Distributed Computing Column 41
Computing Over Dynamic Networks

Idit Keidar
Dept. of Electrical Engineering, Technion
Haifa, 32000, Israel
idish@ee.technion.ac.il

This column focuses on the rather recent topic of *dynamic communication networks*, which are modeled as evolving graphs. Admittedly, the idea that a network can be dynamic is hardly new— networking research has been considering network topology changes and churn for decades. But by and large, past research considered such changes to be exceptions, and focused on adapting to them and re-stabilizing. Only in recent years, we begin to see works that treat dynamic changes as the norm. Such work on constantly-evolving networks may prove important in capturing many real world scenarios, in particular, ones that arise in wireless networks with mobile devices.

In today's column, Fabian Kuhn and Rotem Oshman survey recent work on dynamic networks. They consider adversarial as well as random models of graph evolution. In this context, they explain how the classical graph notions of diameter and cover time generalize to dynamic ones. These notions have important implications on information dissemination in the network. Fabian and Rotem then present an example algorithm for counting and information dissemination on adversarially evolving graphs. As research on dynamic networks is still in its infancy, many questions remain open; the column concludes with a discussion of some promising directions for future research. Many thanks to Fabian and Rotem for this survey!

Call for contributions: I welcome suggestions for material to include in this column, including news, reviews, open problems, tutorials and surveys, either exposing the community to new and interesting topics, or providing new insight on well-studied topics by organizing them in new ways.

Dynamic Networks: Models and Algorithms

Fabian Kuhn
Faculty of Informatics
University of Lugano
6904 Lugano, Switzerland
fabian.kuhn@usi.ch

Rotem Oshman
Computer Science & Artificial Intelligence Lab
MIT
Cambridge, MA 02139, USA
rotem@csail.mit.edu

1 Introduction

The study of dynamic networks has come into popularity recently, and many models and algorithms for such networks have been suggested. In this column we survey some recent work on dynamic network algorithms, focusing on the effect that model parameters such as the type of adversary, the network diameter, and the graph expansion can have on the performance of algorithms. We focus here on high-level models that are not induced by some specific mobility pattern or geographic model (although much work has gone into geographic models of dynamic networks, and we touch upon them briefly in Section 2).

Dynamic network behavior has long been studied in distributed computing literature, but initially it was modeled as a *fault* in the network; as such, it was typically bounded, either in duration or in the number of nodes affected (or both). For example, in the general omission-fault model, if two nodes that could once communicate can no longer send messages to each other, this is treated as a failure of one of the nodes, and the number of faulty nodes is assumed to be bounded. Another example is self-stabilizing algorithms, which are guaranteed to function correctly only when changes to the network have stopped [16]. These models are appropriate for modeling unreliable static networks, but they are not appropriate for mobile and ad hoc networks, where changes are unbounded in number and occur continually.

In the sequel we survey several models for dynamic networks, both random and adversarial, and algorithms for these models. The literature on dynamic networks is vast, and this column is not intended as a comprehensive survey. We have chosen to focus on models and algorithms that exhibit the following properties.

Unceasing changes. Changes to the network never stop, and are not restricted in frequency.

Dynamic topology not subject to the algorithm's control. We concentrate solely on algorithms that execute in a dynamic network, but have no control over how the topology changes.

Terminating computation. In this column we focus on algorithms whose execution eventually completes (in contrast to, e.g., eventually stabilizing computation, where the outputs eventually converge to the right answer, but the participants do not know when this occurs). For deterministic algorithms, we require all all nodes to eventually halt in every execution. When we discuss randomized algorithms, their correctness guarantee is of the form "the algorithm achieves its goal in $f(n)$ rounds with high probability", where "high probability" here means $1 - o(1)$. We note that such algorithms can be used for either terminating or eventually stabilizing computation: if we have the nodes halt after $f(n)$ rounds, then the goal is accomplished with high probability; if instead we repeat the algorithm forever, then with probability 1 the goal is eventually accomplished.

Non-geographical models. With the exception of one model discussed in Section 3.2, the network models we discuss here are not motivated by a geographical distribution of the nodes on the plane or by an underlying mobility pattern for the nodes. Rather, we focus on abstract models, where the dynamic communication topology can reflect any communication network — including both wireless mobile networks and unreliable traditional point-to-point networks.

The rest of the column is organized as follows. In the next section we introduce the basic dynamic graph model used in the work we survey, and point out several basic parameters of the model that can affect algorithmic performance. In Section 3 we define the *dynamic diameter* of a dynamic graph, a generalization of the notion of the diameter in a static graph; we illustrate the role it plays and give some examples of how it is affected by basic model parameters. In Section 4 we discuss a recent result on computation and information dissemination in adversarial dynamic networks, and in Section 5 we survey some of the work on broadcast in dynamic radio networks. Finally, in Section 6 we briefly discuss a few alternative models to the ones discussed in this column, and we conclude in Section 7 with some directions for future research.

2 Evolving Graphs: A Basic Model for Dynamic Networks

For simplicity, and because this almost exclusively is the case considered in the literature, we restrict our attention to dynamic networks with a fixed set of participants. Such a dynamic network can typically be modeled as a *dynamic graph* $G = (V, E)$, where V is a set of nodes that we identify with the participants in the computation, and $E : \mathbb{N}^+ \to 2^{V \times V}$ is a dynamic edge function assigning to each round $r \in \mathbb{N}^+$ a set of edges $E(r)$ for that round. (The discussion here assumes a synchronous model, and this is true of much of the literature. In Section 3 we briefly touch upon asynchronous dynamic networks.) We think of a *round* as occurring between two *times*; round $r \in \mathbb{N}^+$ occurs between time $r - 1$ and time r. Depending on the model, $E(r)$ may be a set of directed or undirected edges. Throughout this survey, we use $n = |V|$ to denote the total number of nodes, and we use $G(r) = (V, E(r))$ to denote the instantaneous communication graph in round r. In the literature, such dynamic graphs have also been termed *evolving graphs* [3, 17, 18, 21].

An execution in a dynamic graph induces a *causal order* $\rightsquigarrow \subseteq (V \times \mathbb{N})^2$, where $(u, t_1) \rightsquigarrow (v, t_2)$ denotes the fact that u's state at time t_1 influences v at time t_2. The causal order is defined in the usual way: it is the transitive and reflexive closure of the relation $\to$, where $(u, t) \to (v, t')$ iff $t' > t$ and either $u = v$ or u sends a message at time t that v receives at time t'. In the case of a synchronous model where nodes broadcast on all their available links in each round, the direct causal order reduces to the following: $(u, t) \to (v, t + 1)$ iff either $u = v$ or $(u, v) \in E(t + 1)$.

Naturally, what can be achieved in a dynamic network and at what cost it can be done largely depends on how the dynamic set of edges is chosen. Consequently, dynamic network models can feature different assumptions about the way the dynamic graph is generated, the properties of each instantaneous graph $E(r)$, and equally important the properties of the causal order, that is, in which way the graph $G(r)$ in round r depends on earlier graphs. In this column we focus on the following properties.

Adversary model. The adversary model determines the method by which the dynamic edge function is generated. We focus on three types of adversaries: an *adaptive worst-case adversary* ("adaptive adversary" for short) generates the dynamic graph on-the-fly; in each round r, the edges $E(r)$ are chosen based on the execution so far, including the states of all the participants at time $r-1$ and their randomness up to time $r-1$ (exclusive). An *oblivious worst-case adversary* ("oblivious adversary") commits to a dynamic graph in advance, choosing an edge function E that does not depend on the randomness of the participants or their states during the execution. (For deterministic algorithms, adaptive and oblivious adversaries are equivalent.)

Finally, a *random-graph adversary* is not an adversary at all, but simply a distribution according to which the instantaneous graph at each time is chosen. Random dynamic graph models are typically Markovian and often synchronous. In that case, the dynamic graph for round $r+1$ is chosen according to some probability distribution that only depends on $G(r)$ (see, e.g., [3]). A special class of Markovian graphs has been considered in [5, 11], where it is assumed that for all $r \geq 1$, each edge $e \notin E(r)$ appears in round $r+1$ (that is, $e \in E(r+1)$) with probability p independent of all other edges, and each edge $e \in E(r)$ disappears in round $r+1$ (that is, $e \notin E(r+1)$) with probability q independent of other edges. We refer to such graphs as *Markovian dynamic graphs with birth rate p and death rate q*. If we choose $q = 1-p$, we obtain an interesting special case, where the graph of each round is an independent Erdős–Rényi random graph $G_{n,p}$ (see, e.g., [23]).

Instantaneous connectivity and expansion. Real communication networks are often dense and well-connected. To capture such properties it is often assumed that each instantaneous graph $(V, E(r))$ has some desirable connectivity properties; for example, it may be assumed that each instantaneous graph is a vertex expander, or that it is k-vertex connected for some $k > 1$, and so on. In adversarial models, it is usually assumed that the graph is at least connected (or strongly connected for directed graphs) in each round. In random graph models the distribution is typically one that guarantees that each instantaneous graph or the union of a few consecutive graphs is with very high probability connected (and often has good expansion to boot).

To formally capture the connectivity of each instantaneous graph, we define a *vertex growth function* $\mathrm{g} : \mathbb{N} \to \mathbb{N}$, which generalizes the usual notion of vertex expansion. We say that a (static) graph $G = (V, E)$ has vertex growth g if for every vertex set $S \subset V$ of size $s = |S| \leq |V|/2$, the set of neighbors $N(S) := \{v \in V \setminus S : \exists u \in S : (u, v) \in E\}$ has size at least $|N(S)| \geq \mathrm{g}(|S|)$.

For example, if G is connected, then the vertex growth of G is at least 1; if G is k-connected, the vertex growth of G is at least k (that is, the vertex growth is at least the constant function $g(s) = k$); and if G has vertex expansion $\alpha > 0$, then G has vertex growth $g(s) = \alpha \cdot s$.

We use $\mathrm{g}^{(f)}$ to denote the vertex growth of a set after f consecutive rounds, counting members of the original set as well as new neighbors: the base case is $\mathrm{g}^{(0)}(s) = s$, and for $f > 1$, the growth is $\mathrm{g}^{(f)}(s) = \mathrm{g}^{(f-1)}(s) + \mathrm{g}\big(\mathrm{g}^{(f-1)}(s)\big)$.

Long-term stability. To model stability inherent in the dynamic network, one may assume for example that topology changes only occur once every certain number of rounds [3]. Alternatively, in [28] we introduce a property called *T-interval connectivity*, which asserts that over any window of T consecutive rounds there is some connected spanning subgraph that remains stable during those T rounds. (This graph is not known to the nodes in advance, and indeed can be chosen adaptively by the adversary.) Edges that are not in the stable graph may appear and disappear arbitrarily.

In the sequel we survey some of the work that considers various settings of the parameters above.

3 Dynamic Diameter and Cover Time

3.1 Adversarial Dynamic Graphs

One of the crucial features affecting the efficiency of computation in a dynamic graph model is its *dynamic diameter* (also called *flooding time*, e.g., in [5, 11]). The dynamic diameter is a bound on the time required for each participant to be causally influenced by each other participant; formally, the dynamic diameter is bounded by D if at all times t and for all $u, v \in V$ it holds that $(u, t) \rightsquigarrow (v, t+D)$. In synchronous broadcast models this definition is straightforward and depends only on the dynamic graph. In non-synchronous models, where typically the nodes do not broadcast constantly, one also has to assume something about the protocol; for example, one may assume that nodes broadcast at least once every ΔH time units, and that there is some upper bound U on message delays. These assumptions induce a well-defined dynamic diameter that depends on the dynamic graph in question as well as on U and on ΔH; see below.

The dynamic diameter is a natural lower bound on the time required for information to propagate through the network, just like the "standard" diameter in static networks: if some node broadcasts a message, and the message is immediately forwarded by all those who receive it, then all nodes receive the message in time that is bounded by the dynamic diameter. One might wonder if this new definition is necessary: can we not simply use a bound on the diameter of each instantaneous graph $G(t)$?

The answer turns out to be negative. A striking example that information dissemination in a dynamic graph can be much slower than in static networks is given in [3], where it is shown that even when the adversary is oblivious, there is a dynamic graph in which the instantaneous graph in each round has diameter 2, but the cover time of a random walk in the graph is $\Omega(2^n)$. This stands in stark contrast to the cover time of a static graph, which is always $O(D \cdot |E| \cdot \log n)$, i.e., polynomial in n and linear in the diameter.

The dynamic graph given in [3] is a "dynamic star" over nodes $0, \ldots, n-1$, where in round r the center of the star is node $r \bmod (n-1)$, and the remaining nodes are leaves (see Fig. 1). In addition, all nodes have self-loops, allowing the random walk to remain at one node for several rounds. Notice that node $n-1$ is never at the center of the star. If the random walk starts at node 0 (which is a leaf in rounds $1, 2, \ldots, n-2$), the only way the walk can reach node $n-1$ is by staying at some leaf for $n-2$ rounds until this leaf becomes the center of the star, and then moving to node $n-1$. (If the random walk moves to the center "too soon" then the process starts over, because the center will itself be a leaf again in the next round.) The probability that the random walk stays at a leaf for $n-2$ consecutive rounds is $1/2^{n-2}$, and hence the expected time until the walk covers all nodes is $\Omega(2^n)$.

It is further shown in [3] that the cover time is bounded from above by $n^{O(n)}$, although in some special cases it is polynomial in n (for example, when the instantaneous graph is always d-regular). However, these upper bounds hold only for oblivious adversaries; an adaptive adversary can prevent a random walk or any other token-circulation mechanism from ever completing. In fact, the adversary can restrict the token to two nodes u_0, u_1, where u_0 is the node that has the token initially and u_1 is some arbitrarily chosen node: to do this, whenever node u_i holds the token, the adversary connects u_i only to u_{1-i}, and connects u_{1-i} to the remainder of the graph. Thus u_i has no choice except to either keep the token or pass it to u_{1-i}.

The dynamic star described above also illustrates why the instantaneous diameter does not reflect the time required to flood a message through the network. If node 0 broadcasts a message on all its links in the

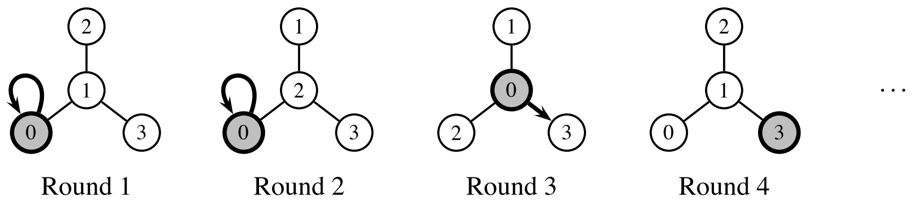

Figure 1: The dynamic star graph from [3] with $n = 4$. Self-loops are generally not shown. Bold nodes and edges indicate the shortest walk from node 0 to node 3 in the dynamic graph.

first round, and in each round this message is re-broadcast by all nodes that know it (that is, all nodes that have heard it in the past), then $n-1$ rounds are required for the message to reach node $n-1$ (in Fig. 1, node 3 does not receive the message until round 3). The *dynamic* diameter of the network is $n-1$, even though the instantaneous diameter is always bounded by 2.

Some instantaneous properties of the communication graph do translate into bounds on the dynamic diameter. For instance, if the instantaneous graph in each round is a vertex expander, then the dynamic diameter is $O(\log n)$; if each instantaneous graph is k-connected, the dynamic diameter is $O(n/k)$ [28]. The key is that unlike a small instantaneous diameter, these properties guarantee that a large number of new nodes are causally influenced by each node in each round. In general we can state the following.

Lemma 3.1. *Let $G = (V, E)$ be dynamic graph such that for every round r, the static graph $G(r)$ has vertex growth g. The dynamic diameter of G is at most $2d$, where d is the smallest integer such that $\mathrm{g}^{(d)}(1) > n/2$.*

Proof sketch. We must show that for any two nodes $u, v \in V$ and time $t \geq 0$, we have $(u, t) \rightsquigarrow (v, t + 2d)$.

The vertex growth of G tells us that at first, the set of nodes causally influenced by u grows quickly. Formally, let $U_{t'} := \{w \in V \mid (u, t) \rightsquigarrow (w, t')\}$ denote the set of nodes causally influenced at time t' by the state of u at time t (see Fig. 2). If $|U_{t'}| \leq n/2$, then the vertex growth of G guarantees that at least $\mathrm{g}(|U_{t'}|)$ edges cross the cut between $U_{t'}$ and $V \setminus U_{t'}$; therefore, the causal influence of u spreads to at least $\mathrm{g}(|U_{t'}|)$ new nodes, and $|U_{t'+1}| \geq |U_{t'}| + \mathrm{g}(|U_{t'}|) = \mathrm{g}^{(1)}(|U_{t'}|)$. This process continues until we reach a time t' where $|U_{t'}| > n/2$; because we chose d so that $\mathrm{g}^{(d)}(1) > n/2$, we have $|U_d| > n/2$.

Once the causal influence of u has reached $n/2$ nodes, we can no longer apply vertex growth to show that it continues to spread quickly. Instead, we now turn our attention to node v, and consider the inverse question: how many nodes causally influence v at time $t + 2d$? Let $V_{t'} := \{w \in V \mid (w, t') \rightsquigarrow (v, t + 2d)\}$ denote the nodes whose state at time t' influences v at time $t + 2d$. We trace the growth of this set going back in time starting from time $t + 2d$. Just as before, as long as $|V_{t'}| \leq n/2$ we have $|V_{t'-1}| \geq \mathrm{g}^{(1)}(|V_{t'}|)$. Therefore, $|V_{t+d}| > n/2$. But now we are done: since $|U_{t+d}| > n/2$ and $|V_{t+d}| > n/2$, there is some node in the intersection, $w \in U_{t+d} \cap V_{t+d}$. This node satisfies both $(u, t) \rightsquigarrow (w, t+d)$ and $(w, t+d) \rightsquigarrow (v, t+2d)$. Consequently, $(u, t) \rightsquigarrow (v, t + 2d)$. □

Similar arguments to the one in Lemma 3.1 have also been applied in slightly different contexts, e.g., [8, 25].

The dynamic diameter in partially-synchronous networks. The majority of this column discusses synchronous dynamic networks, but we now make a brief foray into *partially-synchronous networks*. There are many different versions of partial synchrony in the literature; here we are interested in the model where nodes operate in real time (that is, time is modeled as a real number, rather than a natural number), and each node is equipped with a hardware clock, which is subject to some bounded inaccuracy (called *clock drift*).

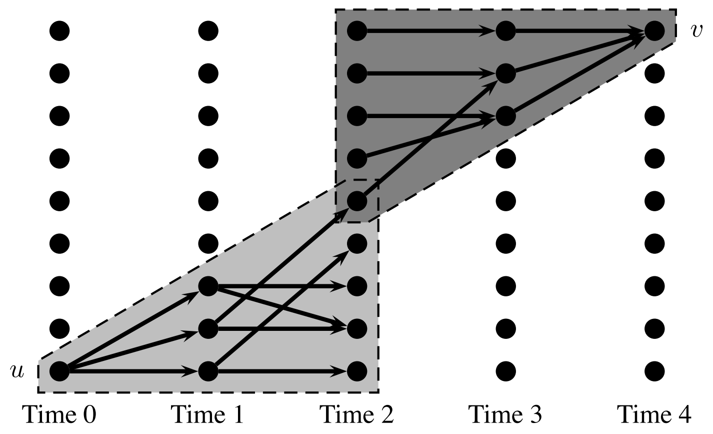

Figure 2: Illustration for the proof of Lemma 3.1, with $n = 9$ and $t = 0$. Each instantaneous graph is 2-connected (vertex growth 2), the smallest d satisfying $\mathrm{g}^{(d)} > 9/2$ is $d = 2$, and the dynamic diameter is 4. The shaded areas indicate $U_{t'}$ for $t' = 0, 1, 2$ in light gray, and $V_{t'}$ for $t' = 2, 3, 4$ in darker gray (with $U_2 \cap V_2 \neq \emptyset$, as argued in the proof). Only edges from nodes in $U_{t'}$ for $t' = 0, 1$ and edges into nodes in $V_{t'}$ for $t' = 3, 4$ are shown.

It is also assumed in this setting that messages suffer from unpredictable delay bounded above by U, and that each node broadcasts at least once every ΔH real time units. The *dynamic diameter* for this partially-synchronous model is defined similarly to the synchronous case: it is an upper bound on the time required for a single message to be flooded throughout the entire network. The technical details of the definition are more complex than for synchronous networks, and we refer to [26] for the precise definition.

In [26] we considered the problem of clock synchronization in this model: the goal is for each node to output a *logical clock*, such that the logical clocks are always well-synchronized (unlike the hardware clocks, which can drift apart over time). It is well-known that the quality of clock synchronization that can be achieved in a static network is closely related to its diameter: no algorithm can avoid a clock difference of $\Theta(D)$ between two distant nodes in the graph, and a clock difference of $\Theta(\log D)$ even between neighboring nodes [31].

For dynamic graphs, the situation is almost the same, replacing D now by the dynamic diameter. There is, however, an additional caveat: when a new communication link appears, it takes $\Theta(D)$ time before the clock difference between its endpoints can be reduced to its final value of $\Theta(\log D)$. Informally, this adjustment time corresponds to the time it takes for information about the new edge to propagate through the network, so that all nodes can adjust their behavior accordingly. In [26] we show that in networks with dynamic diameter D, there is a dynamic clock synchronization algorithm that achieves a clock difference of no more than $O(D)$ between any two nodes at any time in the execution, and a clock difference of $O(\log D)$ between nodes that remain neighbors of each other for $\Theta(D)$ time. The algorithm is based on the optimal static algorithm from [31], but it requires very careful handling of new edges, in order to reduce the clock difference on such edges while at the same time not creating large clock differences on "older" edges that have existed for longer than $\Theta(D)$ time.

3.2 The Diameter of Markovian Dynamic Graphs

Using an argument along the lines of Lemma 3.1, we can derive a bound for the most simple Markovian dynamic graphs. Assume that in each round r, $G(r)$ is a random graph $G_{n,p}$ for some given parameter p; that is, in every round r, every possible edge is present independently with probability p. What is the expected vertex growth of $G_{n,p}$? For any node set $S \subseteq V$ of size $|S| = s$, the probability that a given node $v \notin S$ is not connected to S is $(1-p)^s$ (this is the probability that all s potential edges connecting v to S do not exist). This holds for each of the $n-s$ nodes in $V \setminus S$. Thus we obtain

$$\mathbb{E}\big[|N(S)|\big] = (n-s)\big(1-(1-p)^s\big) > (n-s)\big(1-e^{-ps}\big) \geq (n-s)\frac{\min\{1, ps\}}{2}.$$

We therefore have $\mathrm{g}(s) = \Omega\big(n \cdot \min\{1, ps\}\big)$ "in expectation." Plugging into Lemma 3.1 would lead to a dynamic diameter of $O\big(\log_{1+np}(n)\big)$. Using standard probability-theoretic arguments, it can be shown that indeed this is true with high probability (w.h.p.). Note that for $p = \Omega(1)/n$, we get a logarithmic dynamic diameter; even though the graph is not connected in each round w.h.p. (as p is below the connectivity threshold), still over $O(\log n)$ rounds every node causally influences every other node w.h.p. For constant p, the dynamic diameter becomes a constant. (Closely related to the dynamic diameter of a sequence of independent random graphs is the basic problem of spreading a rumor by a *push-pull gossip protocol*, as introduced in [25]. In such protocols each node chooses a random neighbor to connect to, which induces an independent random graph for that round.)

In [11], Clementi et al. show that the above result generalizes to Markovian dynamic graphs with birth rate p. In addition, they give lower bound on the dynamic diameter for that case.

Theorem 3.2. *[11] For all $0 \leq p, q \leq 1$, the dynamic diameter of a Markovian dynamic graph with birth rate p and* any *death rate q is at most $O\big(\log_{1+pn} n\big)$, independent of q and the initial graph $G(0)$. If $G(0)$ is empty and $p \geq \ln(n)/n$ is above the connectivity threshold of the random graph $G_{n,p}$, this result is asymptotically tight. For general $p > 0$, the dynamic diameter is lower bounded by $\Omega(\log(n)/(np))$.*

Proof Sketch. The upper bound can be shown as for the simpler case where the graph is an independent random graph $G_{n,p}$ in each round. To see this, consider two consecutive rounds r and $r+1$. For an edge $e \notin E(r-1)$, the probability that $e \in E(r)$ is p; for an edge $e \in E(r-1)$, the probability that $e \in E(r) \cup E(r+1)$ is $(1-q)+qp \geq p$. Hence, in the union of any two consecutive rounds, each edge independently occurs with probability at least p. It follows that the dynamic diameter of a Markovian random graph with birth rate p is at most twice as large as the dynamic diameter when the graph in each round is an independent random graph $G_{n,p}$. We have already seen above that the dynamic diameter of such dynamic graphs is w.h.p. $O\big(\log_{1+pn} n\big)$.

For the lower bound, we note that if $E(0) = \emptyset$, the union of the instantaneous graphs in the first d rounds is itself a (static) random graph $G_{n,\hat{p}}$, where $\hat{p} = 1-(1-p)^d$ (the probability that an edge never appears in d rounds). If $p \geq \ln(n)/n$ and $d = \Theta(\log_{np}(n))$, then the diameter of this random graph is $\Theta(\log_{n\hat{p}} n) = \Theta(\log_{np} n)$ [7]. This implies that the diameter of the dynamic graph is $\Omega(\log_{np}(n))$, because in the first d rounds a node can only causally influence nodes at distance at most d from it in the union of the graphs during those d rounds. A somewhat similar, but more involved argument can be made for the case $np < \ln n$. For details, we refer to [11]. □

Subsequent work in [5, 15] analyzes the flooding time of Markovian random graphs with birth rate p and death rate q in the stationary distribution. In that case, the graphs in all rounds are random graphs $G_{n,\hat{p}}$ with edge probability $\hat{p} = p/(p+q)$. If $\hat{p} > \ln(n)/n$, i.e., if $\hat{p}$ is above the connectivity threshold, w.h.p., the dynamic diameter is $\Theta(np/\log(n\hat{p}))$ and thus asymptotically equal to the diameter of $G_{n,\hat{p}}$. The same bound holds for $1 \ll \hat{p} < \ln(n)/n$ and $np \geq \ln(n\hat{p})$, whereas for all other values of $\hat{p}$, the dynamic diameter becomes $\Theta(\log(n)/(np))$.

Clementi et al. have also studied the flooding time of random dynamic geographic graphs [14, 15]. Specifically, in [14], they consider a simple variant of the random way-point model. Nodes are located in a square of side length L. Two nodes are connected by an edge whenever they are at distance at most R. Each node chooses a random destination in the square and moves there with a fixed constant speed s, either by first moving horizontally and then vertically, or by first going vertically and then horizontally. In such a scenario, the network is much denser in the center of the square than in the corners. In [14], it is shown that for sufficiently dense networks (average degree $\Omega(\log n)$) and moderate speeds ($O(1)$ communication radii per time unit), the dynamic diameter of such networks is $O(L/R + L^3 \log(n)/(sR^2 n))$. Note that the required average degree does not guarantee connectivity at each time. Furthermore, the given speed limitation gives an $\Omega(L/R)$ lower bound on the dynamic diameter. The authors point out that there are interesting settings for the parameter values for which the dynamic diameter is strictly larger than $O(L/R)$.

4 Computing in Adversarial Dynamic Graphs

In [28] we considered the problem of general computation in adversarial dynamic graphs, under fairly minimalistic assumptions (some basic results in that direction also appeared in [34]). We assumed that the dynamic graph is always connected (that is, it is 1-interval connected), but nodes do not know in each round what their edges for that round will be; in every round each node generates a message to broadcast, and the adversary then chooses edges for that round and delivers the message to all the sender's neighbors. Further, we assumed that nodes do not initially know the size n of the network, but they do have unique identifiers (UIDs). We asked whether nodes can accomplish tasks such as leader election, determining the number n of nodes in the network, and information dissemination.

Let us say that at time t, node u has *heard from* node v if $(v, 0) \rightsquigarrow (u, t)$. A 1-interval connected graph has vertex growth at least 1; therefore, at time $t < n$, each node has heard from at least $t + 1$ nodes (see Lemma 3.1). If nodes can send large messages, then there is a simple way to determine n. In each round, we have all nodes forward all the UIDs they have heard so far; let $A_u(t)$ denote the set of UIDs collected by node u by time t. At time t, if $|A_u(t)| = t$, then node u halts and outputs t as the count, and if desired, $A_u(t)$ as the set of UIDs participating in the computation.

How do we know, when $|A_u(t)| = t$, that $t = n$ and $A_u(t) = V$? Simple: if $t < n$, then as mentioned above, at time t node u must have heard from at least $t + 1$ nodes; the UIDs of all these nodes are in $A_u(t)$, and hence $|A_u(t)| > t$. In particular, node u cannot stop if $t < n$. On the other hand, at time n we have $|A_u(t)| \geq |A_u(n-1)| = n$; consequently $A_u(n) = V$, and at time n node u stops and outputs n. At that time nodes can also elect a leader using some global decision rule (e.g., the smallest UID), or compute functions of the initial states of the participants, if the initial states are forwarded along with the UIDs.

What if nodes cannot send everything they know in every message? In [28] we show that full information dissemination can be achieved even if only a constant number of UIDs and initial inputs can be sent in each message.

Suppose that our goal is to have each node compute an upper bound on the count (we will return to exact counting and full information dissemination later). We solve this problem via an intermediary problem that we call *k-committee*. In the k-committee problem the nodes must partition themselves into committees (that is, each node outputs a committee ID), such that

(a) Each committee contains no more than k nodes (but can contain fewer than k nodes), and

(b) If $k \geq n$, then there is only one committee containing all the nodes.

The k-committee problem is equivalent to the problem of computing (an upper bound on) the count. One direction is trivial: if the nodes known n, then they can solve k-committee by each forming a singleton committee if $k < n$, and all joining one default committee if $k \geq n$. The other direction is more interesting: if we start from a valid solution for k-committee, we can check whether or not $k \geq n$ using a quick k-round *verification protocol*.

In the verification protocol a node can have two states: positive or negative. Initially all nodes are positive. In each round, positive nodes broadcast their own committee ID, and negative nodes send $\perp$, a special value that differs from all committee IDs. If a positive node hears $\perp$ or a committee ID that differs from its own, it becomes negative, and remains negative thereafter. After k rounds, all positive nodes output "$k \geq n$", and negative nodes output "$k < n$".

To see the correctness of this protocol, suppose first that $k \geq n$. In this case there can only be one committee, so no node ever hears a different committee ID from its own. After k rounds, all nodes are still positive, and all output "$k \geq n$". Next, suppose that $k < n$, and consider some committee c. In each round, if we look at the cut between the members of committee c that are still positive (if any remain) and the rest of the graph, this cut must contain at least one edge, and the endpoint that is not a positive member of c sends on that edge a value that differs from c (either $\perp$ or some other committee ID). Therefore some member of c "drops out" and becomes negative. Thus, in each round where not all members of c are negative, at least one new member becomes negative. Since committee c can initially contain at most k members, after k rounds all members are negative, and all output "$k < n$". This holds for all committees and therefore for all nodes in the graph.

Using the verification protocol we can compute an upper bound on the count: for each $k = 1, 2, 4, 8, \ldots$, we solve k-committee election, then run the verification protocol to check if $k \geq n$. But how do we solve k-committee using only small messages? Our strategy is to use k as a guess for the count, and try to elect k nodes for each committee. We rely on the following observation:

- If $k \geq n$, then the dynamic diameter of the graph is at most $k - 1$ (Lemma 3.1), so $k - 1$ rounds are sufficient for one node u to "invite" any other node v to join its committee, provided that all nodes forward u's invitation when they receive it.

- On the other hand, if $k < n$, then we really do not have to guarantee much; invitations can fail to reach their destination, but all we have to ensure is that *no more than* k nodes join each committee. (Recall that if $k < n$ then even having each node form its own committee is a valid solution.)

We proceed as follows.

(1) First, for $k - 1$ rounds, each node forwards the smallest UID it has heard so far. At the end of the $k - 1$ rounds, any node that has not heard a smaller UID than its own elects itself as a *leader* of a committee (using its own UID for the committee ID). Each leader immediately joins its own committee. Notice that if $k \geq n$, then all nodes hear the smallest UID, and there is only one leader.

(2) Next, the nodes repeat the following $k - 1$ times:

   (a) For $k - 1$ rounds, all nodes forward the smallest UID of a node that has not joined a committee yet.

   (b) Each leader selects the smallest UID it has heard in (a), and issues an *invitation* to this node to join its committee. The invitation is a message carrying the leader's UID and the UID of the invited node.

(c) For $k-1$ rounds, all nodes forward the invitations they hear; if a node hears invitations from more than one leader, it chooses which one to forward arbitrarily (or just forwards none of them).

(d) Each node that receives an invitation addressed to itself in (c) joins the committee of the node that invited it. If a node is invited to join more than one committee, it chooses arbitrarily.

Since each node can only invite at most $k-1$ other nodes to join its committee, the size of each committee is at most k. If $k \geq n$, then we have just one leader, whose invitations all reach their destinations; in $k-1 \geq n-1$ cycles this leader successfully identifies and invites all the other nodes in the graph.

Notice that when $k \geq n$, each node has $n-1$ rounds where it is "singled out" and has its UID forwarded to all nodes in the graph. If nodes remember all the UIDs they hear, they can collect the UIDs of all the participants this way; if nodes have initial inputs, they can attach them to their UIDs, and in this manner we can achieve full information dissemination and exact counting.

The protocol above requires $O(n^2)$ rounds for full information dissemination in 1-interval connected graphs. If the graph enjoys good expansion, then it can take fewer than $n-1$ rounds to invite each node, and the process can be sped up; if the graph is T-interval connected for $T > 1$, then we can pipeline invitations and invite T nodes in $O(n)$ rounds, gaining a factor of T in the round complexity. We refer to [28] for the full details.

5 Dynamic Radio Networks

An important application domain for dynamic network algorithms is wireless mobile ad hoc networks. When communication occurs via wireless radio, the communication model used in the previous sections does not directly apply any more; because all messages are sent over the same shared wireless medium, messages from different senders can collide and interfere. This characteristic is captured by the radio network model as for example defined in [4]. A message sent by some node u reaches all of u's neighbors. A neighbor v can decode the message if and only if no other neighbor of v transmits a message at the model as defined (for example) in [4]. Furthermore, it is typically assumed that a receiver cannot distinguish a message collision from silence.

Consider the basic problem of broadcasting a single message throughout the network. Assume that we have a synchronous dynamic graph $G = (V, E)$ that is not necessarily connected in every round, and that some source node $s \in V$ has a message to broadcast. We say that a node is *informed* if it already received the message.

In [12], it is assumed that in every round, there is some uninformed node v connected to some set of nodes that already know the message. (Note that this requirement on the dynamic graph adversary depends on the algorithm in question, and even on the specific execution of the algorithm. However, it is implied by 1-interval connectivity.) A simple randomized protocol for broadcast, given in [12], is to have each informed node transmit the message with probability p in each round. If an uninformed node v is connected to k informed nodes, the probability that it successfully receives the message is $kp(1-p)^{k-1}$ (the probability that one informed neighbor broadcasts and the rest do not). If we choose $p = \ln(n)/n$, this success probability is $\Theta(\log n/n)$ independent of k. Since at most $n-1$ hops are required to reach all nodes, the broadcast completes in $O(n^2/\log n)$ rounds w.h.p. In [12] it is also shown that there is an adaptive adversary for which this bound is tight.

An interesting speed-up can be achieved if we assume that the dynamic graph $G = (V, E)$ is more stable. If G is ∞-interval connected, i.e., if there is a connected spanning sub-graph that is stable, we have shown in [27] that broadcast is possible in almost-linear time. To do this we have nodes that have known the message for $\Theta(t \log n)$ rounds transmit with probability roughly $1/t$, and we show that broadcast is completed in $O(n \log^2 n)$ rounds w.h.p. The main idea is the following: there is always some node v at the boundary between informed and uninformed nodes in the stable sub-graph. Whenever node v transmits alone among all the nodes in the network, the algorithm makes progress. Intuitively, in order to delay this kind of progress for a long time, the adversary has to allow the message to reach many additional nodes, so that these nodes will interfere and cause collisions whenever v transmits. Because broadcast probabilities decay with time, we can argue that either node v broadcasts alone, or many other nodes recently received the message, and in both cases the algorithm makes amortized progress.

In [13, 27] it is also shown that in ∞-connected dynamic radio networks, broadcast can be solved deterministically. If the nodes have unique identifiers from a space of size $O(n)$, the respective bounds are $\Omega(n^{3/2})$ [13] and $O(n^{3/2}\sqrt{\log n})$ [27]. Note that if we drop the ∞-interval connectivity requirement, an adaptive adversary can prevent the broadcast message from ever reaching more than two nodes.

Finally, Clementi et al. study radio network broadcast in random dynamic graphs in [12]. If for some fixed and generally known parameter $1/n \leq p \leq 1$, the graph for each round is an independent random graph $G_{n,p}$, they show that the time to do broadcast is $\Theta(\log n)$. If p is not known to the algorithm, it has to guess different values for p, at the cost of a logarithmic factor in the overall performance.

6 Other Models for Dynamic Networks and Dynamic Network Algorithms

In this section we briefly survey several alternatives to the models we discussed in this column. This section is not intended as a comprehensive survey; for brevity, we only list a few of the prominent alternatives, and give only one or two examples of each approach.

Liveness and termination. The algorithms we focused on in this column always terminate, even when the network changes continually. There are (at least) two popular alternative modes of computation.

The first alternative is to use *self-stabilizing algorithms*. The basic assumption here is that changes are bounded in time and eventually cease, or that changes are infrequent enough that the algorithm can stabilize between them. While changes continue to occur, the algorithm can behave badly; however, if changes cease for sufficiently long, a self-stabilizing algorithm will eventually gravitate to a correct configuration and resume normal operation. We refer to [16] for a comprehensive treatment of self-stabilizing algorithms. Some algorithms provide a two-part correctness guarantee: a *safety* guarantee, which holds even while the network is dynamic, and a *liveness* (or termination) guarantee, which only holds when the network stops changing. For example, algorithms based on Gafni and Bertsekas' link reversal algorithm from [19] frequently have this flavor; the link reversal algorithm eventually orients a directed network so that all nodes have a path to some fixed sink node, but the time required is $\Theta(n^2)$ rounds [19, 9], and during this time nodes cannot necessarily route messages to the sink. One example that uses link-reversal is [37], which gives a mutual exclusion algorithm in which nodes pass a single token among themselves to determine who can enter the critical section; link reversal is used to route the token from node to node. Safety (mutual exclusion) is always guaranteed because only one node has the token, but liveness (in this case starvation-freedom) is only guaranteed when the network stops changing.

A second alternative to always-terminating computation is *eventually stabilizing computation*. Here it is not assumed that the network eventually stops changing. However, the algorithm never terminates; instead, the outputs of the nodes are continually updated, and they eventually stabilize to a correct answer. (Note that the nodes themselves do not *know* when a correct answer has been reached, so nodes cannot stop executing the algorithm.) A prominent example is *population protocols* [1], where participants are anonymous and have finite (uniformly-bounded) memory; in this model it is possible to have the nodes' outputs stabilize to any semi-linear predicate of the inputs (see [2] for a survey on population protocols).

Dynamic overlay networks. The models we discussed here assume that the network topology is completely adversarially controlled. There is a vast literature on algorithms that choose the communication topology themselves, typically in the form of an *overlay network* that the algorithm superimposes on an underlying fully-connected network. One very popular application domain is *peer-to-peer protocols*, where participants must maintain a small number of connections to peers they select, e.g., [35, 36]. The algorithm must deal with nodes joining and leaving the network (*churn*) by adjusting the overlay network. Examples of protocols that deal with continual concurrent joins and leaves controlled by an adaptive worst-case adversary are given, for instance, in [29, 32]. In the examples given above, the overlay changes only as a response to changes in the set of participating peers; other algorithms, such as those given in [20, 22, 33], induce a constantly-changing overlay even when the set of participants remains static.

Geographic models. Much of the work on dynamic networks is motivated by an underlying distribution of the nodes on the two- or three-dimensional plane, with some geometric constraint governing which nodes have a communication link to each other (perhaps the most popular assumption is that only nodes within some fixed distance of each other can directly communicate). From time to time the nodes change their positions according to some *mobility pattern*, either adversarially controlled or random (e.g., the random way-point model [6, 24, 30]; we discussed one variant of this model in Section 3.2). We refer to [10] for a survey.

7 Conclusion

In this column we briefly surveyed several models for dynamic networks. The main characteristic shared by these models is that they do not assume that the network stops changing at some point, and they do not restrict the number or frequency of changes. Nevertheless, recent results show that always-terminating computation is possible in such networks, even when the network is adversarially controlled.

There are many open research questions in dynamic networks. One major concern is dealing with node failures, or with nodes joining and leaving the network; most of the algorithms we discussed here are not resilient to such changes. For example, in some of the network-wide broadcast algorithms described in Section 5, nodes stop forwarding the message after a certain number of rounds. At some points in the execution there might only be a few nodes still forwarding the message, and if these nodes were to crash, the broadcast would fail. Can broadcast algorithms be made resilient to node crashes, while still completing in a timely manner and allowing nodes to eventually stop sending? As another example, the information dissemination algorithm from Section 4 cannot withstand node failures, nor can it handle new nodes joining the network. Can information dissemination be solved in networks that enjoy good vertex growth (e.g., networks that are always k-connected), despite changes to the set of nodes participating in the computation?

Another interesting direction is testing and adapting to network conditions. Throughout this column we saw that if the network enjoys good expansion or long-term stability, algorithms can be made to run faster. But how do we know if the network is a good expander, or even if it is 1-interval connected? How do we know if it is fairly stable over time? Lightweight tests for such properties, which can run alongside an algorithm and inform it of the condition of the network in the recent past, could allow algorithms to take advantage of a well-behaved network.

In general, it seems that a promising direction for dynamic network algorithms is to cope with the worst, and hope for the best: an algorithm should be able to withstand a badly-behaved and highly dynamic network topology, and always guarantee correctness and termination, albeit not necessarily good performance; if the network is well-behaved for a period of time (e.g., it enjoys some stability, or has good expansion), the algorithm should take advantage of this to improve its performance.

Acknowledgement. The authors would like to thank the Editor, Idit Keidar, for many insightful comments and suggestions.

References

[1] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *Proc. of 23rd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 290–299, 2004.

[2] J. Aspnes and E. Ruppert. An introduction to population protocols. In B. Garbinato, H. Miranda, and L. Rodrigues, editors, *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. Springer-Verlag, 2009.

[3] 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 *Proc. of 35th Coll. on Automata, Languages and Programming (ICALP)*, pages 121–132, 2008.

[4] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *J. of Computer and System Sciences*, 45(1):104–126, 1992.

[5] H. Baumann, P. Crescenzi, and P. Fraigniaud. Parsimonious flooding in dynamic graphs. In *Proc. of 28th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 260–269, 2009.

[6] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2:257–269, 2003.

[7] B. Bollobás. The diameter of random graphs. *Trans. of the American Mathematical Society*, 267(1):41–52, 1981.

[8] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer. Brahms: Byzantine resilient random membership sampling. *Computer Networks*, 53(13):2340–2359, 2009.

[9] C. Busch and S. Tirthapura. Analysis of link reversal routing algorithms. *SIAM J. on Computing*, 35:305–326, 2005.

[10] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communication and Mobile Computing*, 2(5):483–502, 2002.

[11] A. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time in edge-Markovian dynamic graphs. In *Proc. of 27th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 213–222, 2008.

[12] A. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Broadcasting in dynamic radio networks. *Journal of Computer and System Sciences*, 75(4):213–230, 2009.

[13] A. Clementi, A. Monti, and R. Silvestri. Round robin is optimal for fault-tolerant broadcasting on wireless networks. *J. of Parallel and Distribited Computing*, 64(1):89–96, 2004.

[14] A. Clementi, A. Monti, and R. Silvestri. Fast flooding over Manhattan. In *Proc. of 29th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 375–383, 2010.

[15] A. Clementi, F. Pasquale, A. Monti, and R. Silvestri. Information spreading in stationary Markovian evolving graphs. In *Proc. of IEEE Symp. on Parallel & Distributed Processing (IPDPS)*, 2009.

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

[17] A. Ferreira. Building a reference combinatorial model for MANETs. *IEEE Network Magazine*, 18(5):24–29, 2004.

[18] A. Ferreira, A. Goldman, and J. Monteiro. Performance evaluation of routing protocols for MANETs with known connectivity patterns using evolving graphs. *Wireless Networks*, 16(3):627–640, 2009.

[19] E. M. Gafni and D. P. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, 29:11–18, 1981.

[20] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. SCAMP: Peer-to-peer lightweight membership service for large-scale group communication. *Networked Group Communication*, pages 44–55, 2001.

[21] P. Grindrod and D. J. Higham. Evolving graphs: dynamical models, inverse problems and propagation. *Proc. of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 466(2115):753–770, 2009.

[22] M. Gurevich and I. Keidar. Correctness of gossip-based membership under message loss. *SIAM J. of Computing*, 39(8):3830–3859, 2010.

[23] S. Janson, T. Łuczak, and A. Rucinski. *Random graphs*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York, 2000.

[24] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic, 1996.

[25] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. *Proc. of 41st Symp. on Foundations of Computer Science (FOCS)*, pages 565–574, 2000.

[26] F. Kuhn, C. Lenzen, T. Locher, and R. Oshman. Optimal gradient clock synchronization in dynamic networks. In *Proc. of 29th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 430–439, 2010.

[27] F. Kuhn, N. Lynch, C. Newport, R. Oshman, and A. Richa. Broadcasting in unreliable radio networks. In *Proc. of 29th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 336–345, 2010.

[28] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proc. of 42nd Symp. on Theory of Computing (STOC)*, pages 513–522, 2010.

[29] F. Kuhn, S. Schmid, and R. Wattenhofer. A self-repairing peer-to-peer system resilient to dynamic adversarial churn. In *Proc. of 4th Int. Workshop on Peer-To-Peer Systems (IPTPS)*, 2005.

[30] J.-Y. Le Boudec and M. Vojnovic. The random trip model: Stability, stationarity regime, and perfect simulation. *IEEE/ACM Transactions on Networking*, 16(6):1153–1166, 2006.

[31] C. Lenzen, T. Locher, and R. Wattenhofer. Tight bounds for clock synchronization. In *Proc. of 28th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 46–55, 2009.

[32] X. Li, J. Misra, and C. G. Plaxton. Maintaining the ranch topology. *J. of Parallel and Distributed Computing*, 70(11):1142–1158, 2010.

[33] P. Mahlmann and C. Schindelhauer. Peer-to-peer networks based on random transformations of connected regular undirected graphs. In *Proc. of 17th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 155–164, 2005.

[34] R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *Proc. of Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 104–110, 2005.

[35] C. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proc. of 9th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, 1997.

[36] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001.

[37] J. E. Walter, J. L. Welch, and N. H. Vaidya. A mutual exclusion algorithm for ad hoc mobile networks. *Wireless Networks*, 7:585–600, 2001.