Fast Convergence in Self-Stabilizing Wireless Networks*

N. Mitton[†‡] E. Fleury[†] I. Guérin Lassous[†] B. Sericola[⋆] S. Tixeuil[◇]

[†] INRIA ARES / INSA, Lyon, France [⋆] INRIA ARMOR / IRISA, Rennes, France

[◇] LRI - CNRS UMR 8623 / INRIA Grand Large, Orsay, France

Abstract

The advent of large scale multi-hop wireless networks highlights problems of fault tolerance and scale in distributed system, motivating designs that autonomously recover from transient faults and spontaneous reconfiguration. Self-stabilization provides an elegant solution for recovering from such faults. We present complexity analysis for a family of self-stabilizing vertex coloring algorithms in the context of multi-hop wireless networks. Such "coloring" processes are used in several protocols for solving many different issues (clustering, synchronizing...). Overall, our results show that the actual stabilization time is much smaller than the upper bound provided by previous studies. Similarly, the height of the induced DAG is much lower than the linear dependency on the size of the color domain (that was previously announced). Finally, it appears that symmetry breaking tricks traditionally used to expedite stabilization are in fact harmful when used in networks that are not tightly synchronized.

Keywords: coloring, scheduling, stabilization time, multi-hop wireless networks

*This work is supported by the FNS of the French Ministry of Research through the FRAGILE project [5] of the ACI Sécurité et Informatique.

[†‡]Corresponding author: Nathalie.Mitton@insa-lyon.fr - phone:(33) 472-436-327

1 Introduction

Wireless multi-hop networks (such as *ad hoc* or sensor networks) consist of sets of independent mobile wireless nodes that operate without the support of a pre-existing fixed infrastructure. They offer unique benefits for certain environments and applications as they can be quickly deployed. Nodes are self-contained, battery-powered computers with radio links that enable the entities to self-organize into a network, communicate with each other and exchange data. The advent of large-scale multi-hop wireless networks highlights problems of fault tolerance and scale in distributed system, motivating designs that autonomously recover from transient faults and spontaneous reconfiguration. Resuming correct behavior after a fault occurs can be very costly [9]: the whole network may have to be shut down and globally reset in a good initial state. While this approach is feasible for small networks, it is far from practical in large networks such as forecast sensor networks. Self-stabilization [2, 3] is an attractive approach for such problems. It provides a way to recover from faults without the cost and inconvenience of a generalized human intervention: after a fault is diagnosed, one simply has to remove, repair, or reinitialize the faulty components, and the system, by itself, will return to a good global state within a relatively short amount of time.

Distributed self-stabilizing algorithms may thus be used for organizing and/or managing multi-hop wireless networks. For example, the self-stabilizing distributed vertex coloring algorithm can be used for resource allocation [7] or distributed local organization [6, 8] in such networks. The vertex coloring problem, issued from classical graph theory, consists in choosing different colors for any two neighboring nodes in a graph. This problem can easily be generalized to distance k, requiring that any two nodes that are up to k hops away

must have different colors. Intuitively, the vertex coloring algorithm runs as follows: every node repetitively collects colors chosen by its neighbors, and if it detects a conflict with its own color, randomly chooses a fresh color (not taken by its distance-k neighborhood). Such algorithms in multi-hop wireless networks do not try to minimize the number of used colors, but when the graph degree is bounded by a small constant (as it is the case in sensor networks), the expected local stabilization time (*i.e.* the stabilization time in any neighborhood) of the algorithm is also constant. This makes the algorithm independent on n, the number of the nodes in the network, and thus scalable to large networks. Moreover, the directed acyclic graph that is induced by the colors is of constant height, so that self-stabilizing algorithms that are composed with the coloring can stabilize also in constant time. This property was used, in the context of multi-hop wireless networks, in [6] to compute a minimal distance-2 coloring, and in [8] to self-organize the network into clusters, both locally stabilizing in expected constant time. While this distance-k coloring algorithm has provenly fast local stabilization time, the influence of system parameters remains unknown.

In this paper, we study the stabilization time of distance-k coloring algorithms in various settings that are relevant to wireless multi-hop networks. We first provide a theoretical study in synchronous and anonymous networks: we extend the analysis carried out in [1] to the context of anonymous networks. Using simulations, we consider various topologies (grids and random graphs), different kinds of scheduling hypothesis (synchronous and probabilistically asynchronous), and variants of the algorithm that uses network wide identifiers so that priorities can be derived for neighboring nodes. We study the impact of these parameters on the stabilization time of distance-k coloring algorithms.

The remaining of the paper is organized as follows: Section 2 formally presents the system model and the coloring algorithm. In Section 3, we analytically study the stabilizing time considering a synchronous setting, while in Section 4 we provide extensive simulations and comments on various parameters. Section 5 gives concluding remarks.

2 Preliminaries

System model. The system is composed of a set V of nodes in a multi-hop wireless network and each node has a unique identifier. Communication between nodes uses a low-power radio. Each node $p \in V$ can communicate with a subset $N_p \subseteq V$ of nodes determined by the range of the radio signal R; N_p is called the neighborhood of node p. p does not belong to N_p ($p \notin N_p$). In the wireless model, transmission is omni-directional: each message sent by p is effectively broadcast to all nodes in N_p. We also assume that communication capability is bidirectional: $q \in N_p$ iff $p \in N_q$. Define $N_p^1 = N_p$ and for $i > 1$, $N_p^i = N_p^{i-1} \cup \{ r \mid (\exists q : q \in N_p^{i-1} : r \in N_q) \}$ (let's call N_p^i the distance-i neighborhood of p). We assume that the distribution of nodes is sparse: there is some known constant δ such that for any node p, $|N_p| \leq \delta$. Note that sensor networks can control density by adjusting their radius R and/or powering off nodes in areas that are too dense, which is one aim of topology control algorithms.

Notation. We describe algorithms using the notation of guarded assignment statements: $G \rightarrow S$ represents a guarded assignment, where G is a predicate of the local variables of a node, and S is an assignment to local variables of the node. If predicate G (called the *guard*) holds, then assignment S is executed, otherwise S is skipped. We assume that all

such guarded assignments execute atomically when a message is received. At any system state, where a given guard G holds, we say that G is *enabled* at that state.

Execution and scheduling. The life of computing at every node consists of the infinite repetition of evaluating its guarded actions. The scheduler is responsible for choosing enabled processors for executing their guarded rules. In this paper, we consider three possible schedulers: the *synchronous* scheduler, the *probabilistic central* scheduler, and the *probabilistic distributed* scheduler. With the synchronous scheduler, nodes operate in lock steps, and at every step, every node is activated by the scheduler. At every step, the probabilistic central scheduler randomly activates exactly one node. With the probabilistic distributed scheduler, at each step, each node is activated with probability $1/n$. The two last schedulers model the fact that although nodes execute their actions at the same speed on average, there is a chance that their clocks or speeds are not uniform, so that the system is slightly asynchronous. The distributed scheduler is more realistic than the central one, but the latter is often used for proving self-stabilizing algorithms.

Shared Variable Propagation. Nodes communicate with their neighbors using *shared* variables. To keep the analysis simple, we assume that there exists an underlying shared variable propagation scheme that allows nodes to collect shared variables in their neighborhood at distance k, for a fixed k. A possible implementation can be found in [6]. For our purpose, we simply assume that a node is able, in one "macro" step, to read all shared variables in its neighborhood at distance k. This assumption is justified in Appendix.

Coloring Algorithm and DAG construction. The coloring algorithm that we consider uses a single shared variable for each node. Let $\mathcal{C}_p$ be a shared variable that belongs to the

domain Δ; variable $\mathcal{C}_p$ is the *color* of node p. The $\mathcal{S}_{\mathcal{C}_p}$ predicate refers to the set of colors that have been used in the neighborhood at distance k of p: $\mathcal{S}_{\mathcal{C}_p} = \{\mathcal{C}_q \mid q \in N_p^k\}$. Let $\mathsf{random}(S)$ choose with uniform probability some element of set S. Node p uses the following function to compute $\mathcal{C}_p$:

$$\mathsf{newC}(\mathcal{C}_p) = \begin{cases} \mathcal{C}_p & \text{if } \mathcal{C}_p \notin \mathcal{S}_{\mathcal{C}_p} \\ \mathsf{random}(\Delta \setminus \mathcal{S}_{\mathcal{C}_p}) & \text{otherwise} \end{cases}$$

The algorithm for vertex coloring is the following: $\mathsf{N1}$:*true* $\rightarrow$ $\mathcal{C}_p := \mathsf{newC}(\mathcal{C}_p)$

As colors are ordered and unique in a node neighborhood, such a coloring algorithm can lead to the construction of a Directed Acyclic Graph (DAG) when (virtually) drawing a directed edge from each node to its only neighbor with the lowest color (if such a node exists).

Local Stabilization. With respect to any given node v, a solution for the coloring problem at distance k is *locally stabilizing* for v with convergence time t if, for any initial system state, after at most t time units, every subsequent system state satisfies the property that any node w at distance less than k from v is such that $C_w \neq C_v$. For randomized algorithms, this definition is modified to specify expected convergence times (all stabilizing randomized algorithms we consider are probabilistically convergent in the Las Vegas sense). In [6], the authors show that Algorithm $\mathsf{N1}$ self-stabilizes with probability 1 and has constant expected local stabilization time.

Uniform vs. Non Uniform Networks. In theory, the coloring algorithm $\mathsf{N1}$ could work in uniform and anonymous networks (where nodes do not have unique identifiers and execute the same code). Collecting the neighborhood at distance k generally requires identifiers. Then, it is possible to tweak the algorithm to use these identifiers to break symmetry and expedite convergence: when at least two neighbors have a conflicting color, the node with

the lowest identifier never changes its color. Thus, we have the guarantee that at any step, at least one of the conflicting nodes gets a stable color. In the remaining of the paper, we distinguish two operating modes for the algorithm: the **all** mode refers to the mode where all conflicting nodes draw a new color (anonymous networks), while the **all but one** mode refers to the version where the previous algorithm applies (uniform networks where one conflicting node does not change its color).

3 Analysis

In this section, we theoretically compute the expected stabilization time of the coloring protocol N1, *i.e.* the expected number of steps before a node has a color that is not already used in its distance-k neighborhood. From [6], we already know that when the degree is upper bounded by a constant(δ), the expected local stabilization time is also upper bounded by a constant. However, the actual constant is not given in [6], and the one that can be derived from the algorithm is high (about δ^6 for a distance-3 coloring).

The other metric of interest for our purpose is the height of the DAG that is induced by the colors. Indeed, when the coloring algorithm is used as a building block for subsequent algorithms, the stabilization time of those algorithms is generally in the order of the color DAG height, a lower DAG height inducing a smaller stabilization time. In [6], the authors show that the height of the DAG is bounded by $|\Delta| + 1$, where Δ denotes the color domain.

For the theoretical study, we consider only the synchronous scheduler, and we model the coloring protocol by a successive set of random draws. The main goal is to assign a color to each node. A way to model this problem is to consider that the color domain is represented

by a set of M urns in which one must randomly distribute L balls which are going to represent the nodes. In each neighborhood, the goal is then to have only one ball (one node) associated to a given urn (one color).

This model has already been used in [1] (for the NAP protocol), in which the authors analyze the stabilization time of a self-addressing network where two links must receive a unique prefix in the network. In this model, the urns symbolize the prefix and the balls the links. Each link chooses a random prefix in a prefix domain. If two links have chosen the same prefix, the one with the lowest ID keeps it while the other one(s) choose(s) a new prefix among the ones not already assigned. The analysis and the calculus carried out in [1] thus roughly correspond to our **all but one** mode. In this section, we extend the analysis in the **all** mode.

Note that this theoretical study only matches for complete networks, *i.e.* where each node can communicate with all the other nodes. In multi-hop networks, it is possible that two neighboring nodes (A and B) with no conflicting colors simultaneously draw a new identical color, because they each have another conflicting neighbor (not visible to A or to B). But, it is important to note that this theoretical study gives a lower bound on the actual stabilization time (this is further refined in Section 4).

The algorithm can be modeled in terms of urns/balls as follows.

Algorithm 1 COLORING_PROCESS(L, M)

▷ *Input: M urns and L balls*
▷ *Pre-condition: $M \geq L$*
if ($L \neq 0$) **then**
    Randomly throw the L balls in the M urns;
    if (case = 'all') **then**
        Keep aside all urns that contain exactly one ball with their ball inside ;

end
if (case = 'all but one') **then**
Keep aside all urns containing at least one ball and one of their balls;
end
Let note $c \leq M$ the number of "correct" urns that we keep aside;
Call COLOR_PROCESS($L - c$, $M - c$);
end

By repeating the process, eventually every ball will be stored in a correct urn and every urn will contain at most one ball. Let N denote the number of iterations needed to reach such a configuration, *i.e.* the number of calls to the recursive procedure of coloring. We are interested in computing the distribution and the expectation of the random variable N. We consider the homogeneous discrete-time Markov chain $X = \{X_n,\, n \in \mathbb{N}\}$, on the finite state space $\mathcal{I} = \{0, 1, \ldots, L\}$ where the event $\{X_n = i\}$ represents the fact that, after n transitions (or calls to the coloring procedure), the procedure COLOR_PROCESS($M-i$, $L-i$) is currently executed. In other words, $\{X_n = i\}$ represents the fact that, after n transitions, exactly i urns and balls have been kept aside, *i.e.* exactly i urns contain exactly one ball. The Markov chain starts in state 0 with probability 1, which means that at the beginning, none of the urns and balls have been kept aside. The random variable N can be defined more formally, for $L \geq 1$, as $N = \sum_{n\geq 0} \sum_{i=0}^{L-1} 1_{\{X_n=i\}}$. N is the number of transient states of X visited before absorption.

We denote by $\mathcal{P}(L, M) = (p_{i,j}(L, M))_{(i,j)\in\mathcal{I}^2}$ the transition probability matrix of the Markov chain X, where $p_{i,j}(L, M)$ represents the probability to have exactly j urns and balls kept aside at the time $n + 1$ given that i urns and balls have been kept aside at time n. $p_{i,j}(L, M)$ is thus the probability to obtain exactly $j - i$ urns each with one ball when

throwing $L-i$ balls into $M-i$ urns. For all $i \leq j$, we thus have

$$p_{i,j}(L,M) = p_{i,j} = p_{0,j-i}(L-i, M-i). \tag{1}$$

The Markov chain X is clearly acyclic and the state L is the absorbing state of X. This means that for every $i \in \mathcal{I} - \{L\}$ and $j \in \mathcal{I}$, we have $p_{i,j}(L,M) = 0$ for $i > j$ and $p_{L,L}(L,M) = 1$.

The transition probability matrix $\mathcal{P}(L,M)$ has thus the following form:

$$\mathcal{P}(L,M) = \begin{pmatrix} p_{0,0} & p_{0,1} & p_{0,2} & \cdots & \cdots & p_{0,L} \\ 0 & p_{1,1} & p_{1,2} & p_{1,3} & \cdots & p_{1,L} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & p_{L-1,L-1} & p_{L-1,L} \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}.$$

The computation of the matrix $\mathcal{P}(L,M)$ depends on the drawing hypothesis (**all but one** or **all** mode). But, once the transition probability matrix is computed, the way of evaluating the stabilization time of the coloring algorithm is the same whatever the drawing hypothesis. The **all but one** mode is a little bit simpler because it implies $p_{i,i}(L,M) = 0$ for every $0 \leq i \leq L-1$, which means that the stabilization time N of the coloring algorithm is bounded: $N \leq L$. We give in the following subsection the transition probability matrix of each mode before giving the distribution and the expectation of the stabilization time N of the coloring algorithm.

Computation of the transition matrix $\mathcal{P}(L, M)$. We now explicit the transition probability $p_{i,j}(L, M)$ of the Markov chain X for all $i \leq j$ for both modes.

Matrix for the all mode: In the **all** mode, all nodes in conflict choose a new color among the free ones. At each step, we keep aside the urns and their balls if they contain exactly one ball. Unlike the **all but one** mode, we may keep none urn and ball at a step and thus the diagonal values of $\mathcal{P}(L, M)$ are not null.

From relation (1), we only have to compute the transition probability $p_{0,j-i}(L-i, M-i)$ for $i \leq j$. The computation of the transition probabilities $p_{i,j}(L, M)$ yields to the computation of the transition probabilities $p_{0,j}(L, M)$.

By definition of the transition matrix as stated above, the transition probability $p_{0,j}(L, M)$ is the probability to obtain exactly j urns containing only one ball when throwing randomly L balls into M urns. The case $j = L$ yields to the generalized birthday problem and thus we have:

$$p_{0,L}(L, M) = \frac{M!}{(M-L)!M^L} \tag{2}$$

For $j < L$, we proceed as follows. We throw L balls into M urns. We denote by $K_0(L, M)$ the number of empty urns and by $K_1(L, M)$ the number of urns with exactly one ball inside. Let $a_{L,M}(k, j)$ denote the joint distribution of both random variables $K_0(L, M)$ and $K_1(L, M)$, that is $a_{L,M}(k, j) = \mathbb{P}[K_0(L, M) = k,\ K_1(L, M) = j]$. The transition probability

$p_{0,j}(L,M)$ can thus be expressed as

$$p_{0,j}(L,M) = \mathbb{P}[K_1(L,M) = j] = \sum_{k=0}^{M} a_{L,M}(k,j).$$

In order to compute all the probabilities $a_{L,M}(k,j)$ we can proceed by recursion on integer L by conditioning on the result of the throw of the last ball. More precisely, in order to obtain k empty urns and j urns with only one ball inside when throwing L balls in M urns we need:

1. Either obtaining $k+1$ empty urns and $j-1$ urns with exactly one ball inside at the end of the $L-1$ first throws of $L-1$ balls in M urns and throwing the last ball in one of the $k+1$ empty urns.

2. Either obtaining k empty urns and $j+1$ urns with exactly one ball inside at the end of the $L-1$ first throws of $L-1$ balls in M urns and throwing the last ball in one of the $j+1$ urns containing exactly one ball.

3. Or obtaining k empty urns and j urns with exactly one ball inside at the end of the $L-1$ first throws of $L-1$ balls in M urns and throwing the last ball in one of the $M-(j+k)$ urns that contain at least 2 balls.

Thus, from that decomposition and for $L \geq 2$, we have:

$$a_{L,M}(k,j) = \frac{k+1}{M} a_{L-1,M}(k+1, j-1) 1_{\{j \geq 1\}} + \frac{j+1}{M} a_{L-1,M}(k, j+1) + \frac{M-(j+k)}{M} a_{L-1,M}(k,j)$$

where $1_{\{c\}}$ is the indicator function equal to 1 if condition c is true and 0 otherwise. For

$L = 1$ we trivially have as initial value for the recursion:

$$a_{1,M}(k,j) \quad = \quad 1_{\{k=M-1,\, j=1\}}. \tag{3}$$

It is also easy to check that $a_{L,M}(k,j) = 0$ if either $j > L$ either $k = M$ or $j + k > M$. Note that for $j = L$, we have $a_{L,M}(k,L) = 0$ for $k \neq M - L$ and $a_{L,M}(M-L,L) = p_{0,L}(L,M)$ which is given by relation (2) and that clearly satisfies the recurrence relations.

Matrix for the all but one mode: This case is the one given in [1]. We only give here the results. In the **all but one** mode, at each step, we keep aside at least one urn together with one of the balls contained inside. This means that the transition probability $p_{i,i}(L,M) = 0$ for every $0 \leq i \leq L-1$. The Markov chain X is thus strictly acyclic and the stabilization time N of the coloring algorithm is bounded by L.

Based on a well-known result about the number of ways of throwing r different balls in n different urns such that exactly m urns are non-empty [4], the transition probability matrix is given according to (1), for $i < j$, by

$$p_{i,j}(L,M) \;=\; \binom{M-i}{j-i} \sum_{k=0}^{j-i} \binom{j-i}{k} (-1)^k \left(\frac{j-i-k}{M-i}\right)^{L-i}.$$

Distribution and expectation of N. Now, given the transition probability matrix of the Markov chain X, we are able to determine the distribution ($\mathbb{P}[N=n]$ for $n = 0, \ldots, \infty$) and the expected value $\mathbb{E}[N]$ of the random variable N for both modes. The mean value $\mathbb{E}[N]$ is also actually the mean number of steps needed before stabilization of our coloring algorithm and so its stabilization time.

These calculus are detailed in [1]. We directly use them here. They are easily derived from the classical results on Markov chains. Let Q denote the sub-matrix obtained from $\mathcal{P}(L, M)$ by removing the last line and the last column which correspond to the absorbing state L. We denote by α the row vector containing the initial probability distribution of the transient states of X, *i.e.* $\alpha = (\mathbb{P}[X_0 = i])_{i=0,\ldots,L-1}$. As already seen, the Markov chain starts in state 0 with probability 1 and, thus we have $\alpha = (1, 0, \ldots, 0)$. With these notations, we have

$$\mathbb{P}[N = n] = \alpha Q^{n-1}(I - Q)\mathbb{1}, \text{ for } n \geq 1,$$

$$\mathbb{P}[N > n] = \sum_{k=n+1}^{\infty} \alpha Q^{k-1}(I - Q)\mathbb{1} = \alpha Q^n \mathbb{1}, \text{ for } n \geq 0,$$

and

$$\mathbb{E}[N] = \sum_{n=0}^{\infty} \mathbb{P}[N > n] = \alpha (I - Q)^{-1} \mathbb{1},$$

where I is the identity matrix and $\mathbb{1}$ is the column vector with all entries equal to 1, both of dimension L. If $V = (V_i)_{0 \leq i \leq L-1}$ is the column vector defined by $V_i = \mathbb{E}[N|X_0 = i]$, we have $\mathbb{E}[N] = V_0$. The vector V of conditional expectations is given by $V = (I - Q)^{-1} \mathbb{1}$, which means that it is solution to the linear system $(I - Q)V = \mathbb{1}$, which can also be written as $V = \mathbb{1} + QV$ or equivalently, since the matrix $\mathcal{P}(L, M)$ is acyclic, as

$$V_i = \frac{1}{1 - p_{i,i}} \left(1 + \sum_{j=i+1}^{L-1} p_{i,j} V_j \right) \text{ for } i = L - 2, \ldots, 0,$$

with $V_{L-1,L-1} = 1/(1 - p_{L-1,L-1})$.

We are now able to compute $\mathbb{E}[N] = V_0$ by recurrence from V_i. We use these theoretical

results in the next sections to compare with the simulation outcomes.

4 Simulations

We performed simulations to evaluate the stabilization time of the coloring algorithms over different assumptions, to analyze the directed acyclic graph deriving from it in each case, and to compare those values to the theoretical ones in order to validate our analytical model. As mentioned in the previous sections, we are mainly interested in the stabilization time under several scheduling and coloring hypothesis but also on the height of the derived DAG.

Simulation model. We use a simulator we developed and that assumes an ideal MAC layer. In each case, each statistic is the average over 1000 simulations. All coloring algorithms studied below are analyzed over a same node distribution.

We say that a node executes an action (or acts, for short) when it checks its neighbors' color and if needed, chooses another color among the available ones. For a distance-k coloring, a color $c \in \Delta$ is said available for a node u if $\forall v \in N_u^k, \mathcal{C}_v \neq c$. As mentioned in Section 2, we studied the coloring algorithm for different kinds of scheduling hypothesis: Synchronous (every node acts at every step), Probabilistic Central (exactly one random node acts at every step) and Probabilistic Distributed (at each step, each node acts with probability $\frac{1}{n}$). Each scheduling hypothesis is studied for both modes of coloring: **all but one** mode (for each pair of conflicting nodes, every node but one chooses another color) and **all** mode (every conflicting node chooses another color).

We have run Algorithm N1 for distance-1 and distance-2 colorings with these scheduling

hypothesis and modes over a random geometric topology and a grid with a varying number of neighbors per node and different sizes of color domain Δ . The initial stabilizing algorithm has been designed using $|\Delta| = (max_{p \in V}|N_p|)^{2 \times k}$, k being the coloring distance, but as we are interested in the impact of this domain size over the stabilization time and the DAG height, we have also run Algorithm N1 for $|\Delta| = 2 \times \left(max_{p \in V}|N_p^k|\right)$. Moreover, as in *sensor* networks, nodes do not have a general view of the network and thus have no way *a priori* to know the maximum degree in the graph, we have also run simulations where each node has its own color domain size such that $\forall p \in V$, $|\Delta_p| = (|N_p|)^{2 \times k}$. In these settings, we collect the stabilization time of the coloring algorithm as well as the height of the induced DAG. In the synchronous mode, every node acts with probability 1 (so the expected time before acting is 1 step) whereas in both probabilistic modes, every node acts within expected n steps. As schedulers are meant to model scheduling properties rather than implemented scheduling mechanisms, in a "real" system a particular node would act within a constant expected time. So, in order to be able to compare the three schedulers and to be fair when considering the stabilization time of different schedulings, the number of steps before stabilization in probabilistic modes is divided by n.

In the geometric approach, nodes are randomly deployed using a Poisson Process in a 1×1 square with various levels of intensity λ (from 500 to 1000). In such processes, λ represents the mean number of nodes per surface unit. The communication range R is set to 0.1 in all tests. We thus have a mean number of neighbors per node ranging from 15 to 32. In the grid, we consider topologies where each central node has 4 and 8 neighbors.

Theory vs. Simulation. As mentioned in Section 3, we expect that our theoretical results give an accurate lower bound on the stabilization time of the coloring algorithm. Indeed, when considering a complete graph, once a node u has chosen an available color, every other node in the network knows that color (as a neighboring node of u) and thus will not choose it anymore. However, in a non-complete graph, a situation such as illustrated in Figure 1 (for distance-1 coloring) may occur and thus, the average stabilization time would be higher than the one in a complete graph, therefore our theoretical analysis only leads to a lower bound. For example, let's run the algorithm over the graph plotted on Figure 1, under the synchronous scheduling and the **all** mode. Every node has to choose a color from 0 to 4 until getting a locally unique color. Node B has two neighbors, A and C. The example shows the colors drawn at each step by each node. As there still exist conflicts at the end of the three steps, the algorithm would not stop yet. But, from the theoretical analysis, if we consider that A, B and C are in a complete graph, then at step 2, C has its color (3) and at step 3, A and B have their color (1 and 2). So at the end of the three steps, with the theoretical analysis, every urn has been kept aside and the system should have stabilized.

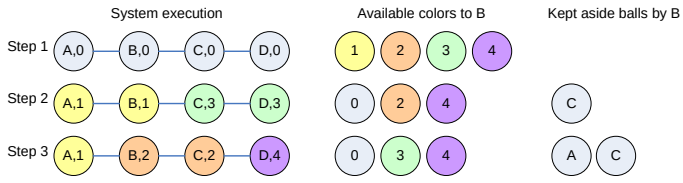
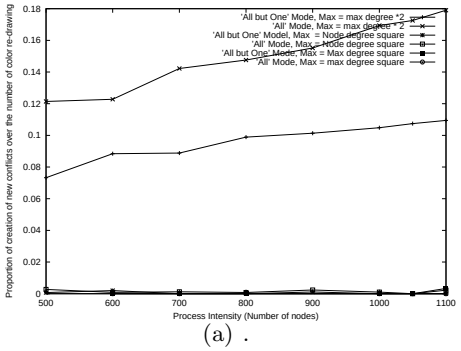


Figure 1: A possible execution for distance-1 coloring in a synchronous network.

In order to evaluate the difference between the analytical lower bound and the simulated stabilization time, we compute by simulation the number of times that two neighboring nodes not originally in mutual conflict have to both choose another color and both get the same

color. Results for a distance-1 synchronous scheduling coloring are given in Figure 2. As we can see, the greater the color domain size, the more unlikely this case is to appear (almost 0% of the cases when the color domain size is quadratic in the maximum degree of the nodes). We also note that, even for a small color domain size (such as twice the maximum degree), this case does not occur very often (less than 18% in the worst case). Thus, we can expect that in practice, our analytical result gives a very accurate lower bound on the stabilization time of the coloring protocol.



(a) .

+	**All but one** mode - Max = max degree ×2
×	**All** mode - Max = max degree ×2
*	**All but one** mode - Max = node degree square
□	**All** mode - Max = node degree square
■	**All but one** mode- Max = max degree square
⊖	**All one** mode - Max = max degree square

(b) Legend

Figure 2: Proportion of nodes creating a new conflict over the number of nodes choosing another color.

Tables 1 and 2 compare analytical and simulation results for the Synchronous scheduling in both modes, when using two different color domain sizes $|\Delta|$, for grids and random topologies. As expected, theoretical results give tight lower bounds of the simulation outcome. Note that the size of the grid has no influence on the results, the number of neighbors being the only key parameter.

Stabilization time. Figure 3 presents the stabilization time for each scheduling hypothesis and both modes when using different sizes of color domain, for a random spatial distribution of the nodes. The most striking result compared to those of [6] is that the stabilization

4 neighbors							
all but one				**all**			
$2 * Max$		Max^2		$2 * Max$		Max^2	
Theory	Simulation	Theory	Simulation	Theory	Simulation	Theory	Simulation
1.88	1.89	1.51	1.64	2.14	2.14	1.56	1.61
8 neighbors							
all but one				**all**			
$2 * Max$		Max^2		$2 * Max$		Max^2	
Theory	Simulation	Theory	Simulation	Theory	Simulation	Theory	Simulation
1.51	1.56	1.14	1.22	1.56	1.67	1.15	1.21

Table 1: Theory and simulations results for the stabilization time for synchronous distance-1 coloring with $|\Delta| = (max_{p \in V}|N_p|)^2)$ and $|\Delta| = 2 \times (max_{p \in V}|N_p|)$ in grids.

	500 nodes	600 nodes	700 nodes	800 nodes	900 nodes	1000 nodes
Mean degree	15.7	18.8	22.0	25.1	28.3	31.4
all but one						
Theory	2.35	2.40	2.44	2.50	2.53	2.56
Simulation	2.78	2.91	2.87	2.94	2.99	2.97
all						
Theory	2.83	2.91	2.94	2.95	3.05	3.08
Simulation	3.25	3.31	3.29	3.28	3.42	3.41

Table 2: Theory and simulations results for the stabilization time for synchronous distance-1 coloring with $|\Delta| = 2 \times (max_{p \in V}|N_p|)$ in random geometry topologies.

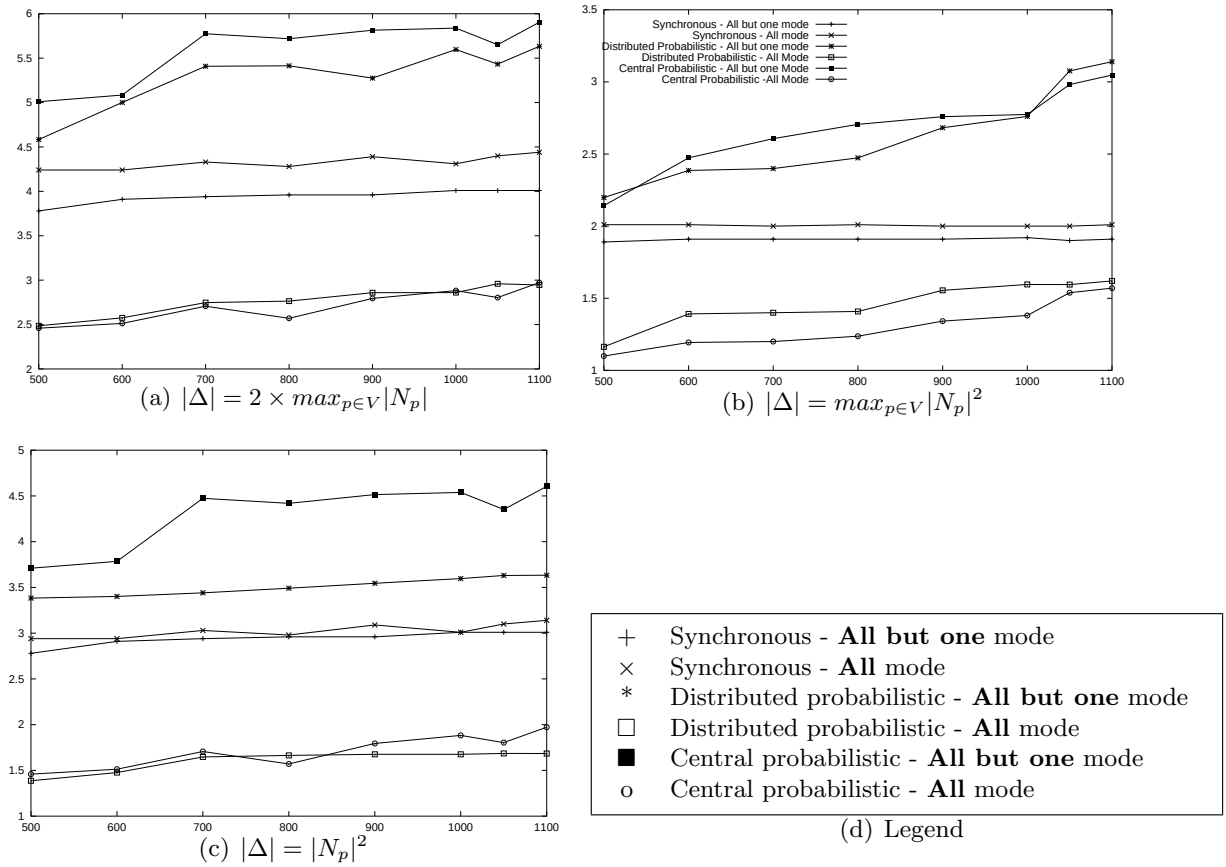
time is much lower than expected. From the results of [6], the stabilization time is at least linear in $|\Delta|$ (being upper bounded by a constant when δ also is a constant). In contrast, our simulation results show a sub-linear (in $|\Delta|$) stabilization time, since considering $|\Delta| = 2 \times max_{p \in V}|N_p|$ *vs.* $|\Delta| = max_{p \in V}|N_p|^2$ merely divides by two the stabilization time. Also, doubling the degree of the nodes (when the network grows from 500 nodes to 1100 nodes) does not double the stabilization time. Especially, with the synchronous and distributed probabilistic schedulers, the stabilization time remains upper bounded by a constant.

In all cases, whatever the coloring distance k and the color domain Δ, we can note that the behavior of both probabilistic schedulers is similar. With the probabilistic scheduling hypothesis, in order to stabilize, the scheduler has to choose a node in conflict. In the **all but one** mode, only one node per pair of conflicting nodes actually chooses another color.

The probabilistic schedulers thus have less chance to elect a conflicting node in the **all but one** mode than in the **all** mode (almost half chances less). Therefore, with the **all** mode, these probabilistic schedulings achieve better stabilization time (almost half time) than with the **all but one** mode. In the synchronous mode, at each step, every node acts. As in the **all but one** mode, in any pair of conflicting nodes, already one of those nodes has a stable color, so the stabilization time is lower than in the **all** mode. Note that for *sensor* networks, nodes are rarely tightly synchronized, so that the most realistic model is the distributed probabilistic scheduler, so the **all** mode is to be preferred in this context. This mode also is the easiest to implement in sensor networks as it does not require that every node knows the identity of the nodes in its k-neighborhood.

Influence of the size of the color domain. Since the most realistic model for *sensor* networks is the distributed probabilistic one with **all** mode, Figures 4(a) and 4(b) plot the stabilization time with these hypothesis for the distance-1 coloring algorithm as well as the height of the induced DAG, using different sizes of color domain. Nevertheless, results for the DAG height are similar whatever the coloring hypothesis.

Results clearly show that a higher domain size $|\Delta|$ induces a lower stabilization time and a higher DAG. There thus is a trade-off to do between these two characteristics depending of the application that will use the coloring. However, and although theoretical results show that the DAG height can be up to $|\Delta| - 1$, simulation results show that the actual height is in fact much lower, and most certainly sub-linear in $|\Delta|$.



(a) $|\Delta| = 2 \times max_{p \in V}|N_p|$

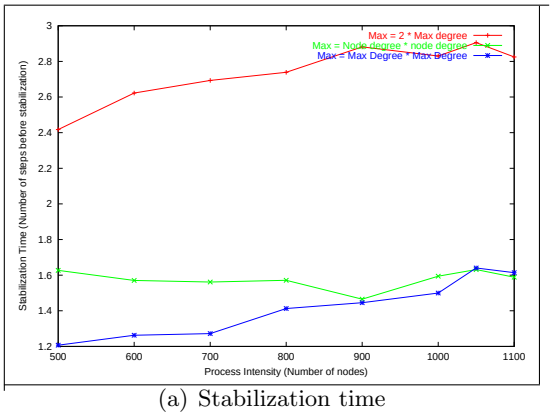
(b) $|\Delta| = max_{p \in V}|N_p|^2$

(c) $|\Delta| = |N_p|^2$

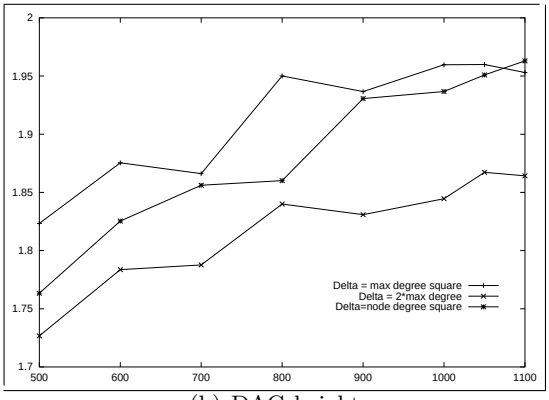
(d) Legend

Figure 3: Stabilization time of the distance-1 coloring process for different modes and scheduling hypothesis over a geometric node distribution with different sizes of color domain.

Influence of the coloring distance. Figure 5 plots the stabilization time for distance k-coloring with **all** mode, assuming the distributed probabilistic scheduler in 4-neighbor and 8-neighbor grids, respectively. The used color domain size is $(max_{p \in V}|N_p|)^{2 \times k}$. In order to evaluate the time to collect all colors from the neighborhood at distance k in a wireless sensor network, we reason as follows. Consider the subgraph that is centered in node p and contains all nodes up to k hops away from p. Now assume that the expected time for nodes in a 1-hop neighborhood to communicate with all their neighbors is upper bounded by a constant C (See Appendix). In the worst case, this subgraph is a tree where each node has degree



(a) Stabilization time



(b) DAG height

Figure 4: Influence of the color domain size on the stabilization time and the DAG height.

δ except for the leaves that have degree 1, overall δ^{k-1} nodes. After one time unit, $1/C$ of the nodes have transmitted correctly. After one extra time unit, $1/C$ of the *remaining* nodes have transmitted correctly. So, after $\log_C(\delta^{k-1})$ time units, all nodes up to k hops from p have transmitted correctly. So, those nodes have correct distance-one information. Repeating this pattern k times, after $k \times \log_C(\delta^{k-1}) = O(k^2)$, nodes are aware of the colors of their k-neighborhood. In Figure 5, values have thus been multiplied by k^2.

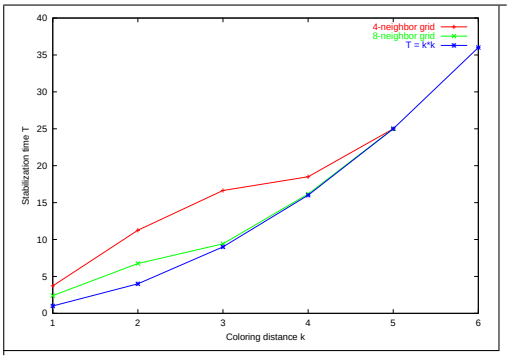


Figure 5: Stabilization time with respect of time for message transmission as a function of the coloring distance.

It turns out that when k increases, the stabilization time eventually increases as fast as k^2, independently of the topology. This means that the number of colors used is sufficiently

large so that the coloring itself is almost performed in constant time, but the communication time becomes the main bottleneck of the algorithm.

5 Concluding remarks

Distance-k coloring is a useful mechanism for multi-hop wireless networks. In [6], distance-3 coloring was used to construct a TDMA schedule and in [8], distance-2 coloring permitted to expedite density-based cluster construction. Further applications could be derived, *e.g.* distance-k maximal independent set construction, by having nodes that have locally minimal color in their k-neighborhood be part of the independent set, and remaining nodes that do not see distance-k neighbors with lower colors in the independent set join the independent set.

In this paper, we have first extended the theoretical analysis of [1] to anonymous networks (**all but one** mode). Then, by simulations, we have evaluated the impact of two considered modes (**all but one** and **all** modes), of different scheduling policies and of the color domain on the stabilization time and the induced DAG height. We have shown analytically and by simulation that the stabilization time and the DAG height of such coloring protocols in multi-hop wireless networks are low.

Further studies are needed to get a more realistic bound that takes into account the probabilistic nature of the MAC layer used in such networks. Indeed, as k grows, the size of the messages gets longer (in the order of δ^{k-1}, since a node needs to communicate information about its neighborhood at distance $k-1$ to each of its neighbors), so the probability that collisions occur between neighbors grows, and the delay increases.

References

[1] G. Chelius, E. Fleury, B. Sericola, and L. Toutain. On the NAP Protocol. Technical Report 5701, INRIA, 2005.

[2] E. Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the ACM*, 17:643–644, 1974.

[3] S. Dolev. *Self-Stabilization*. The MIT Press, 2000.

[4] M. Eisen. *Introduction to Mathematical Probability Theory*. Prentice Hall, E. Cliffs, 1969.

[5] FRAGILE. Failure Resilience and Application Guaranteed Integrity in Large-scale Environments. `http://www.lri.fr/~fragile/`.

[6] T. Herman and S. Tixeuil. A Distributed TDMA Slot Assignment Algorithm for Wireless Sensor Networks. In *AlgoSensors'2004*, number 3121 in LNCS, pages 45–58, Turku, Finland, July 2004.

[7] N Lynch. Distributed algorithms. *Morgan Kaufmann*, 1996.

[8] N. Mitton, E. Fleury, I. Guérin-Lassous, and S. Tixeuil. Self-stabilization in self-organized wireless multihop networks. In *Proceedings of WWAN'05*. IEEE Press, 2005.

[9] Raida Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*. Addison-Wesley Longman, 2000.

[10] H. Zhai, Y. Kwon, and Y. Fang. Performance analysis of IEEE 802.11 MAC Protocols in wireless LANs. *Wireless communications and mobile computing*, 4:917–931, 2004.

A Appendix

In this section, we justify the following assumption "a node is able, in one step, to read all shared variables of its neighbors". In more details, we show that the expected time before receiving information from every neighbor is upper bounded by a constant.

In [10], the authors provide a performance analysis of IEEE 802.11 MAC protocols in wireless LANs. By considering a graph with n stations that are all within the transmission range of one another (*i.e.* the communication graph is a complete graph), the authors model the backoff timer nodes trigger in 802.11 before transmitting, which depends on the collisions that have occurred before. They deduce in particular the probability P_{suc} that there is one successful transmission among the n stations in a considered time slot. A transmission is considered as successful if there is exactly one station that emits in this slot. If p_c is the probability that there is at least one packet transmission in the medium among n stations (p_c is also given in [10]), we have:

$$P_{suc} = (n-1)((1-p_c)^{(n-2)/(n-1)} + p_c - 1)$$

We now show that the time before all neighbors of a node successfully communicate is upper bounded by a constant, on average. Let the random variable X be the number of slots needed before the n stations be able to transmit information to their neighbors. In the best case, each station chooses a time slot to transmit different from every other. We thus have: $\mathbb{P}[X < n] = 0$ and $\mathbb{P}[X = n] = P_{suc}^n$.

Then, $\mathbb{P}[X = k, k > n]$ is the probability that at the end of the $(k-1)$ first time slots,

$(n-1)$ stations have successfully emitted and that the n^{th} station succeeds to transmit during slot k. We thus have: $\mathbb{P}[X = k, k > n] = \binom{k-1}{k-n+1}\binom{n}{n-1}(1 - P_{suc})^{(k-n+1)}P_{suc}^n$

We can deduce the mean number of slots $\mathbb{E}[X]$ needed before every n stations are able to transmit an information to their neighbors.

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{k=0}^{\infty} k\mathbb{P}[X = k] \\
&= n\mathbb{P}[X = n] + \sum_{k=n+1}^{\infty} k\mathbb{P}[X = k] \\
&= P_{suc}^n \times \left(n + \sum_{k=n+1}^{\infty} k \binom{k-1}{k-n+1}\binom{n}{n-1}(1 - P_{suc})^{(k-n+1)} \right)
\end{aligned}$$

This can be derived into:

$$\begin{aligned}
\mathbb{E}[X] &= P_{suc}^n \times \left(n + n(n+1)(\frac{1}{P_{suc}^n} - (n+1) + nP_{suc}) \right) \\
&= nP_{suc}^n \times \left(1 + (n+1)(\frac{1}{P_{suc}^n} - (n+1) + nP_{suc}) \right)
\end{aligned}$$

As P_{suc} only depends on n and that we assume that n is upper bounded by a constant, $\mathbb{E}[X]$ also is a constant. Therefore, in average, a node is able to read all shared variables in its neighborhood within constant time.