Distributed Computing in Fault-Prone Dynamic Networks *

Philipp Brandes
Computer Engineering and Networks Lab (TIK)
ETH Zurich
pbrandes@ethz.ch

Friedhelm Meyer auf der Heide
Heinz Nixdorf Institute & Department of Computer Science
University of Paderborn
fmadh@upb.de

ABSTRACT

Dynamics in networks is caused by a variety of reasons, like nodes moving in 2D (or 3D) in multihop cellphone networks, joins and leaves in peer-to-peer networks, evolution in social networks, and many others. In order to understand such kinds of dynamics, and to design distributed algorithms that behave well under dynamics, many ways to model dynamics are introduced and analyzed w.r.t. correctness and efficiency of distributed algorithms. In [16], Kuhn, Lynch, and Oshman have introduced a very general, worst case type model of dynamics: The edge set of the network may change arbitrarily from step to step, the only restriction is that it is connected at all times and the set of nodes does not change. An extended model demands that a fixed connected subnetwork is maintained over each time interval of length T (T-interval dynamics). They have presented, among others, algorithms for counting the number of nodes under such general models of dynamics.

In this paper, we generalize their models and algorithms by adding random edge faults, i.e., we consider fault-prone dynamic networks: We assume that an edge currently existing may fail to transmit data with some probability p. We first observe that strong counting, i.e., each node knows the correct count and stops, is not possible in a model with random edge faults. Our main two positive results are feasibility and runtime bounds for weak counting, i.e., stopping is no longer required (but still a correct count in each node), and for strong counting with an upper bound, i.e., an upper bound N on n is known to all nodes.

1. INTRODUCTION

Dynamic networks have received a lot of attention in the recent past. They appear in different scenarios like nodes moving in 2D (or 3D) in multihop cellphone networks, joins and leaves in peer-to-peer networks, evolution in social networks, and many others. In order to understand such kinds of dynamics, and to design distributed algorithms, many ways to model dynamics are introduced and analyses w.r.t. correctness and efficiency of distributed algorithms. In [16], Kuhn, Lynch, and Oshman have introduced very general, worst case type models of dynamics. In their *basic model*, the set of edges of the network may change arbitrarily from step to step, but a connected graph must be maintained. The node set does not change, will always have n elements. An extension of this model demands that a fixed connected subnetwork is maintained over each time interval of length T (T-*interval dynamics*). They presented, among others, algorithms for counting the number of nodes under such general models of dynamics, and prove an $\mathcal{O}(\frac{n^2}{T} + n)$ bound for the number of steps needed.

Counting is a very basic task and a requirement for many, more complex tasks like, e.g., the all-to-all token dissemination in which every node starts with exactly one token which has to be spread to every other node in the network. Without a correct count, the nodes cannot know whether they have received every other token.

The aim of this paper is to extend the above-mentioned models of dynamics from [16] by introducing random edge faults, i.e., by assuming (besides the topology changes as in their models) that, in each step, each edge may fail with some probability p. Our main contributions are impossibility results and algorithms for distributed counting under T-interval dynamics and random edge faults.

1.1 Further Related Work

Dynamic networks have been in the focus for some time now. Usually these networks are not allowed to be fully dynamic in the sense as described above. One common constraint is that every edge that existed reappears infinitely often which is used e.g., in [4]. A more specific variant is requiring some sort of periodicity as in [8] or in [9] for network exploration. The notion closest to ours is used in [17, 20] requiring connectivity in every round but has no further constraints in which some basic results for this type of network are shown. A network property similar to T-interval dynamics is introduced in [14]. A unifying framework for time-varying graphs is presented in [5]. Broadcasting in dynamic networks has also been considered in radio networks where interference comes into play [6].

There is already some work on counting and information dissemination. An overview is in [10] and more recently in [2]. General works in the area of distributed algorithms are [18, 22]. Information spreading has been considered as part

*This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre *On-The-Fly Computing* (SFB 901).

TADDS December 17 2012, Roma, Italy

of gossiping algorithm, e.g., in [12] where only one message has to be spread throughout the network.

Somewhat related is gossip-based information aggregation in [13], but they assume a random process to model dynamics. Information aggregation in a static network has been studied as well, e.g., in [15, 19, 21]. This allows to compute any function, depending on some initial values, at one node. Edge failures and leaves/joins of nodes are considered in [18] and [2]. The concept of eventual stabilization and self-stabilizing algorithms is used in [1, 3, 11]. An overview can be found in [7]. In these scenarios the network stops changing at some point. Thus, the algorithms are only required to compute their result from an arbitrary starting point but in a static environment.

1.2 Our Contributions

In this paper we extend the T-interval dynamics by introducing random edge faults, i.e., each edge of the current graph may fail temporarily (i.e., in the current step) with some probability $p < 1$, where p is known to all nodes. We differentiate between two notions of distributed counting:

Strong Counting An algorithm for strong counting has a runtime bound $t(n)$ such that each node stops with the correct count n within $t(n)$ steps.

Weak Counting An algorithm for weak counting has a runtime bound $t(n)$ such that each node has the correct count n after $t(n)$ steps, but the execution of the algorithm does not necessarily stop.

We assume that, in case of random edge faults, the runtime bounds is always guaranteed, but the correctness ("all nodes know the correct count n") only holds with some high probability.

Due to space limitations, we will only consider T-interval dynamics. The simpler algorithms (both with and without random edge faults) for the basic model (1-interval dynamics) are not mentioned here. The specializations of the results mentioned below for $T = 1$ yield similar results.

In [16] it is shown that strong counting in fault-free networks under T-interval dynamics can be done in time $t(n) = \mathcal{O}\left(\frac{n^2}{T} + n\right)$.

In this paper we show the following results. Let us first assume that p is known to the nodes.

- Strong counting is not possible, if there are random edge faults, even in static networks.
- Weak counting is possible under T-interval dynamics and random edge faults with failure probability $p \in (0,1)$ in $\mathcal{O}\left(\frac{n^2}{T} \cdot \left(\frac{\log T}{\log(\frac{1}{p})}\right)^2 \frac{1}{1-p}\right)$ if $p > \frac{1}{T}$, and $\mathcal{O}\left(\frac{n^2}{T}\right)$ else.
- Strong counting is possible under T-interval dynamics and random edge faults with failure probability $p \in (0,1)$ if the nodes know an upper bound N on the number n of nodes. The runtime $t(n, N)$ is by an additive term $\mathcal{O}(\log(\frac{1}{p}) \cdot n \cdot \log(N))$ larger than for weak counting.

1.3 Organization of the Paper

In Section 2, the models and the algorithms for counting by Kuhn, Lynch, and Oshman are introduced. In Section 3, we observe that strong counting is no longer possible if random edge faults are allowed. In Section 4, we present and analyze a modified version of the disseminate procedure from [16] which can cope both with T-interval dynamics and random edge faults. Section 5 presents our two counting algorithms, Section 5.1 considers the case "p is unknown".

2. THE DYNAMIC NETWORK MODELS BY KUHN, LYNCH, AND OSHMAN

In this section we formally describe the models for dynamic networks introduced in [16] and sketch their algorithms used to solve the counting problem.

We consider a network with a set of n nodes each with a unique identifier (UID). The topology of this network can be changed by an adversary in every step, the only restriction is that it has to be connected at all times. In a step, each node broadcasts some message to its neighbors. A message is short, may consist of at most a constant number of UIDs. The nodes broadcast without any information about their neighbors; no neighborhood discovery is used. A more restricted model for dynamics is T-interval dynamics: in every time interval of length T, a stable connected subgraph persists. A major contribution of [16] is to demonstrate how to use this restriction on dynamics for speeding up distributed counting.

2.1 Distributed Algorithms for Counting

In this subsection we briefly describe the distributed algorithm for counting the number n of nodes under T-interval dynamics described in [16]. This algorithm, which is executed by every node, consists of two building blocks, namely the procedures *k-Committee Election* and *k-Verification*.

k-Committee Election partitions the node set in committees, i.e., nodes which have chosen the same committee UID (C-UID), each committee of size at most k. Further, if $k \geq n$, then there is only one committee and each node knows all n UIDs.

k-Verification starts with a partition of the node set produced by k-Committee Election. Every node v knows k and has a binary variable x_v initially set to 1. If there is only one committee, *all* x_v remain unchanged. If there is more than one committee, *all* x_v will be 0 at the end of the procedure.

The counting algorithm now starts with $k = 2$ and doubles k in each iteration. For each such k, it executes k-Committee Election followed by k-Verification. The next iteration only starts if $x_v = 0$ holds for all nodes v. By the above, the algorithm stops as soon as $k \geq n$. In this situation, each node outputs the number of UIDs it knows. By the description above, this means that each node outputs the correct number n of nodes.

k-Verification executes k steps. In each step, each node v with $x_v = 1$ broadcasts its C-UID. If it hears from some neighbor a different C-UID or the special symbol $\perp$, it sets x_v to 0 and broadcasts $\perp$ from now on. The correctness is shown in [16].

k-Committee Election is based on the procedure ***Disseminate*** (A, k) [16]. Here A is a set of at least T UIDs, arbitrarily distributed among the nodes; k is a positive integer known to all nodes. Disseminate(A, k) then makes sure

that each of the T smallest of the UIDs from A is known to at least k nodes. In particular, if $k \geq n$ then all nodes know at least the T smallest UIDs from A. The original procedure Disseminate(A, k) is equivalent to Algorithm 1 with $l = \frac{T}{k}$ and $x = 1$ and thus contains two loops. The set A contains all the UIDs the node knows whereas S denotes all the UIDs, which it has already broadcast in this inner loop. Thus, after each step the set A gets updated with the newly received UIDs. To the set S the most recently broadcast value is added. Thus, in each step the smallest value, which has not already been sent in this execution of the loop, is selected and broadcast.

In [16] it is shown that this algorithm correctly executes Disseminate(A, k) and needs $\lceil k/T \rceil \cdot 2T$ (parallel communication-) steps, in each of which only single UIDs are broadcast by the nodes.

k-Committee Election can be done by repeatedly executing Disseminate(A, k) two times. First, the T smallest UIDs of nodes not yet in a committee are broadcast to k nodes. Afterwards, each node whose UID is the smallest it knows, invites T nodes to join its committee. For this, it forms T tokens of the kind (a, b), where a is its own UID and b a UID it knows. A node with UID b that receives at least one token of the form (a, b) joins the committee of the node a – if there are several candidates for a, choose the smallest. We repeat the two executions of Disseminate(A, k) $\lceil k/T \rceil$ times to allow the leader to issue up to k invites. A node that does not receive any token, forms its own committee. This procedure creates committees of size at most k. Further, if $k \geq n$, then there is only one committee and each node knows all n UIDs.

3. STRONG COUNTING IS NOT POSSIBLE UNDER RANDOM EDGE FAULTS

In this section we observe that the strong counting result from [16] is impossible if we allow random edge-faults, even in static networks.

THEOREM 1. *Strong counting with (known) edge-failure probability $p \in (0, 1)$ is not possible, even if we only demand constant success probability.*

PROOF. Assume there is a strong counting algorithm with runtime bound $t(n)$ that counts correctly with success probability at least $1 - e^{-1} > 0$, given edge-failure probability $p \in (0, 1)$. Now we fix some n and consider a ring of size $2T(n)$. We say that a pair of edges (e_1, e_2) is in distance n if its removal results in two paths one of which has n nodes. If we choose $T(n) \geq (\frac{1}{p})^{2t(n)}$, the following holds:

CLAIM 1. *With probability at least $1 - e^{-1} > 0$, there is a pair of edges (e_1, e_2) in distance n which is faulty during the first $t(n)$ steps.*

PROOF. The probability that a fixed pair of edges is faulty during the first $t(n)$ steps is $p^{2t(n)}$. As there are $T(n)$ pairwise disjoint pairs of edges in distance n, we may conclude:
Pr($\exists$ a pair of edges (e_1, e_2) in distance n which is faulty during the first $t(n)$ steps) $\geq 1-(1-p^{2t(n)})^{T(n)} \geq 1-e^{-1} > 0$. □

Now consider the path P of length n between such two edges. The nodes in P never have contact to nodes outside P during the first $t(n)$ steps. Thus, they will stop and output the wrong count n after $t(n)$ steps, a contradiction to the correctness of the count. □

4. DISSEMINATION UNDER T-INTERVAL DYNAMICS AND EDGE FAULTS

The contribution of this section is to introduce a modified version of the procedure Disseminate(A, k) that can cope with random edge faults for any connectivity parameter T. Assume such faults appear with some probability p which is known to the nodes. The task of Disseminate(A, l, x) is to perform s-dissemination for some number s, i.e., to disseminate each of the s smallest UIDs from A to at least $\min\{k, n\}$ nodes. The procedure uses two integer-valued parameters x and l, with $x < T$ and extends the original one in the following respects:

- The outer loop is executed l times instead of $\frac{T}{k}$ times.
- In each execution of the inner loop, Line 7-9 is executed x times instead of just once, each time with the same UID b to be broadcast. *(Reducing the failure property of a transmission along an edge from p to p^x.)*
- The inner loop only consists of $\frac{2T}{x}$ instead of $2T$ rounds. *(Ensuring that an execution of the inner loop has $2T$ steps, i.e., a stable connected subgraph is present in the (underlying, fault-free) graph during the whole execution of the inner loop.)*

The pseudo-code can be seen in Algorithm 1. Our first task

Algorithm 1 Procedure Disseminate(A, l, x)

```
1: S ← ∅
2: for i = 1, ..., l do
3:   for r = 1, ..., 2T/x do
4:     if S ≠ A then
5:       b ← min(A \ S)
6:       for k = 1, ..., x do
7:         broadcast b
8:         receive b_1, ..., b_y from neighbors
9:         A ← A ∪ {b_1, ..., b_y}
10:      end for
11:      S ← S ∪ {b}
12:    end if
13:  end for
14:  S ← ∅
15: end for
```

will be to provide suitable values for l and x, i.e., for the number of runs of the two loops, so that high success probability for the Disseminate procedure is guaranteed. For some "easy" combinations of p and T, the choices $l = T/k$ and $x = 1$, i.e., the original algorithm for the fault-free case, is good enough. For other combinations of p and T we will derive suitable choices for l and x. These results are then applied to counting in Section 5. These results assume p to be known to the nodes.

Notations: We refer to the i'th execution of the inner loop as the i'th run. A run consists of $\frac{2T}{x}$ rounds, each of which consists of x steps. G_i denotes the stable connected graph that exists during the whole i'th run.

In order to prove our results, consider the i'th run of the procedure and a UID b^* which is among the s smallest not

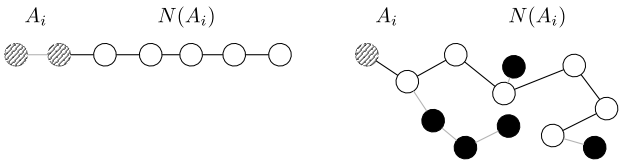

Figure 1: Nodes from set A_i are striped and nodes from the set $N(A_i)$ in white, other nodes in black. The edges from $E(A_i)$ in black and other edges in gray.

yet disseminated at the beginning of this run. We want to define a condition which is sufficient to guarantee that in this run, at least s new nodes are informed about b^*.

In order to define such a sufficient condition, let A_i denote the set of nodes informed about b^* at the beginning of the i'th run. Let $E(A_i)$ be some fixed set of s edges of G_i so that each connected component of $E(A_i)$ contains at least one node from A_i. Consider a set $N(A_i)$ of $\min\{s, n-|A_i|\}$ nodes that are incident to $E(A_i)$ but not in A_i.Note that such a set always exists, and that each node of $N(A_i)$ is connected to some node in A_i via some path composed of edges from $E(A_i)$. Examples are shown in Figure 1. Consider the binary random variable X_i which is true if each edge $e \in E(A_i)$ is fault-free in at least one of the x steps of each of the first $2s$ rounds of the i'th run. With this notation, the following lemma gives the desired sufficient condition.

LEMMA 1. *If X_i is true, then all $\min\{s, n-|A_i|\}$ nodes from $N(A_i)$ are informed about b^* during the i'th run.*

PROOF. If X_i is true, then each transmission of b^* along an edge $e \in E(A_i)$ in one of the first $2s$ rounds is successful, because at least one of the x attempts is successful. A proof analogous to the one for Lemma 6.1 from [16] shows that b^* is forwarded along all these edges within the first $2s$ rounds. Thus, all $\min\{s, n-|A_i|\}$ nodes from $N(A_i)$ are informed about b^* after these $2s$ rounds. □

It is easy to calculate the probability that X_i holds.

LEMMA 2. *Assume $T \geq 2$.*

(i) For $p > \frac{1}{T}$, $\Pr(X_i) \geq (e^{-1}(1-p))$ holds, if $s = \frac{T}{2\log(T)} \cdot \log(\frac{1}{p})$ and $x = 2 \cdot \frac{\log(T)}{\log(\frac{1}{p})}$.

(ii) For $p \leq \frac{1}{T}$, $\Pr(X_i) \geq \frac{1}{2}e^{-1}$ holds for $x = 2$ and $s = \frac{T}{2}$.

PROOF. Clearly, $\Pr(X_i) = (1-p^x)^{2s^2}$.

Thus, $\Pr(X_i) = (1-p^x)^{2s^2} \geq (1-p^x)^{(\frac{1}{p})^x(2s^2p^x)} \geq (e^{-1}(1-p^x))^{2s^2p^x} \geq (e^{-1}(1-p))^{2s^2p^x}$. Using that for all $y \in (0,1)$, $(1-y)^{1/y} \geq e^{-1}(1-y)$ holds. If $p > \frac{1}{T}$, the choices of x and s from (i) therefore imply that $\Pr(X_i) \geq (e^{-1}(1-p))$. The result for $p \leq \frac{1}{T}$ follows similarly. □

The following observation allows us to apply Chernoff bounds.

LEMMA 3. *The random variables $X_1, \cdots, X_l$ are independent.*

PROOF. Fix an arbitrary family S of subsets $A'_1, \cdots A'_l$ of the node set V. Define the random variables $X_1^S, \cdots, X_l^S$ for this sequence in the same way as we defined $X_1, \cdots, X_l$ for the (random) family $A_1, \cdots, A_l$ before. Clearly, $\Pr(X_1^S \wedge \cdots \wedge X_l^S) = \prod_{i=1}^{l} \Pr(X_i^S) = (1-p^x)^{2s^2 \cdot l}$. As this identity holds for every such family, it also holds for the family produced by a run of the dissemination procedure. □

THEOREM 2. *If $p > \frac{1}{T}$, then the procedure Disseminate (A, l, x) executes s-dissemination for $s = \frac{T}{2\log(T)}\log(\frac{1}{p})$ with probability at least $1 - e^{-\frac{k}{2T}}$. It obeys the runtime bound $\mathcal{O}\left(k \cdot \frac{(\log T)}{(\log(\frac{1}{p}))} \cdot \frac{1}{1-p}\right)$. If $p \leq \frac{1}{T}$, then it executes s-dissemination for $s = \frac{T}{2}$ with probability at least $1 - e^{-\frac{k}{2T}}$ obeying the runtime bound $\mathcal{O}(k)$.*

PROOF. Choose $l = l_k := 2 \cdot \frac{1}{1-p} \cdot e \cdot \frac{k}{s}$ and define $X := \sum X_i$.

Then Lemma 2 yields that $E(X) \geq 2 \cdot \frac{k}{s}$. As shown in Lemma 3, we may apply the Chernoff bound. Choosing $\epsilon = \frac{1}{2}$, we obtain: $\Pr(X \leq \frac{k}{s}) \leq \Pr(X \leq \frac{1}{2}E(X)) \leq e^{-\frac{k}{4s}} \leq e^{-\frac{k}{2T}}$.

Plugging in the values for s from Lemma 2 yields the theorem. Let us remark that the spreading of different messages can, as in [16], be considered independent of each other. □

5. COUNTING UNDER T-INTERVAL DYNAMICS AND EDGE FAULTS

In this section, we present our two counting algorithms.

Weak counting is easy: Running the modified procedure Disseminate(A, l, x) for $\frac{k}{s}$ times results in a k-dissemination. Now running this k-dissemination for $k = 2, 4, 8, \ldots$ guarantees that, as soon as $k \geq n$, all nodes know all UIDs and can output the correct count n. Using Theorem 2 yields:

THEOREM 3. *The above procedure executes weak counting. If $p > \frac{1}{T}$, then all node output the correct count n after $\mathcal{O}\left(\frac{n^2}{T} \cdot \left(\frac{\log T}{\log(\frac{1}{p})}\right)^2 \cdot \frac{1}{1-p}\right)$ steps. If $p \leq \frac{1}{T}$, they do so after $\mathcal{O}\left(\frac{n^2}{T}\right)$ steps. The bounds hold with probability at least $1 - e^{-\frac{n}{2T}}$.*

Now we show that strong counting is possible if the nodes know an upper bound N on the number n of nodes. We follow the algorithm from [16]. As sketched in Section 2, the k-dissemination used for weak counting can also be used for creating k-committees. Thus, in order to follow the algorithmic idea from [16], we need a k-Verification that can cope with edge faults.

LEMMA 4. *Assume that some number $N \geq n$ is known to all nodes. Let $\alpha > 1$. Then, running k-Verification for $(\frac{\log N}{\log(\frac{1}{p})(\alpha+2))} + 1)(k-1) = \mathcal{O}(k \cdot \frac{\log N}{\log(\frac{1}{p})})$ steps yields:*

- *If $k < n$ then, with probability $\geq 1 - n^{-\alpha}$, all nodes v finally know that there is more than one committee. ($x_v = 0$).*
- *If $k \geq n$, then all nodes v know finally that there is only one committee ($x_v = 1$).*

PROOF. The second claim is clear.

Let $k < n$. Consider a fixed committee C of size $k' \leq k$. During the execution of the algorithm, we always refer to C as the subset of the original set C which consists of those nodes that have not yet heard about another committee. In the fault-free case, C shrinks in every step, as long as C is not empty. This holds because, in each step, the current graph contains an edge that connects some node of the current C to some node outside the current C. Thus, in case of random edge-faults, in each step t in which $C \neq \emptyset$ holds, $\Pr(C \text{ does not shrink in step } t) \leq p$. Thus,

$$\begin{aligned} & \Pr(C \neq \emptyset \text{ after } c(k'-1)\text{steps}) \\ \leq\ & \Pr(\text{There are at least } c(k'-1)-(k'-1) \\ & \text{steps, in which } C \neq \emptyset \text{ and does not shrink}) \\ \leq\ & \binom{c(k'-1)}{c(k'-1)-(k'-1)} \cdot p^{c(k'-1)-(k'-1)} \\ \leq\ & (c \cdot e \cdot p^{c-1}). \end{aligned}$$

Now we choose $c = (\frac{\log N}{\log(\frac{1}{p})})(\alpha+2)+1$. Then, for sufficiently large N, $\Pr(C \neq \emptyset \text{ after } c(k-1) \text{ steps}) \leq n^{-(\alpha+1)}$. (For the calculation consider the cases $p \leq \frac{1}{2}$ and $p > \frac{1}{2}$, and assume that $N \gg \log(\frac{1}{p})$.) As we only have at most n committees, the union bound yields the desired result. □

Now we can repeat k-Committee Election and k-Verification for $k := 2, 4, 8, \ldots$ until $k \geq n$. The argumentation from Section 2 ensures that now all nodes v have the correct count n and finish the last verification with $x_v = 1$. Thus the algorithm stops. Combining the runtime bounds from Theorem 3 and the above lemma yields our result about counting with an upper bound.

THEOREM 4. *If an upper bound N on the number n of nodes is known to all nodes, then strong counting can be done. If $p > \frac{1}{T}$, then it needs runtime $\mathcal{O}(\frac{n^2}{T} \cdot (\frac{\log T}{\log(\frac{1}{p})})^2 \frac{1}{1-p} + \log(\frac{1}{p}) \cdot n \cdot \log(N))$. If $p \leq \frac{1}{T}$, then runtime $\mathcal{O}(\frac{n^2}{T} + \log(\frac{1}{p}) \cdot n \cdot \log(N))$ suffices. The bounds hold with probability at least $1 - n^{-\alpha}$.*

5.1 p Unknown

If the nodes do not know the error probability p, strong counting is not possible, even if an upper bound N on n is known. But we have the following positive result.

THEOREM 5. *With an additional $\log(n)$ factor, weak counting can still be performed with a modified version of our algorithm above if p is unknown.*

6. REFERENCES

[1] Y. Afek, B. Awerbuch, and E. Gafni. *Applying static network protocols to dynamic networks*, pages 358–370. IEEE, 1987.

[2] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics, 2nd edition.* Wiley, 2004.

[3] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. Saks. *Adapting to Asynchronous Dynamic Networks*, pages 557–570. 1992.

[4] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Deterministic Computations in Time-Varying Graphs: Broadcasting under Unstructured Mobility. In *IFIP TCS*, pages 111–124, 2010.

[5] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-Varying Graphs and Dynamic Networks. *Networks*, page 20, 2010.

[6] A. E. F. Clementi, F. Pasquale, A. Monti, and R. Silvestri. Communication in dynamic radio networks. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, PODC '07, pages 205–214, New York, NY, USA, 2007. ACM.

[7] S. Dolev. Self-Stabilization. *Techniques*, (May), 2003.

[8] P. Flocchini, M. Kellett, P. C. Mason, and N. Santoro. Mapping an unfriendly subway system. In *Proceedings of the 5th international conference on Fun with algorithms*, FUN'10, pages 190–201, Berlin, Heidelberg, 2010. Springer-Verlag.

[9] P. Flocchini, B. Mans, and N. Santoro. Exploration of Periodically Varying Graphs. *CoRR*, abs/0909.4369, 2009.

[10] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988.

[11] R. Ingram, P. Shields, J. E. Walter, and J. L. Welch. *An asynchronous leader election algorithm for dynamic networks*, pages 1–12. IEEE Computer Society, 2009.

[12] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 0:565–574, 2000.

[13] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. *44th Annual IEEE Symposium on Foundations of Computer Science 2003 Proceedings*, pages 482–491, 2003.

[14] F. Kuhn, T. Locher, and R. Oshman. *Gradient clock synchronization in dynamic networks*, pages 270–279. ACM, 2009.

[15] F. Kuhn, T. Locher, and R. Wattenhofer. Tight bounds for distributed selection. *SPAA 07*, pages 145–153, 2007.

[16] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. *Proceedings of the 42nd ACM symposium on Theory of computing STOC 10*, page 513, 2010.

[17] F. Kuhn, R. Oshman, and Y. Moses. Coordinated consensus in dynamic networks. In *PODC '11*, pages 1–10, New York, NY, USA, 2011. ACM.

[18] N. A. Lynch. *Distributed Algorithms*, volume 21. Morgan Kaufmann, 1996.

[19] A. Negro, N. Santoro, and J. Urrutia. Efficient distributed selection with bounded messages. *Parallel and Distributed Systems IEEE Transactions on*, 8(4):397–401, 1997.

[20] R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. *Workshop on Discrete Algothrithms and Methods for MOBILE Computing and Communications*, page 104, 2005.

[21] B. Patt-Shamir. A note on efficient aggregate queries in sensor networks. *Theoretical Computer Science*, 370(1-3):254–264, 2007.

[22] G. Tel. *Introduction to Distributed Algorithms.* Cambridge University Press, 2000.

APPENDIX

Counting Under T-Interval Dynamics and Edge Faults with p Unknown

For completeness' sake we extend Subsection 5.1. Let us first elaborate on the impossibility of strong counting.

THEOREM 6. *If p is unknown, strong counting is not possible, even if an upper bound N on n is known (and the runtime bound depends on n and N).*

PROOF. (sketch) Assume time bound $t(n, N)$ and choose $n = 2$. If $1-p \ll \frac{1}{t(2,N)}$, the edge connecting the two nodes is faulty during the whole computation, with high probability and thus the nodes would output the incorrect value. □

We now state a slightly more detailed version of Theorem 5.

THEOREM 7. *Weak counting with p unknown is possible. If $p > \frac{1}{T}$, then all node output the correct count n after $\mathcal{O}\left(\frac{n^2}{T} \cdot \left(\frac{\log T}{\log(\frac{1}{p})}\right)^2 \cdot \frac{1}{1-p} \cdot \log(n)\right)$ steps. If $p \leq \frac{1}{T}$, they do so after $\mathcal{O}\left(\frac{n^2}{T} \cdot \log(n)\right)$ steps. The bounds hold with probability at least $1 - e^{-\frac{n}{2T}}$.*

PROOF. We will now sketch an algorithm that allows for weak counting under T-interval dynamics and random edge faults, even if p is unknown to the nodes. Note that our modified disseminate procedure from Section 4 uses the knowledge of p for determining the value x that tells how often an attempt to broadcast an item is repeated in the inner loop or, equivalently, how $s \approx \frac{T}{x}$ is chosen. Our approach is to test, for every estimate $k = 1, 2, 4, \ldots$, some values k' also as powers of 2. This means we have $\log(k)$ values for k'. Afterwards, choose p' such that $k = \frac{k'^2}{s^2} \cdot \frac{1}{1-p'} \cdot T$ holds. These pairs are chosen sequentially. We execute dissemination with $k' = 2, 4, 8, \ldots$ and the corresponding value for p'. As soon as $k' > n$ and $p' > p$, our algorithm will output the correct count, w.h.p. This yields: If p is not known to the nodes, it is possible by enumerating over different pairs, n', p' such that $k = \frac{n'^2}{s^2} \cdot \frac{1}{1-p} \cdot T$ and using powers of 2 for k, to perform weak counting in weak counting can be performed w.h.p. in time $\mathcal{O}((\frac{n^2}{T} \cdot (\frac{\log T}{\log(\frac{1}{p})})^2 \frac{1}{1-p} \cdot T) \cdot \log n)$ if $p > \frac{1}{T}$, and in time $\mathcal{O}\left(\left(\frac{n^2}{T} \cdot \frac{1}{1-p}\right) \cdot \log n\right)$ if $p \leq \frac{1}{T}$. □