if K is an upper bound on the oit, in at most K rounds another node will hear from u (by definition of the oit).

Protocol *Hear_from_known*. We denote by r the current round. Each node u keeps a list $Infl_u$ in which it keeps track of all nodes that first heard of $(\ell, 0)$ (the initial state of the leader) by u (u was between those nodes that first delivered $(\ell, 0)$ to nodes in $Infl_u$), a set A_u in which it keeps track of the $Infl_v$ sets that it is aware of initially set to $(u, Infl_u, 1)$, and a variable *timestamp* initially set to 1. Each node u broadcast in every round (u, A_u) and if it has heard of $(\ell, 0)$ also broadcasts $(\ell, 0)$. Upon reception of an id w that is not accompanied with $(\ell, 0)$, a node u that has already heard of $(\ell, 0)$ adds (w, r) to $Infl_u$ to recall that at round r it notified w of $(\ell, 0)$ (note that it is possible that other nodes also notify w of $(\ell, 0)$ at the same time without u being aware of them; all these nodes will write (w, r) in their lists). If it ever holds at a node u that $r > \max_{(v \neq u, r') \in Infl_u}\{r'\} + K$ then u adds (u, r) in $Infl_u$ (replacing any existing $(u, t) \in Infl_u$) to denote the fact that r is the maximum known time until which u has performed no further propagations of $(\ell, 0)$. If at some round r a node u modifies its $Infl_u$ set, it sets $timestamp \leftarrow r$. In every round, a node u updates A_u by storing in it the most recent $(v, Infl_v, timestamp)$ triple of each node v that it has heard of so far (its own $(u, Infl_u, timestamp)$ inclusive), where the "most recent" triple of a node v is the one with the greatest *timestamp* between those whose first component is v. Moreover, u clears multiple (w, r) records from the $Infl_v$ lists of A_u. In particular, it keeps (w, r) only in the $Infl_v$ list of the node v with the smallest id between those that share (w, r). Similarly, the leader collects all $(v, Infl_v, timestamp)$ triples in its own A_ℓ set. Let *tmax* denote the maximum timestamp appearing in A_l, that is the maximum time for which the leader knows that some node was influenced by $(\ell, 0)$ at that time. Moreover denote by I the set of nodes that the leader knows to have been influenced by $(\ell, 0)$. Note that I can be extracted from A_ℓ by $I = \{v \in V : \exists u \in V, \exists timestamp, r \in \mathbb{N} \text{ s.t. } (u, Infl_u, timestamp) \in A_\ell \text{ and } (v, r) \in Infl_u\}$. If at some round r it holds at the leader that for all $u \in I$ there is a $(u, Infl_u, timestamp) \in A_\ell$ s.t. $timestamp \geq tmax + K$ and $\max_{(w \neq u, r') \in Infl_u}\{r'\} \leq tmax$ then the leader outputs $|I|$ or I depending on whether counting or all-to-all dissemination needs to be solved and halts (it can also easily notify the other nodes to do the same in $K \cdot |I|$ rounds by a simple flooding mechanism and then halt).

The above protocol can be easily made to work without the assumption of a unique leader. The idea is to have all nodes begin as leaders and make all nodes prefer the leader with the smallest id that they have heard of so far. In particular, we can have each node keep an $Infl_{(u,v)}$ only for the smallest v that it has heard of so far. Clearly, in $O(D)$ rounds all nodes will have converged to the node with the smallest id in the network.

Theorem 15 ([MCS14]). *Protocol* Hear_from_known *solves counting and all-to-all dissemination in $O(D + K)$ rounds by using messages of size $O(n(\log K + \log n))$, in any dynamic network with dynamic diameter D, and with* oit *upper bounded by some K known to the nodes.*

Proof. In time equal to the dynamic diameter D, all nodes must have heard of ℓ. Then in another $D+K$ rounds all nodes must have reported to the leader all the direct outgoing influences that they performed up to time D (nodes that first heard of ℓ by that time) together with the fact that they managed to perform no new influences in the interval $[D, D+K]$. Thus by time $2D+K = O(D+K)$, the leader knows all influences that were ever performed, so no node is missing from its I set, and also knows that all these nodes for K consecutive rounds performed no further influence, thus outputs $|I| = n$ (for counting) or $I = V$ (for all-to-all dissemination) and halts.

Can these termination conditions be satisfied while $|I| < n$, which would result in a wrong decision? Thus, for the sake of contradiction, assume that *tmax* is the time of the latest influence that the leader is aware of, that $|I| < n$, and that all termination conditions are satisfied. The argument is that if the termination conditions are satisfied then (i) $I = \text{future}_{(\ell,0)}(tmax)$, that is the leader knows precisely those nodes that have been influenced by its initial state up to time $tmax$. Clearly, $I \subseteq \text{future}_{(\ell,0)}(tmax)$ as every node in I has been influenced at most by time $tmax$. We now show that additionally $\text{future}_{(\ell,0)}(tmax) \subseteq I$. If $\text{future}_{(\ell,0)}(tmax)\backslash I \neq \emptyset$, then there must exist some $u \in I$ that has influenced a $v \in \text{future}_{(\ell,0)}(tmax)\backslash I$ at most by time $tmax$ (this follows by observing that $\ell \in I$ and that all influence paths originate from ℓ). But now observe that when the termination conditions are satisfied, for each $u \in I$ the leader knows a $timestamp_u \geq tmax + K$, thus the leader knows all influences that u has performed up to time $tmax$ and it should be aware of the fact that $v \in \text{future}_{(\ell,0)}(tmax)$, i.e. it should hold that $v \in I$, which contradicts the fact that $v \in \text{future}_{(\ell,0)}(tmax)\backslash I$. (ii) The leader knows that in the interval $[tmax, tmax + K]$ no node in $I = \text{future}_{(\ell,0)}(tmax)$ performed a new influence. These result in a contradiction as $|\text{future}_{(\ell,0)}(tmax)| = |I| < n$ and a new influence should have occurred in the interval $[tmax, tmax + K]$ (by the fact that the oit is upper bounded by K).

Optimality follows from the fact that *a node u can know at time t that* $\text{past}_{(u,t)}(0) = V$ *only if* $\text{past}_{(u,t)}(K) = V$. This means that u must have also heard of the K-states of all nodes, which requires $\Theta(K + D)$ rounds in the worst case. If $\text{past}_{(u,t)}(K) \neq V$, then it can be the case that there is some $v \in V\backslash\text{past}_{(u,t)}(K)$ s.t. u has heard v's 0-state but not its K-state. Such a node could be a neighbor of u at round 1 that then moved far away. Again, similarly to Theorem 10, we can have arbitrarily many nodes to have no neighbor until time K (e.g. in the extreme case were oit is equal to K) and then from time K onwards are only connected to node v. As u has not heard from the K-state of v it also cannot have heard of the 0-state of arbitrarily many nodes.

An interesting improvement is to limit the size of the messages to $O(\log n)$ bits probably by paying some increase in time to termination. We almost achieve this by showing that an improvement of the size of the messages to $O(\log D + \log n)$ bits is possible (note that $O(\log D) = O(\log Kn)$) if we have the leader initiate *individual conversations* with the nodes that it already knows to have been influenced by its initial state. We have already successfully applied a similar technique in Section 4. The protocol, that we call *Talk_to_known*, solves counting and all-to-all dissemination in $O(Dn^2 + K)$ rounds by using messages of size $O(\log D + \log n)$, in any dynamic network with dynamic diameter D, and with oit upper bounded by some K known to the nodes.

We now describe the *Talk_to_known* protocol by assuming again for simplicity a unique leader (this, again, is not a necessary assumption).

Protocol *Talk_to_known*. As in *Hear_from_known*, nodes that have been influenced by the initial state of the leader (i.e. $(\ell, 0)$) constantly forward it and whenever a node v manages to deliver it then it stores the id of the recipient node in its local $Infl_v$ set. Nodes send in each round the time of the latest influence (i.e. the latest new influence of a node by $(\ell, 0)$), call it $tmax$, that they know to have been performed so far. Whenever the leader hears of a greater $tmax$ than the one stored in its local memory it reinitializes the process of collecting its future set. By this we mean that it waits K rounds and then starts again from the beginning, talking to the nodes that it has influenced itself, then to the nodes that were influenced by these nodes, and so on. The goal is for the leader to collect precisely the same information as in *Hear_from_known*. In particular, it sorts the nodes that it has influenced itself in ascending order of id and starts with the smallest one, call it v, by initiating a $talk(\ell, v, current\_round)$ message. All nodes forward the most recent *talk* message (w.r.t. to their timestamp component) that they know so far.

Upon receipt of a new $talk(\ell, v, timestamp)$ message (the fact that it is "new" is recognized by the timestamp), v starts sending $Infl_v$ to the leader in packets of size $O(\log n)$, for example a single entry each time, via $talk(v, \ell, current\_round, data\_packet)$ messages. When the leader receives a $talk(v, \ell, timestamp, data\_packet)$ message where $data\_packet = END\_CONV$ (for "END of CONVersation") it knows that it has successfully received the whole $Infl_v$ set and repeats the same process for the next node that it knows to have been already influenced by $(\ell, 0)$ (now also including those that it learned from v). The termination criterion is the same as in *Hear_from_known*.

Theorem 16 ([MCS14]). *Protocol* Talk_to_known *solves counting and all-to-all dissemination in $O(Dn^2 + K)$ rounds by using messages of size $O(\log D + \log n)$, in any dynamic network with dynamic diameter D, and with* oit *upper bounded by some K known to the nodes.*

Proof. Correctness follows from the correctness of the termination criterion proved in Theorem 15. For the bit complexity we notice that the timestamps and $tmax$ are of size $O(\log D)$ (which may be $O(\log Kn)$ in the worst case). The data packet and the id-components are all of size $O(\log n)$. For the time complexity, clearly, in $O(D)$ rounds the final outgoing influence of $(\ell, 0)$ will have occurred and thus the maximum $tmax$ that will ever appear is obtained by some node. In another $O(D)$ rounds, the leader hears of that $tmax$ and thus reinitializes the process of collecting its future set. In that process and in the worst case the leader must talk to $n-1$ nodes each believing that it performed $n-1$ deliveries (this is because in the worst case it can hold that any new node is concurrently influenced by all nodes that were already influenced and in the end all nodes claim that they have influenced all other nodes) thus, in total, it has to wait for $O(n^2)$ data packets each taking $O(D)$ rounds to arrive. The K in the bound is from the fact that the leader waits K rounds after reinitializing in order to allow nodes to also report whether they performed any new assignments in the $[tmax, tmax + K]$ interval.

6 Local Communication Windows

We assume here an underlying *communication network*, which is modeled by an undirected simple connected graph $C = (V, A)$, where V is a set of nodes and A is a set of undirected edges. We associate each $u \in V$ with a finite integer $c_u \geq 1$, called u's *neighborhood cover time* (or *cover time* for simplicity), and let $c \in \mathbb{N}^V$ be a vector of cover times indexed by nodes. We denote by $N(u)$ the *neighborhood* of node u in C, that is $N(u) = \{v \in V : \{u, v\} \in A\}$.

A dynamic graph $G = (V, E)$ has now $E : \mathbb{N}_{\geq 1} \to \mathcal{P}(A)$. We say that G *respects* a vector of cover times $c \in \mathbb{N}^V$ if for all $r \in \mathbb{N}$ and all $u \in V$ it holds that $\{v \in V : \{u, v\} \in \bigcup_{i=r}^{r+c_u-1} E(i)\} = N(u)$ (or $\geq$ in case we would allow a node to possibly communicate outside its underlying neighborhood); that is each node u must cover all its possible neighbors in any c_u-window of the dynamic graph. Note that again we do not require the instantaneous graphs to be connected. Nonetheless, it is not hard to see that this definition guarantees that each node may eventually influence every other node in the network. We are interested in protocols that are correct for all possible pairs (G, c), where G is a dynamic graph that respects the vector of cover times c.

First note that if nodes do not know their cover times nor some upper bound on them, then non-trivial halting computations are impossible. To see this, consider any protocol that terminates in k steps on some dynamic graph G. Now augment G with some dynamic graph D that has its first communication with G at time $k+1$ and notice that termination on G occurs at time k without any consideration of D.

We focus on the case in which each node u knows its precise cover time c_u. First of all, notice that for all cover times c there is a dynamic graph that respects c, namely the static graph in which $E(r) = A$ for all $r \in \mathbb{N}$. However, not all cover times c *admit a worst-case dynamic graph*, that is one in which for all $u \in V$ there is an $r \in \mathbb{N}$ such that $|\{v \in V : \{u, v\} \in \bigcup_{i=r}^{r+c_u-2} E(i)\}| < |N(u)|$. It is not hard to see that a cover-time vector c admits a worst-case dynamic graph G iff $\forall u \in V, \exists v \in N(u)$ such that $c_v \geq c_u$.

An interesting question is whether nodes can verify if a given vector of cover times admits a worst-case dynamic graph. In fact, we want nodes to accept if all cover times are consistent and fix inconsistent cover times otherwise. Let C_u be an upper bound on c_u. Each node u must check whether there is some $v \in N(u)$ such that $C_v \geq C_u$. u broadcasts C_u for C_u rounds. If $C_v < C_u$ for all $v \in N(u)$, then u sets C_u to $\max_{v \in N(u)}\{C_v\}$, otherwise it accepts.

We now deal with the problem of information dissemination and counting and present a protocol for the latter problem.

Let $u = u_0, u_1, \ldots, u_k = v$ be a simple path p joining nodes u, v. The worst case time for u to influence v by messages traversing p is $l(p) = \sum_{i=1}^{k-1} \min\{c_{u_i}, c_{u_{i+1}}\}$ (called *length* or *maximum delay*). Extend l as $l(u, v) = \min_{p \in P(u,v)} l(p)$, where $P(u, v)$ is the set of all simple paths joining u, v. In the dynamic networks under consideration we have that the dynamic diameter is $D = \max_{u,v \in V} l(u, v)$. It is obvious that if all nodes knew some upper bound $H \geq D$ then each node could halt after H rounds knowing that it has influenced and been influenced by all other nodes. A natural question is whether nodes can achieve this without knowing D in advance. For example, is there a terminating algorithm for *counting* (i.e. for computing n) if nodes only know their exact cover times? In the sequel, we answer this question in the affirmative.

Let

– $\mathrm{psum}_{(u,t')}(t) := \sum_{v \in \mathrm{past}_{(u,t')}(t)} c_v$, and
– $\mathrm{fsum}_{(u,t)}(t') := \sum_{v \in \mathrm{future}_{(u,t)}(t')} c_v$.

Lemma 5. *For all times t, t' such that $t \leq t'$, all nodes u, v, and all $k \geq 1$, if $v \in \mathrm{past}_{(u,t')}(t)$ for all E then $v \in \mathrm{past}_{(u,t'+k)}(t + k)$.*

Proof. Take any $v \in \mathrm{past}_{(u,t')}(t)$. To show that $v \in \mathrm{past}_{(u,t'+k)}(t+k)$ we notice that for any dynamic edge function E' there exists E such that $E'(r + k) = E(r)$ for all $t \leq r \leq t'$. □

Lemma 6. $\mathrm{past}_{(u,t)}(0) \subseteq \mathrm{past}_{(u,\mathrm{psum}_{(u,t)}(0)-c_v)}(0)$.

Proof. We show that $v \in \mathrm{past}_{(u,t)}(0)$ implies $v \in \mathrm{past}_{(u,\mathrm{psum}_{(u,t)}(0)-c_v)}(0)$. The time-node $(v, 0)$ has influenced (u, t) via a simple path p that only visits nodes from $\mathrm{past}_{(u,t)}(0)$ since $(v, 0) \rightsquigarrow (w, j) \rightsquigarrow (u, t)$ for any intermediate node w implies $w \in \mathrm{past}_{(u,t)}(0)$; to see this note that $(w, 0) \rightsquigarrow (w, j)$ for all $j \geq 0$. Clearly, the longest such path p' is a path that is hamiltonian in $C[\mathrm{past}_{(u,t)}(0)]$ [8] beginning from u and ending at v. Since $l(p) \leq l(p') \leq \mathrm{psum}_{(u,t)}(0) - c_v$ and $(v, 0) \rightsquigarrow (u, l(p))$ it also holds that $(v, 0) \rightsquigarrow (u, \mathrm{psum}_{(u,t)}(0) - c_v)$ or equivalently $v \in \mathrm{past}_{(u,\mathrm{psum}_{(u,t)}(0)-c_v)}(0)$. □

Lemma 7. *For all nodes $u \in V$ and times $t \geq 0$ we have:*

1. $|\mathrm{past}_{(u,\mathrm{psum}_{(u,t)}(0))}(0)| \geq \min\{|\mathrm{past}_{(u,t)}(0)| + 1, n\}$ *and*
2. $|\mathrm{future}_{(u,0)}(\mathrm{fsum}_{(u,0)}(t))| \geq \min\{|\mathrm{future}_{(u,0)}(t)| + 1, n\}$.

[8] We denote by $G[V']$ the subgraph of a graph G induced by nodes in V'.

Proof. We only prove the first statement since the second is symmetric. The only interesting case is when $|\mathrm{past}_{(u,t)}(0)| < n$ in which case there exists $w \in V \backslash \mathrm{past}_{(u,t)}(0)$. By Lemma 6, $\mathrm{past}_{(u,t)}(0) \subseteq \mathrm{past}_{(u,\mathrm{psum}_{(u,t)}(0)-c_v)}(0) \subseteq \mathrm{past}_{(u,\mathrm{psum}_{(u,t)}(0))}(0)$. So we just need to show that there is a $w \in \mathrm{past}_{(u,\mathrm{psum}_{(u,t)}(0))}(0) \backslash \mathrm{past}_{(u,t)}(0)$. Connectivity ensures that there is some $\{w, v\} \in A$, for $w \in V \backslash \mathrm{past}_{(u,t)}(0)$ and $v \in \mathrm{past}_{(u,t)}(0)$. Clearly $(w, 0) \rightsquigarrow (v, c_v)$. Since $(v, 0) \rightsquigarrow (u, \mathrm{psum}_{(u,t)}(0)-c_v)$, by Lemma 5 $(v, c_v) \rightsquigarrow (u,\ \mathrm{psum}_{(u,t)}(0))$. Transitivity ensures that $(w, 0) \rightsquigarrow (u, \mathrm{psum}_{(u,t)}(0))$ and $w \in \mathrm{past}_{(u,\mathrm{psum}_{(u,t)}(0))}(0)$. □

Lemma 7 provides us with the following criterion for a node to detect when it has been causally influenced by all other nodes: $|\mathrm{past}_{(u,\mathrm{psum}_{(u,t)}(0))}(0)| = |\mathrm{past}_{(u,t)}(0)| \Rightarrow |\mathrm{past}_{(u,t)}(0)| = V$. That is, at any time t, any new influence of the state of u by some initial state must occur at most by time $\mathrm{psum}_{(u,t)}(0)$. If this time elapses without any new influence then u knows that it has been causally influenced by all other nodes. An easier to perform but equivalent test is the following: $t = \mathrm{psum}_{(u,t)}(0) \Rightarrow |\mathrm{past}_{(u,\mathrm{psum}_{(u,t)}(0))}(0)| = |\mathrm{past}_{(u,t)}(0)| \Rightarrow |\mathrm{past}_{(u,t)}(0)| = V$. In the following proposition, we use the latter criterion to solve counting.

But first define an edge weight $w(e)$ for each edge $e \in A$ as $w(e) := min c_u, c_v$. We are then guaranteed that an edge e appears at least once in every time interval of length $w(e)$. This implies that within time $W := \sum_{e \in D(C)} w(e)$, where $D(C)$ is a diameter of C (that is within time equal to the weighted diameter of C), everyone hears from everyone else and then another $\sum_{u \in V} c_u - W$ rounds are needed for the nodes to know that they are done.

Proposition 6. *Counting can be solved in $O(\sum_{u \in V} c_u)$ rounds using messages of size $O(n \log n + \sum_{u \in V} c_u)$.*

Proof. Each node u maintains a set of UIDs A_u, where initially $A_u(0) = \{u\}$, and a vector $c_u[]$ of cover times indexed by UIDS in A_u, where initially $c_u = (c_u)$. In each round r, u sends (A_u, c_u) to all its current neighbors, stores in A_u all received UIDs and, for each new UID $v \notin A_u(r-1)$, u stores t_v in c_u. Moreover, nodes keep track of the round number. At the end of each round r, if $r = \sum_{v \in A_u(r)} c_u[v]$ node u halts and outputs $|A_u|$; otherwise, u continues on to the next round. □

7 Conclusions

In this chapter, we discussed several recently introduced models and problems regarding computational network analysis which we treated from a theoretical point of view. In Section 4, we studied the fundamental naming and counting problems (and some variations) in networks that are anonymous, unknown, and possibly dynamic. Network dynamicity was modeled by the 1-interval connectivity model [KLO10], in which communication is synchronous and a (worst-case) adversary chooses the edges of every round subject to the condition that each instance is connected. We first focused on static networks with broadcast where we proved that, without a leader, counting is impossible to solve and that naming is impossible to solve even with a leader and even if nodes know n. These impossibilities carry over to dynamic networks as well. We also showed that a unique leader suffices in order to solve counting in linear time. Then we focused on dynamic networks with broadcast. We conjectureed that dynamicity renders nontrivial computation impossible. In view of this, we allowed the nodes know an upper bound on the maximum degree that will ever appear and showed that in this case the nodes can obtain an upper bound on n. Finally, we replaced broadcast with *one-to-each*, in which a node may send a different message to each of its neighbors. Interestingly, this natural variation was proved to be computationally equivalent to a full-knowledge model, in which unique names exist and the size of the network is known.

Then, in Section 5, we discussed the model of [MCS14], which was the first in the literature to consider worst-case dynamic networks that are free of any connectivity assumption about their instances. To enable a quantitative study we introduced some novel generic metrics that capture the speed of information propagation in a dynamic network. We proved that fast dissemination and computation are possible even under continuous disconnectivity. In particular, we presented optimal termination conditions and protocols based on them for the fundamental counting and all-to-all token dissemination problems.

There are many open problems and promising research directions related to the above findings. We would like to achieve satisfactory lower and upper bounds for counting and information dissemination. Techniques from [HCAM12] or related ones may be applicable to achieve quick token dissemination. It would be also important to refine the metrics proposed in this section so that they become more informative. For example, the oit metric, in its present form, just counts the time needed for another outgoing influence to occur. It would be useful to define a metric that counts the number of new nodes that become influenced per round which would be more informative w.r.t. the speed of information spreading. Note that in our work (and all previous work on the subject) information dissemination is only guaranteed under continuous broadcasting. How can the number of redundant transmissions be reduced in order to improve communication efficiency? Is there a way to exploit visibility to this end? Does predictability help (i.e. some knowledge of the future)?

References

[AAD$^+$06] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, pages 235–253, mar 2006.

[AAER07] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20[4]:279–304, November 2007.

[AFR06] J. Aspnes, F. E. Fich, and E. Ruppert. Relationships between broadcast and shared memory in reliable anonymous distributed systems. *Distributed Computing*, 18[3]:209–219, February 2006.

[AGVP90] 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.

[AH00] S. Albers and M. Henzinger. Exploring unknown environments. *SIAM J. Comput.*, 29[4]:1164–1188, 2000.

[AKL08] C. Avin, M. Koucký, and Z. 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 (ICALP), Part I*, pages 121–132, Berlin, Heidelberg, 2008. Springer-Verlag.

[Ang80] D. Angluin. Local and global properties in networks of processors (extended abstract). In *Proceedings of the twelfth annual ACM symposium on Theory of computing (STOC)*, pages 82–93, New York, NY, USA, 1980. ACM.

[APRU12] J. Augustine, G. Pandurangan, P. Robinson, and E. Upfal. Towards robust and efficient computation in dynamic peer-to-peer networks. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 551–569. SIAM, 2012.

[AR07] J. Aspnes and E. Ruppert. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science*, 93:98–117, October 2007.

[ASW88] H. Attiya, M. Snir, and M. K. Warmuth. Computing on an anonymous ring. *J. ACM*, 35[4]:845–875, October 1988.

[AW04] H. Attiya and J. Welch. *Distributed computing: fundamentals, simulations, and advanced topics*, volume 19. Wiley-Interscience, 2004.

[BCF09] H. Baumann, P. Crescenzi, and P. Fraigniaud. Parsimonious flooding in dynamic graphs. In *Proceedings of the 28th ACM symposium on Principles of distributed computing (PODC)*, pages 260–269. ACM, 2009.

[Ber96] K. A. Berman. Vulnerability of scheduled networks and a generalization of Menger's theorem. *Networks*, 28[3]:125–134, 1996.

[Bol98] B. Bollobás. *Modern Graph Theory.* Springer, corrected edition, July 1998.

[BV99] P. Boldi and S. Vigna. Computing anonymously with arbitrary knowledge. In *Proceedings of the 18th annual ACM symposium on Principles of distributed computing (PODC)*, pages 181–188. ACM, 1999.

[CFQS12] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27[5]:387–408, 2012.

[CMM$^+$08] A. E. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time in edge-markovian dynamic graphs. In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing (PODC)*, pages 213–222, New York, NY, USA, 2008. ACM.

[CMM12] J. Chalopin, Y. Métivier, and T. Morsellino. On snapshots and stable properties detection in anonymous fully distributed systems (extended abstract). In *Structural Information and Communication Complexity*, volume 7355 of *LNCS*, pages 207–218. Springer, 2012.

[CMN$^+$11] I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis. Passively mobile communicating machines that use restricted space. *Theoretical Computer Science*, 412[46]:6469–6483, October 2011.

[Dol00] S. Dolev. *Self-stabilization.* MIT Press, Cambridge, MA, USA, 2000.

[Dot14] D. Doty. Timing in chemical reaction networks. In *Proc. of the 25th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 772–784, 2014.

[DP90] X. Deng and C. Papadimitriou. Exploring an unknown graph. In *31st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 355–361. IEEE, 1990.

[DPR$^+$13] C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *Proc. of the 24th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2013.

[FPPP00] P. Fraigniaud, A. Pelc, D. Peleg, and S. Pérennes. Assigning labels in unknown anonymous networks (extended abstract). In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, PODC '00, pages 101–111. ACM, 2000.

[Hae11] B. Haeupler. Analyzing network coding gossip made easy. In *Proceedings of the 43rd annual ACM symposium on Theory of computing (STOC)*, pages 293–302. ACM, 2011.

[HCAM12] B. Haeupler, A. Cohen, C. Avin, and M. Médard. Network coded gossip with correlated data. *CoRR*, abs/1202.1801, 2012.

[HS12] P. Holme and J. Saramäki. Temporal networks. *Physics reports*, 519[3]:97–125, 2012.

[KKK00] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing (STOC)*, pages 504–513, 2000.

[KLO10] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 513–522. ACM, 2010.

[KMO11] F. Kuhn, Y. Moses, and R. Oshman. Coordinated consensus in dynamic networks. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing (PODC)*, pages 1–10, 2011.

[KO11] F. Kuhn and R. Oshman. Dynamic networks: models and algorithms. *SIGACT News*, 42:82–96, March 2011. Distributed Computing Column, Editor: Idit Keidar.

[Lam78] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21[7]:558–565, 1978.

[Lyn96] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann; 1st edition, 1996.

[MCS11a] O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Mediated population protocols. *Theoretical Computer Science*, 412[22]:2434–2450, May 2011.

[MCS11b] O. Michail, I. Chatzigiannakis, and P. G. Spirakis. *New Models for Population Protocols.* N. A. Lynch (Ed), Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool, 2011.

[MCS12] O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Brief announcement: Naming and counting in anonymous unknown dynamic networks. In *Proceedings of the 26th international conference on Distributed Computing (DISC)*, pages 437–438, Berlin, Heidelberg, 2012. Springer-Verlag.

[MCS13] O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Naming and counting in anonymous unknown dynamic networks. In *15th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 281–295. Springer, 2013.

[MCS14] O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Causality, influence, and computation in possibly disconnected synchronous dynamic networks. *Journal of Parallel and Distributed Computing*, 74[1]:2016–2026, 2014.

[MMCS13] G. B. Mertzios, O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Temporal network optimization subject to connectivity constraints. In *40th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 7966 of *Lecture Notes in Computer Science*, pages 663–674. Springer, July 2013.

[MS14] O. Michail and P. G. Spirakis. Simple and efficient local codes for distributed stable network construction. In *Proceedings of the 33rd ACM Symposium on Principles of Distributed Computing (PODC)*, 2014. To appear.

[Orl81] J. B. Orlin. The complexity of dynamic languages and dynamic optimization problems. In *Proceedings of the 13th annual ACM symposium on Theory of computing (STOC)*, pages 218–227. ACM, 1981.

[OW05] R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of the 2005 joint workshop on Foundations of mobile computing (DIALM-POMC)*, pages 104–110, 2005.

[Sch02] C. Scheideler. Models and techniques for communication in dynamic networks. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 27–49, 2002.

[Soi09] A. Soifer. *The Mathematical Coloring Book: Mathematics of Coloring and the Colorful Life of its Creators*. Springer; 1st edition, 2009.

[YK96] M. Yamashita and T. Kameda. Computing on anonymous networks. i. characterizing the solvable cases. *Parallel and Distributed Systems, IEEE Transactions on*, 7[1]:69–89, 1996.