Self-stabilization in Self-organized Wireless Multihop Networks*

N. Mitton† B. Sericola‡ S. Tixeuil§ E. Fleury¶ I. Guérin Lassous‖

November 24, 2009

Abstract

In large scale multihop wireless networks, flat architectures are typically not scalable. Clustering was introduced to support self-organization and enable hierarchical routing. When dealing with multihop wireless networks, robustness is a crucial issue due to the dynamics of such networks. Several algorithms have been designed for clustering but to date, in none of them the self-stabilization features of the resulting structure have been investigated.

In this paper, we show that a clustering algorithm, known for its good robustness properties, is actually self-stabilizing. We propose several enhancements to the scheme to reduce the stabilization time and thus improve stability in a dynamic environment. The key technique to these enhancements is a localized self-stabilizing algorithm for Directed Acyclic Graph (DAG) construction. We provide extensive studies (both theoretical and experimental) that show that our approach enables efficient yet adaptive clustering in wireless multihop networks.

keywords: multihop wireless networks, clustering, self-stabilization, scalability, coloring, theoretical analysis, Markov chains.

1 Introduction

Ad hoc networks or wireless sensor networks (wireless multihop networks) are composed of devices that communicate via wireless interfaces. They do not require any fixed infrastructure nor human intervention. Both are strongly based on self-organization and self-stabilization. Even if every mobile can move everywhere, and thus can disappear or appear in the network at any time, the network manages the topology changes and provides the connectivity between any pair of terminals. If current wireless network interface cards allow the communication between mobiles that are in communication range, a routing protocol is required to provide

*This work is supported by the ANR projects SHAMAN and ALADDIN. Additional support from FRACAS INRIA ARC.

†INRIA, LIFL (CNRS/USTL), 59650 Villeneuve d'Asq, France - nathalie.mitton@inria.fr

‡INRIA, IRISA, 35042 Rennes Cedex, France - Bruno.Sericola@irisa.fr

§Univ. Pierre & Marie Curie - Paris 6, France - Sebastien.Tixeuil@lip6.fr

¶Université de Lyon, ENS lyon/LIP, INRIA/D-NET - 69364 Lyon Cedex 07, France - Eric.Fleury@inria.fr

‖Université Lyon, 1 LIP (UMR INRIA - Université Lyon I - CNRS - ENS Lyon 5668) - 69364 Lyon Cedex 07, France - Isabelle.Guerin-Lassous@ens-lyon.fr

an end-to-end connectivity within the network between any pair of nodes. As there are no dedicated devices in the network, all mobiles are potential routers. Such networks have become very popular due to their ease of use. Their applications range from the network extension when cabling is not possible or too expensive to spontaneous networks in case of natural disasters where the infrastructure has been totally destroyed by going through the monitoring and the collect of data with wireless sensor networks.

Due to the dynamics of such networks (devices mobility and/or instability of the wireless medium), routing protocols for fixed or/and wired networks are not adapted. Ad hoc routing protocols proposed in the MANET working group at IETF[1] are all flat routing protocols. It means that there is no hierarchy and all terminals have the same role. If flat protocols are quite effective on small and medium networks, they are not suitable on large scale networks due to bandwidth and processing overhead. Hierarchical routing seems to be more adapted to such large networks. It often relies on a specific partition of the network, called *clustering*: the devices are gathered into clusters according to some criteria and specific routing protocols are used within and between the clusters. In addition to its scalability, such an organization presents numerous advantages such as synchronizing mobiles in a cluster or attributing new services zones. Many algorithms have been designed for the clustering step. As mentioned in [45], "one measure of robustness of the topology is given by the maximum number of nodes that need to change their topology information as a result of a movement of a node".

As current clustering protocols usually run continuously, it is expected that the system is dynamic (nodes may leave or join the network), so an algorithm for the distributed construction of a clustering should be able to reconfigure on the fly. *Self-stabilization* [12, 13, 49] is an elegant approach to forward recovery from transient faults as well as initializing a large-scale system. Informally, a self-stabilizing system is able to recover from any transient fault in finite time, without restricting the nature or the span of those faults.

To the best of our knowledge, very few studies on clustering in sensor networks focus on the robustness and self-organization properties of the described protocols. Moreover, when faults are considered, evaluation is carried out by simulations only and never by using a theoretical approach. In this paper, we apply self-stabilization principles to a clustering protocol that was previously proposed in [34]. With a theoretical approach, which can be applied to several other clustering schemes, we show that, under some assumptions, the algorithm is self-stabilizing. We also improve the robustness by adding extra-advanced features and we show that the resulting algorithm is still self-stabilizing. These self-stabilizing properties are further validated practically by simulations. We also add a mechanism to ensure *local* self-stabilization (*i.e.* stabilization in some part of the network has no influence on stabilization in other parts of the network) and a low (expected constant) stabilization time in every case. This mechanism, based on a coloring process, is analyzed theoretically and by simulations. In more details, we study the stabilization time of distance-k coloring algorithms in various settings that are relevant to wireless multihop networks. We first provide a theoretical study in synchronous and anonymous networks. Using simulations, we then consider various topologies (grids and random graphs), different kinds of scheduling hypothesis (synchronous and probabilistically asynchronous) and different sizes of color domain. We study the impact of these parameters on the stabilization time of distance-k coloring algorithms. This study helps performing the better choice for each parameter according to the use of the coloring scheme.

[1] http://www.ietf.org/html.charters/manet-charter.html

It is worth highlighting that in this paper, we mainly focus on the self-stabilization aspects of this clustering protocol. We chose this particular density-based clustering algorithm because of the applications designed over it such as energy efficient broadcasting [37, 35] or memory and energy efficient hierarchical routing [36]. Nevertheless, our study may be applied to any other k-clustering protocol such as DDR [44].

The outline of the paper is as follows. A brief state-of-the-art on clustering algorithms in multihop wireless networks is given in Section 2. The description of the clustering heuristic we focus on is presented in Section 3. The model, assumptions and notations are presented in Section 4. In Section 5, we describe a distributed algorithm for the clustering heuristic, and discuss some improvements for robustness. We formally prove that this algorithm is self-stabilizing in Section 6. The properties of the proposed algorithm and the various improvements are also evaluated by simulations in Section 7.

2 Related Work

Flat routing protocols (like the classical reactive or proactive protocols) are not really suitable for large wireless multihop networks. Indeed, such routing protocols become ineffective on a large scale because of bandwidth (flooding of control messages) and processing (routing table computation) overhead. One solution to solve this scalability problem is to introduce a hierarchical routing by gathering geographically close nodes into clusters [24]. Clustering has been very studied these latter years and many techniques for cluster formation and cluster-head selection have been proposed. A very good and detailed review on clustering techniques is available in [52].

First solutions such as [51] build clusters in a centralized way and assume that nodes are static and can adapt their range. Then, more scalable solutions have appeared and have proposed to build clusters in a distributed way. Most solutions aim to identify subsets of nodes within the network and to bind each of them to a unique leader. They differ in the criterion used to elect the most appropriate node as a leader. Some of them gather nodes into homogeneous clusters by using an identity criterion (*e.g.*, the lowest identity [2, 47]), a fixed connectivity criterion (for instance maximum degree [9, 44], k-hops clusters [16]) or a value computed from different metrics (as connectivity and identity criteria in max-min d-cluster [1], [6]). To maintain the cluster structure, most of the solutions try to keep either a fixed cluster diameter [1], a fixed cluster radius [26, 47], a constant number of nodes in each cluster [46, 50] or a suitable link quality [19]. Most of the clustering techniques build 1-clusters, which means that every node is neighboring a cluster-head. This is for instance the case in [2] or [9]. 1-clusters provide many possibilities of utilizations as resource management but are not very adapted for large scale multihop wireless networks. Indeed, a small modification in the network topology (due to the mobility of one node for instance) often implies new computations to build the new clusters and to elect the cluster-heads. To overcome this drawback, MacDonald and Znati [33] have introduced a smart approach in which every node does not necessarily belongs to a cluster and will join or not based on the probability that a path exists. Moreover, building and maintaining clusters with a constant feature (like the diameter or the number of nodes) may generate a significant number of useless clusters. For instance, why separating a set of nodes that can communicate just because it can not fit into one cluster since the constant feature is not respected?

Therefore, we are more interested in our case in clustering heuristics building k-clusters (clusters in which every node is at most k hops away from its cluster-head) and not based on *a priori* features. k-clustering [44, 1, 47, 50, 16, 34] algorithms are more recent and thus less numerous than algorithms building 1-clusters.

To the best of our knowledge, only two of them do not fix the cluster radius: DDR [44] and the density-based algorithm [34]. In both algorithms, the cluster radius depends on the topology and may be different from one cluster to another one. DDR uses the node degree to elect the cluster-heads while in [34], a new metric, a density criteria is introduced. This density metric allows to limit the exchanged traffic generated while clusters are re-built and the nodes' tables updated. Therefore it presents good properties of robustness. The density metric has been studied in [34] with both simulations and a stochastic analysis. It outlines that the number of cluster-heads computed with this metric is bounded and decreases when the nodes intensity (number of nodes per surface unit) increases. This is an advantage because if many nodes are in the communication range of each other, there is no need to separate them into different clusters as they can hear each other. Moreover, this heuristic has revealed to be more stable towards node mobility than other metrics, like the degree in DDR [44] and the Max-Min [1] heuristics, as proved in [34].

In this paper, we focus on the density-based heuristic to prove self-stabilization because it is the only one from the literature which has been used for further applications such as broadcasting [35] and indirect routing [36]. These applications have been proven to be energy-efficient and memory-aware. In addition, self-stabilization transparently enables self-organization and fault-tolerance, while the additional property of *local* stabilization enables scalability. Nevertheless, note that the self-stabilization scheme that we introduce here can also be added to any other k-clustering protocol. It is worth noting that in this paper, we mainly focus on the self-stabilization aspects of such algorithms and thus do not return on the cluster properties exhibited by the clustering algorithms.

There exists a number of self-stabilizing distributed schemes for self-organizing a network, such as node coloring [17, 4, 3, 32], link coloring [31, 32] maximal independent set [20], maximal matching [29, 28], etc. However, the high level presentation of those solutions (most use the local shared memory model) and the use on system-wide identifiers (in the case [17, 29, 28]) create an artificial dependency betwen the stabilization time of the system and its size. This complexity issue is clearly impractical for large-scale networks we consider in this paper.

In order to ensure local stabilization in a low time, we add a coloring process to the clustering algorithm. The vertex coloring problem, which is one of the most studied problems in 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. Then, from the color distribution, a DAG (Directed Acyclic Graph) can be easily and locally drawn by (virtually) drawing a directed edge from each node to its only neighbor with the lowest color (if such a node exists). Afterward, this DAG can be used in different applications. For example, the self-stabilizing distributed vertex coloring algorithm is used in sensor networks for resource allocation [27, 11], distributed local organization [18, 23], routing [25, 21], or local topology construction [32]. The coloring process being local provides additional advantages: it is possible to easily extend to scheme to support *malicious* failures [42, 43, 30, 31], *i.e.* nodes that deviate from their behavior in order to harm the system.

The coloring problem itself has already been extensively studied in the literature but most of the works aim either to reduce the number of colors [15] or to find a coloring algorithm that minimizes the complexity with a given number of colors [22] or to minimize both the complexity and the number of colors [41]. In our case, we are not interested in minimizing the exact number of colors, but the order of the number of colors. We provide a formal analysis which gives the average number of steps before stabilization given a number of colors and we characterize the impact of the coloring parameters on the stabilization time and the induced DAG height.

3 Density driven clustering

3.1 System model

The system is composed of a set V of n nodes in a multihop 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. Note that p does not belong to N_p ($p \notin N_p$). In the wireless model, transmission is omnidirectional: 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 when sensor nodes are randomly spread on the area to be monitored, the average degree of a node is $\tilde{\delta} = \lambda \pi R^2$ where R is the communication radius and λ is the spatial intensity. 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. We note $\mathcal{H}(p)$ the cluster-head of the cluster owning node p.

3.2 The density metric criteria

The notion of *density*, firstly introduced in [34], characterizes the *relative* importance of a node in the network and in its 1-neighborhood. The underlying idea is that if some nodes move in N_p, changes will affect the microscopic view of node p (for instance its degree $|N_p|$ will change) but its macroscopic view will not drastically change since globally the network does not change and its N_p globally remains the same. The density will smooth changes down in N_p by considering the ratio between the number of links and the number of nodes in the 1-neighborhood. This can be achieved through the use of Hello Packets in which the neighborhood table is piggybacked like in classic Hello-based protocols like OLSR. The definition is the following:

Definition 1 *The* density *of a node $p \in V$ is*

$$d_p = \frac{|\{e = (v,w) \in V^2 \text{ s.t. } w \in \{p\} \cup N_p \text{ and } v \in N_p\}|}{|N_p|}$$

3.3 Cluster-head selection and cluster formation

Each node watches its neighborhood. If this one changes at each step, the node is considered as too mobile to be integrated into a cluster and does not participate to the clustering algorithm (or it may cause instability into the cluster formation). Otherwise, it locally computes its density value and regularly broadcasts it to all its 1-neighbors. Each node is thus able to compare its density value to its 1-neighbors' and decide by itself whether it joins one of them (the one with the highest density value) or it wins in its 1-neighborhood and elects itself as a cluster-head. If there are some joint winners (at least two neighbors with the same highest density value), the smallest identity is used to decide between them. In this way, two neighbors can not be both cluster-heads. If node p has joined node w, we will say that w is node p's parent. A node's parent can also have joined another node and so on. The cluster-head will be the node which has elected itself as its own cluster-head. A cluster can then extend itself until it reaches a cluster frontier. As each node joins a node within its 1-neighborhood, the cluster can also be seen as a directed tree where the cluster-head is the root (that we will denote as *clustering* tree in the following).

Figure 1(a) shows a graph example for which Table 1(b) gives the node density and Figure 1(c) plots the clusters and trees built by the clustering algorithm.

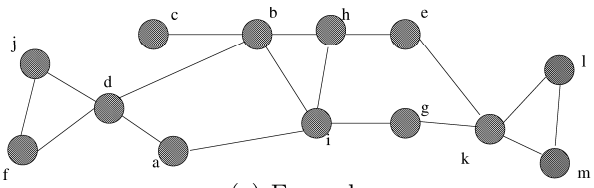
It is worth noting that the role of the clusterhead greatly depends on the use which is made of the cluster organization. This density-based algorithm has already been well studied and analyzed in [34, 35]. It has been compared to several other clustering schemes from the literature and outperforms them in terms of reliability, homogeneous construction and stability towards node mobility and scalability. Two main applications of the density-based clustering have been proposed in the literature: broadcasting [35] and indirect routing [36]. The broadcasting performed over this structure has been shown to be energy efficient. The indirect routing has been proven to need only $O(1)$ memory space and to be energy efficient as well. In both applications, clusterheads do not have a bigger role than other nodes and thus should not run out of energy quicker. Their main function is to name the cluster. Indeed, in all these applications, clusterheads do not even need to communicate directly.

It is also to be highlighted that in this paper, we focus on the benefits of the introduction of the self-stabilization scheme described in Section 5. We chose to apply this scheme to the density-based algorithm because of the good features it has already exhibited. Nevertheless, our self-stabilization scheme could be applied to any similar k-clustering scheme such as [44, 1]. Therefore, the study of this clustering scheme as well as any comparison to any other algorithm is out of the scope of this work.

4 Notations

We follow the same assumptions and principles as the ones given in [18].

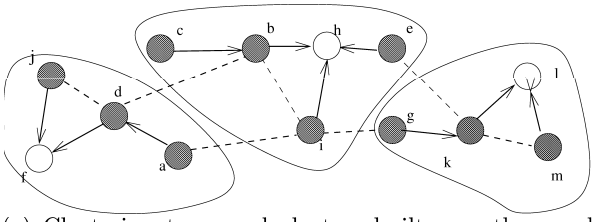
Hypothesis. We assume that the implementation of CSMA/CA satisfies the following point: there exists a constant $\tau > 0$ such that the probability of a frame transmission without collision is at least τ (this corresponds to typical assumptions for multi-access channels [5]; the independence of τ for different frame transmissions indicates that we assume a memoryless probability distribution in a Markov model). This assumption is justified in Appendix.



(a) Example.

Nodes	a	b	c	d	e	f	g	h	i	j	k	l	m
Degree	2	4	1	4	2	2	2	3	4	2	4	2	2
# links	2	5	1	5	2	3	2	4	5	3	5	3	3
Density	1	1.25	1	1.25	1	1.5	1	1.33	1.25	1.5	1.25	1.5	1.5

(b) Density of nodes of graph plotted in Figure 1(a).



(c) Clustering trees and clusters built over the graph plotted in Figure 1(a) (Cluster-heads appear in white, arrows symbolize the edges in the clustering tree, dotted lines are other wireless links).

Figure 1: Illustration of the clustering algorithm

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. Some guards can be event predicates that hold upon the event of receiving a message. 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 Semantics. The life of computing at every node consists of the infinite repetition of evaluating its guarded actions. We assume that every action is executed within a constant time finding a guard and executing its corresponding assignment or skipping it when the guard is *false*. Generally, we suppose that when a node executes its program, all statements with *true* guards are executed within a constant time (done, for example, in round-robin order).

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 [18]. 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.

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 our coloring process analysis, 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.

5 A locally self-stabilizing solution

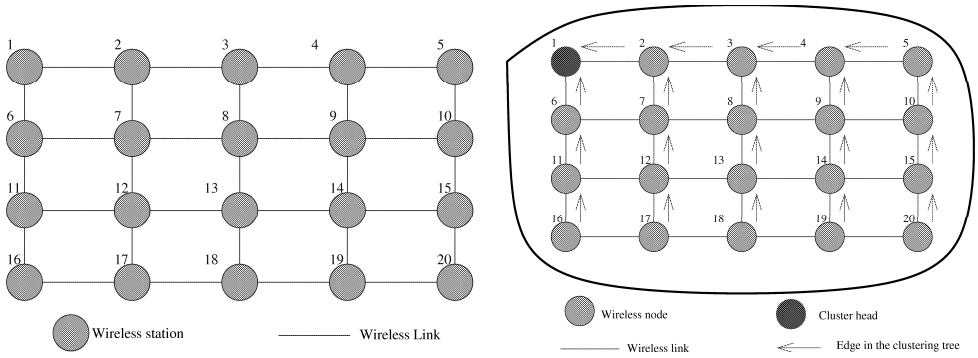
In this section we present a self-stabilizing algorithm that implements the density-driven clustering heuristic presented in Section 3. We consider that the algorithm stabilizes when each node knows to which cluster it belongs, *i.e.* when it knows its cluster-head's identity. The stabilization time is thus related to the depth of the clustering trees.

5.1 Identifying the worst case

In the chosen clustering algorithm, as in every clustering algorithm using the node identity as the last decision, as it is also the case for the DDR heuristic [44], the worst case is encountered *(i)* when every node has the same deciding value, *i.e.* in our case the density value, and *(ii)* when node identifiers are unique in the network and badly distributed. Such a topology is illustrated in Figure 2(a). Indeed, in a grid (or in a chain), every interior node has the same degree and the same density value. Therefore, when running the clustering algorithm, a node will only use the node identity to elect its parent. If the identifiers are badly distributed as in Figure 2(a), the algorithm builds only one cluster which may have a diameter as big as the diameter of the network as illustrated in Figure 2(b). This may cause scalability problems because, even if the system eventually self-stabilizes, the stabilization time is likely to depend on this diameter, which is not scalable. Moreover, it is obvious that building such a cluster is useless as we could have used the network without clusters instead.

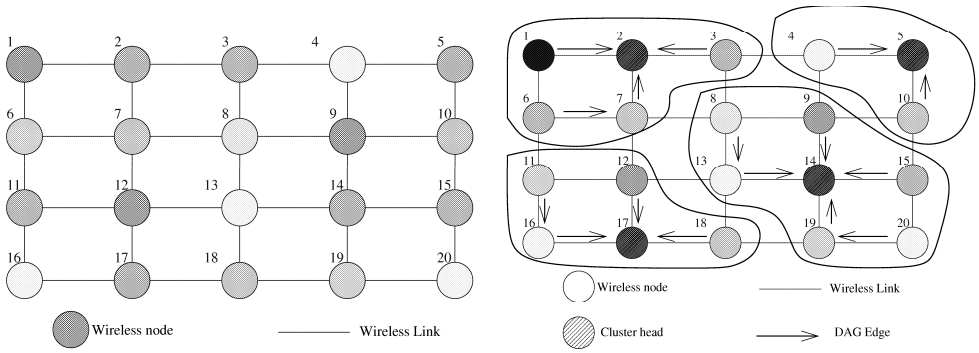
This high complexity is due to the fact that network-wide unique identifiers are used to arbitrate symmetric situations, which appears to be a drawback in many situations concerning self-stabilization [48].

To overcome this drawback, it can be useful to give nodes virtual names or colors, from a constant space of colors, in a way which ensures that colors are locally unique. If we suppose that these colors are directed, a DAG (Directed Acyclic Graph) can be constructed by using them and by orienting edges between neighbors from the highest color to the lowest one. Then, during the clustering process, in case of tie of density value, nodes do not use the node Id anymore but the colors. Figure 3 shows how the worst case is solved in this way.



(a) No stabilizing topology (b) Clustering result

Figure 2: Worst case illustration.



(a) A possible coloring in the grid (b) Clustering result

Figure 3: Solving the worst case illustration.

Figure 3(a) shows a possible coloring for the grid. Figure 3(b) shows the result of the clustering algorithm run over the grid. The cluster radius and thus the stabilization time are much more reduced.

5.2 Coloring Algorithm and DAG construction

We call a coloring at distance k or a k-coloring, $k \in I\!R^*$, a coloring where each node has a unique color in its k-neighborhood.

The coloring algorithm N1 that we consider is the one used in [18] but uses a much smaller color-space Ω. $|\Omega|$ is equal to Δ^6 in [18] with $\Delta = \max_{p \in V} |N_p|$, while Δ^2 is sufficient for a coloring at distance 1 since in this case a node can be in conflict with at most Δ other nodes.

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

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

The algorithm for vertex coloring is the following:

$$\mathsf{N1}{:}\mathit{true} \;\rightarrow\; \mathcal{C}_p := \mathsf{newC}(\mathcal{C}_p)$$

As colors are ordered and unique in a node neighborhood, such a coloring algorithm can lead to the construction of a 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 $\mathcal{C}_w \neq \mathcal{C}_v$. 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 [18], the authors show that Algorithm N1 self-stabilizes with probability 1 and has constant expected local stabilization time.

Indeed, adding this coloring process in the clustering algorithm aims at reducing the stabilization time. Therefore, the stabilization time of the coloring process must be very low and negligible in front of the stabilization time of the clustering stabilization time. In Section 6, we analyze theoretically and by simulation the stabilization time of this coloring process in order to ascertain this feature. As we will see, this coloring process indeed has a low stabilization time. But, first, let see how we use the colors in the density-based clustering algorithm.

5.3 DAG Utilization for Density-driven Cluster Construction

Each node p maintains two shared variables, denoted by d_p and $\mathcal{H}(p)$. d_p denotes the density value of node p given in Definition 1. $\mathcal{H}(p)$ denotes the cluster-head chosen by p.

We define $\prec$ as a binary total order such that $p \prec q$ if and only if $d_p < d_q$ or $(d_p = d_q) \wedge (\mathcal{C}_q < \mathcal{C}_p)$.

Let $\max_{\prec}$ denote the maximum function associated to this total order. When a node p computes the result of $\prec$ or $\max_{\prec}$, it uses the cached values of its neighborhood (assuming $\boxtimes\mathcal{C}_p = \mathcal{C}_p$ and $\boxtimes d_p = d_p$).

We now define the clusterHead choice function:

$$\mathsf{clusterHead} = \begin{cases} p & \text{if } \forall q \in N_p, q \prec p, \\ \mathcal{H}(\max_{\prec}\{q \in N_p\}) & \text{otherwise.} \end{cases}$$

The cluster-head algorithm runs as follows:

$$\begin{array}{ll} \mathsf{R1}: & \mathit{true} \;\rightarrow\; d_p := \mathsf{density} \\ \mathsf{R2}: & \mathit{true} \;\rightarrow\; \mathcal{H}(p) := \mathsf{clusterHead} \end{array}$$

5.4 Improving Stability

We improve the stability of the algorithm by adding some selection criteria. First, when two nodes compete for being cluster-heads (they have the same density value), the winner will be, first, the one which was cluster-head before (if it exists), then the one with the lowest DAG color. This scheme adds stability into the cluster organization by limiting cluster reconstruction. Cluster-heads remain cluster-heads as long as possible. It is a good property since the only cluster-heads' role is to give an identity to the clusters. This refinement preserves the structure of our stabilization proof, since it is equivalent to define the total order relation $\prec$ as $p \prec q$ if and only if $(d_p < d_q)$ or $(d_p = d_q) \wedge (\mathcal{H}(q) = q) \wedge (\mathcal{H}(p) \neq p)$ or $(d_p = d_q) \wedge (\mathcal{H}(p) \neq p) \wedge (\mathcal{H}(q) \neq q) \wedge (\mathcal{C}_q < \mathcal{C}_p)$. In addition, the height of the new $\text{DAG}_{\prec}$ is similar to the height of the previous one.

Second, if a node p is a 1-neighbor of two different cluster-heads u and v (which are not directly linked), it will initiate a fusion between u and v's clusters: if p has chosen v as cluster-head, that means that $u \prec v$ and that v will remain a cluster-head unlike u. This ensures that *(i)* a cluster-head is not too off-centered in its own cluster, *(ii)* a cluster has at least a diameter of two, and *(iii)* that two cluster-heads are distant of at least three hops. Again, this refinement preserves our stabilization proof, since it is sufficient to use the alternative clusterHead function:

$$\mathsf{clusterHead} = \begin{cases} p & \text{if } (\forall q \in N_p, q \prec p) \wedge (\forall q \in N_p^2 s.t. \mathcal{H}(q) = q \implies q \prec p) \\ \mathcal{H}(\max_{\prec}\{q \in N_p\}) & \text{otherwise} \end{cases}$$

The condition for being a cluster-head thus becomes "I am locally maximal (in the sense of $\prec$) and any cluster-head in my 2-neighborhood is smaller than me (and they should not remain cluster-heads)". The remaining of the algorithm (and thus proof) is the same.

6 Proof of correctness

The proof of correctness of our self-stabilizing algorithm is presented in two parts: the proof related to the coloring layer is presented in Section 6.1, while the proof related to the clustering layer is presented in Section 6.2.

6.1 Coloring layer

In our case, we use this coloring protocol in order to ensure a low stabilization time to a clustering algorithm but it could be used independently for any other application in sensor networks, as we already mentioned in Section 2. For example, the self-stabilizing distributed vertex coloring algorithm can be used for resource allocation [27, 11], distributed local organization [18, 39, 38] in sensor networks. In every case, the stabilization time of the coloring process is required to be low. Indeed, as already mentioned, this stabilization time is very important since it must not counterbalance the performances brought to the stabilization time of the clustering protocol.

From [18], we already know that when the degree is upper bounded by a constant Δ (see Section 4), the expected local stabilization time is also upper bounded by a constant. However, the actual constant is not given in [18], 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 [18], the authors show that the height of the DAG is bounded by $|\Delta| + 1$, where Δ denotes the color domain.

In theory, the coloring algorithm `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). The **all** mode fits well for wireless networks for instance where nodes try to minimize the size of the packets they have to exchange to save bandwidth. In the **all** mode, they do not need to exchange their id. The **all but one** mode may be used for instance in a DHCP network where nodes know the Id of each other [7, 8].

For the theoretical study, we consider a 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 [7, 8] (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 [7, 8] 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 multihop 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 7.2).

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 either exactly one ball in the all mode or at least one ball in the all but one mode. 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 $N = \sum_{n>0} \sum_{i=0}^{L-1} 1_{\{X_n = i\}}$. 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

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

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, where the dependence on L and M has been removed to simplify writing.

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

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 subsections the transition probability matrix of each mode before giving the distribution and the expectation of the stabilization time N of the coloring algorithm.

6.1.1 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.

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

Theorem 1 *In the **all** mode, the transition probability matrix $\mathcal{P}(L,M)$ is given, for $i,j = 0,\ldots,L$, by*

$$\begin{cases} p_{i,j}(L,M) &= 0 \text{ for } i > j \\ p_{i,j}(L,M) &= \displaystyle\sum_{k=0}^{M} a_{L-i,M-i}(k,j-i) \text{ for } i \neq L \text{ and } i \leq j \\ p_{L,L}(L,M) &= 1, \end{cases}$$

where the $a_{L,M}(k,j)$ are given recursively, for $L,M \geq 1$ with $L \leq M$, $k = 0,\ldots,M$ and $j = 0,\ldots,L$, by

$$a_{1,M}(k,j) = 1_{\{k=M-1,j=1\}}$$

and, for $L \geq 2$, by

$$\begin{aligned} a_{L,M}(k,j) &= 0 \text{ for } k = M \text{ or } k+j > M \\ a_{L,M}(k,L) &= 0 \text{ for } k \neq M-L \\ a_{L,M}(k,j) &= \frac{k+1}{M} a_{L-1,M}(k+1,j-1) 1_{\{j\geq 1\}} + \frac{j+1}{M} a_{L-1,M}(k,j+1) 1_{\{j \leq L-2\}} \\ &\quad + \frac{M-(j+k)}{M} a_{L-1,M}(k,j) 1_{\{j \leq L-1\}} \text{ otherwise.} \end{aligned}$$

Proof: As seen previoulsy, we clearly have $p_{i,j}(L,M) = 0$ for $i > j$ and $p_{L,L}(L,M) = 1$. 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:

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

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:

$$\begin{aligned} a_{L,M}(k,j) &= \frac{k+1}{M} a_{L-1,M}(k+1,j-1) 1_{\{j\geq 1\}} + \frac{j+1}{M} a_{L-1,M}(k,j+1) 1_{\{j\leq L-2\}} \\ &\quad + \frac{M-(j+k)}{M} a_{L-1,M}(k,j) 1_{\{j\leq L-1\}} \end{aligned}$$

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

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

It is also easy to check that $a_{L,M}(k,j) = 0$ if 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. □

Let us now compute the transition probability matrix $\mathcal{P}(L,M)$ in the **all but one** mode. This case is the one given in [7, 8]. In that 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.

Theorem 2 *In the* ***all but one*** *mode, the transition probability matrix* $\mathcal{P}(L,M)$ *is given, for* $i,j = 0,\ldots,L$, *by*

$$\begin{cases} p_{i,j}(L,M) &= 0 \text{ for } i > j \\ p_{i,i}(L,M) &= 0 \text{ for } i \leq L-1 \\ p_{i,j}(L,M) &= \binom{M-i}{j-i} \sum_{k=0}^{j-i} \binom{j-i}{k} (-1)^k \left(\frac{j-i-k}{M-i}\right)^{L-i} \text{ for } i < j \\ p_{L,L}(L,M) &= 1. \end{cases}$$

Proof: The result is trivial for $i \geq j$. For $i < j$, it is 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 [14]. According to (1), we then have, for $i < j$,

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

□

6.1.2 Distribution and expectation of the stabilization time N.

Now, given the transition probability matrix $\mathcal{P}(L,M)$ of the Markov chain X, we are able to determine the distribution $(\mathbb{P}[N=n])_{n\geq 0}$ and the expected value $\mathbb{E}[N]$ of the random variable N for both modes (**all** and **all but one**). 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.

This calculus is detailed in [7]. We directly use it here. It is 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. In both modes (**all** and **all but one**), the matrix Q is upper triangular, so the expectation $\mathbb{E}[N]$ can be easily computed as follows.

We introduce the column vector $V = (V_i)_{0<i<L-1}$ defined by $V_i = \mathbb{E}[N|X_0 = i]$. Since $X_0 = 0$ with probability 1, 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 Q is upper triangular, as

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

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

We are now able to compute $\mathbb{E}[N] = V_0$ by recurrence from V_i, which actually is the mean expected time of the coloring process. We use these theoretical results in the next sections to compare with the simulation outcomes.

6.2 Clustering Layer

Now we theoretically computed the stabilization time of the coloring process, we have to analyze the stabilization time of the clustering process.

Lemma 1 *Each node p has a correct density value d_p within an expected constant time.*

Proof: After an expected constant time, each node p has a correct view of its neighborhood at distance two. Then, after R1 is executed, the density d_p of p is correct. □

Lemma 2 *Each node p has a correct cluster-head value $\mathcal{H}(p)$ within an expected constant time.*

Proof: Assume that all nodes have correct density values (this is true after an expected constant time by Lemma 1). After the shared variable d_p has been communicated without collision to all nodes in N_p (this occurs in an expected constant time), each node has a correct cache value of all density values in its neighborhood. We now consider the DAG induced by the $\prec$ relation (thereafter denoted by DAG$_\prec$). In an expected constant time, the roots of DAG$_\prec$ have a correct cluster-head value (that is their own identifier). Now assume that all nodes up to distance n from the roots of the DAG$_\prec$ have a correct cluster-head value. When R2 is executed on nodes at distance $n+1$ from the roots of DAG$_\prec$, those nodes get a correct cluster-head value (because the cluster-head is deterministically determined *(i)* by the density and local topology – which are fixed – and *(ii)* by the cluster-head values of nodes at distance up to n from the roots of the DAG$_\prec$). By induction, the time needed for stabilization is proportional to the height of the DAG$_\prec$.

We now prove that the height of the DAG$_\prec$ is bounded by a constant value. Node colors are bounded by a constant $|\Omega|$. The number of edges in the neighborhood at distance one is bounded by Δ^2, the number of neighbors at distance one is bounded by Δ, so the number of possible values for the density function is at most Δ^3. Overall, the color-space of values in the DAG$_\prec$ is $|\Omega|\Delta^3$, which is bounded by a constant (see Section 4). As a result, the height of the DAG$_\prec$ is also bounded by a constant.

The algorithm stabilizes in an expected time proportional to the height of the $\text{DAG}_{\prec}$, and the height of the $\text{DAG}_{\prec}$ is constant, so the expected time for stabilization is also constant. □

Thus, this analysis shows that the stabilization time of the clustering process, and so of any protocol based on a coloring process, is directly proportional to the height of the DAG induced by the colors. The authors of [40] provide a theoretical analysis of the height of the DAG as a function of the size of the color domain size $|\Omega|$ and the number of nodes in the worst case. For such assumptions, the worst case is encountered in a chain, when nodes can only be in conflict with at most two other nodes.

7 Simulations

As in Section 6, we present our simulations for the two layers of the self-stabilizing algorithm respectively in Sections 7.2 and 7.3.

7.1 Simulation model

We use a home-made simulator that assumes an ideal MAC layer. Considering an ideal MAC layer allows us to extract problems relative to our protocols only without being mistaken by some problems occurring at a lower layer. Nodes are randomly deployed using a Poisson Point process with different intensity levels λ^2 in a 1×1 square with a transmission radius R. From it, two nodes u and v are considered as neighbors if the Euclidean distance between u and v is lower than R, following the Unit Disk Graph model [10] and the assumptions of Section 4. Each given statistic is the average over 1000 simulations. Each node chooses a color by running Algorithm N1 and then builds a DAG like stipulated in Section 5.2.

7.2 Coloring layer

In this section, we perform 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.

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 $c \in \Omega$ is said available for a node u if $\forall v \in N_u^k, \mathcal{C}_v \neq c$. As mentioned in Section 4, 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).

[2] In such process, λ corresponds to the mean number of nodes per surface unit and the mean node degree δ is such that $\delta = \lambda \pi R^2$.

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 Ω. The initial stabilizing algorithm has been designed using $|\Omega| = (max_{p \in V}|N_p|)^{2 \times k}$, 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 $|\Omega| = 2 \times \left(max_{p \in V}|N_p^k|\right)$. 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 $\forall p \in V$, $|\Omega_p| = (|N_p|)^{2 \times k}$. 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 schedulers, the number of steps before stabilization in probabilistic modes is divided by n.

In the geometric approach, nodes are randomly deployed using a Poisson Point Process in a 1×1 square with various levels of intensity λ (from 500 to 1000). 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 6.1, we expect that our theoretical results give an accurate lower bound on the stabilization time of the coloring algorithm. Indeed, each node stops running the coloring algorithm when it has kept aside all its neighbors. In the theoretical approach, as we consider a complete graph, when a node x has been kept aside, it has been kept aside by all its neighbors. That means that it has found its definitive color and, as the graph is complete, every node in the graph is neighbor of node x, every node is aware of the color of x and will not choose it anymore. Thus, nodes and colors kept aside will never run again the coloring process.

But, in a non-complete graph, balls and urns kept aside should re-integrate the coloring process as illustrated on Figure 4 (for distance-1 coloring). Let's run the algorithm over the graph plotted on Figure 4, 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. Black lines symbolize wireless links between nodes, *e.g.* node B has two neighbors, A and C. The example shows the colors drawn at each step by each node. At the end of Step 2, node B is not in conflict with node C and thus, C and color 3 can be kept apart by node B. But node C has a conflicting color with node D and thus has not stopped the coloring process yet. C and B both choose another color and may draw the same one. So, the coloring process has not stabilized yet whereas node B has kept aside all its neighbors. Therefore our theoretical analysis only leads to a lower bound.

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 5. As we

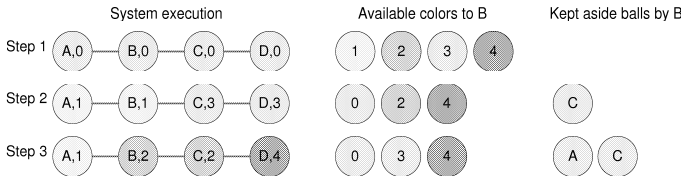
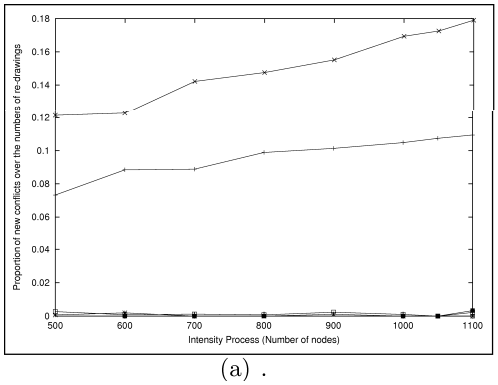
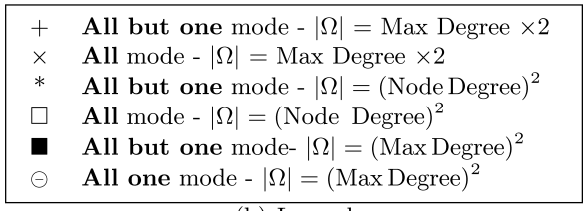


Figure 4: A possible execution for distance-1 coloring in a synchronous network.

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) .



(b) Legend

Figure 5: Proportion of nodes creating a new conflict over the number of nodes choosing another color with a synchronous scheduler.

Tables 1 and 2 compare analytical and simulation results for the Synchronous scheduling in both modes, when using two different color domain sizes $|\Omega|$, 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 6 presents the stabilization time for each scheduling hypothesis and both modes when using different sizes of color domain, for a random spatial distribution of nodes. The most striking result compared to those of [18] is that the stabilization time is much lower than expected. From the results of [18], the stabilization time is at least linear in $|\Omega|$ (being upper bounded by a constant when δ also is a constant). In contrast, our simulation results show a sub-linear (in $|\Omega|$) stabilization time, since considering $|\Omega| = 2 \times max_{p \in V} |N_p|$ *vs.* $|\Omega| = max_{p \in V} |N_p|^2$ 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,

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 $|\Omega| = (max_{p \in V}|N_p|)^2)$ and $|\Omega| = 2 \times (max_{p \in V}|N_p|)$ 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

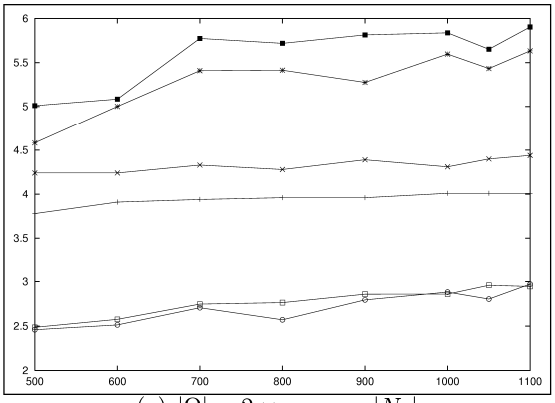
Table 2: Theory and simulations results for the stabilization time for synchronous distance-1 coloring with $|\Omega| = 2 \times (max_{p \in V}|N_p|)$ in random geometry topologies.

the stabilization time remains upper bounded by a constant.

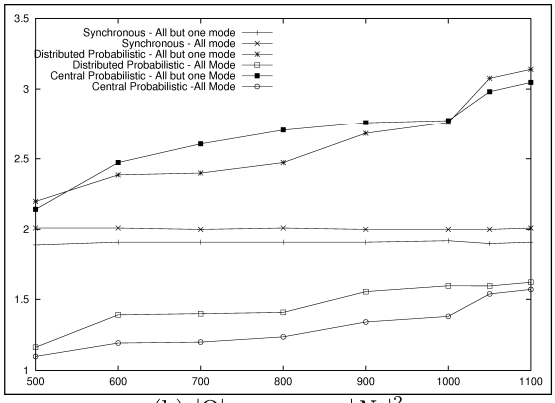
In all cases, whatever the coloring distance k and the color domain Ω, we can note that the behavior of both probabilistic schedulers is similar. With the probabilistic scheduling hypothesis, in order to stabilize, the scheduler has to choose a node in conflict. In the **all but one** mode, only one 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 schedulers 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 7(a) and 7(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.

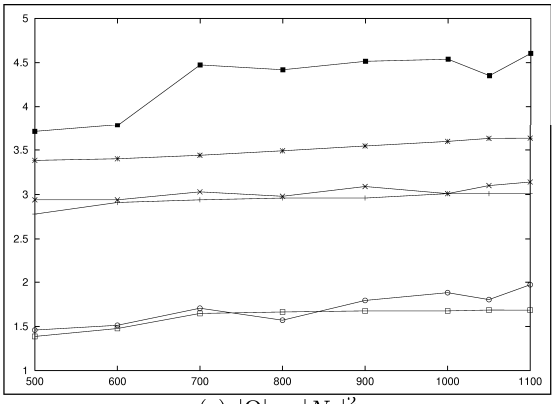
Results clearly show that a higher domain size $|\Omega|$ induces a lower stabilization time and a higher DAG. There thus is a trade-off to do between these two characteristics depending of



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



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

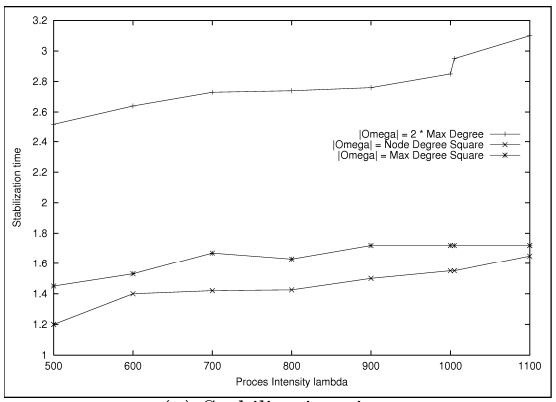


(c) $|\Omega| = |N_p|^2$

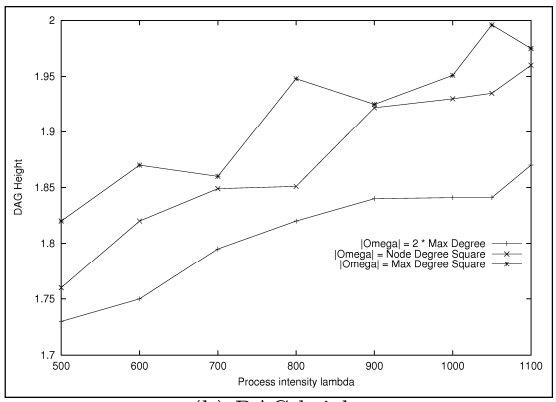
+ Synchronous - **All but one** mode
× Synchronous - **All** mode
* Distributed probabilistic - **All but one** mode
□ Distributed probabilistic - **All** mode
■ Central probabilistic - **All but one** mode
o Central probabilistic - **All** mode

(d) Legend

Figure 6: 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.



(a) Stabilization time



(b) DAG height

Figure 7: Influence of the color domain size on the stabilization time and the DAG height with a distributed probabilistic scheduler in the **all** mode.

the application that will use the coloring. A formal treatment for computing the average height of the DAG has been presented in [40], where analytical bounds show that the height of the DAG is upper bounded by a constant value.

Influence of the coloring distance. Figure 8 plots the stabilization time for distance k-coloring with the **all** mode, assuming the distributed probabilistic scheduler in 4-neighbor and 8-neighbor grids, respectively. The used color domain size is $|\Omega| = (max_{p \in V}|N_p|)^{2 \times k}$. 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 W (See Appendix). In the worst case, this subgraph is a tree where each node has degree Δ except for the leaves that have degree 1, overall Δ^{k-1} nodes. After one time unit, $1/W$ of the nodes have transmitted correctly. After one extra time unit, $1/W$ of the *remaining* nodes have transmitted correctly. So, after $\log_W(\Delta^{k-1})$ 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 $k \times \log_W(\Delta^{k-1}) = O(k^2)$, nodes are aware of the colors of their k-neighborhood. In Figure 8, values have thus been multiplied by k^2.

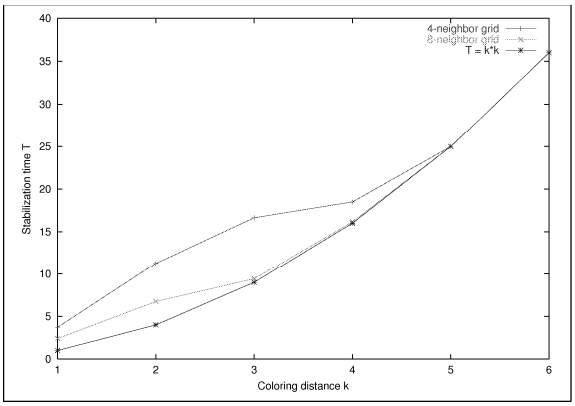


Figure 8: Stabilization time with respect of time for message transmission as a function of the coloring distance.

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.

7.3 Clustering layer

As mentioned in Section 5, we suppose that there exists a constant $\tau > 0$ such that the probability of a frame transmission without collision is at least τ. Yet, we can suppose that in a bounded time W, each node is able to locally broadcast one frame and then receive all packets sent by its 1-neighbors. We justify this assumption in Appendix. We call such a W time unit a step, during which each node can receive each packet of all its 1-neighbors. After

one step, each node can discover its 1-neighbors. After two steps, each node can discover its 2-neighbors and then compute its density. After only three steps, each node knows its parent. Then, the number of steps required to discover its cluster-head identity directly depends on the distance in number of hops from the node to its cluster-head in the clustering tree and is bounded by the depth of this tree.

We performed simulations in order to estimate the importance of the introduction of the DAG. Note that the coloring obtained by Algorithm N1 does not depend on the kind of scheduler neither the mode used for coloring. This settings only impact the stabilization time of the coloring process. Thus, since for clustering we use only the result of the coloring process, the way colors have been assigned does not matter in this section.

We measure the following criteria: number of cluster-heads per surface unit, clustering tree height (also in order to evaluate the stabilization time since both are proportional) and cluster-head eccentricity. Table 3 shows these criteria for $\lambda = 1000$ and different values of R. Note that results are given for $\lambda = 1000$ but simulations have shown that the algorithm behavior the same way for different values of λ.

We note $e(\mathcal{H}(u)) = \max_{v s.t. \mathcal{H}(v)=u}(d(\mathcal{H}(u), v))$ in number of hops, the *eccentricity* of the cluster-head of node u inside its cluster, where $d(u, v)$ is the minimum number of hops to reach v from u. Whatever the transmission radius is (and so the node degree), we can note that the mean cluster-head eccentricity and tree height do not vary too much. This confirms our assumption that the transmission of the cluster-head identity can be expected within a constant and low time. At last, let's note that in the cases where nodes and node Id are homogeneously and randomly distributed, the use of the DAG does not bring much help. This is due to the fact that in such a node distribution, a node uses very rarely the Id to choose its parent because density values are well-distributed and rarely equal.

We now consider a scenario where nodes are distributed over a grid with Ids increasing from left to right and from the bottom to the top. All interior nodes have the same density value and the only criterion used to select a cluster-head is the Id. As the node Ids are not well distributed, all nodes will finally join the same head. Table 4 shows the obtained results in this case. One can note that the DAG construction is very useful in such a case as it allows to drastically reduce the number of steps needed before stabilization. Figure 9 shows an example of clusters organization obtained for a radius equal to $R = 0.05$ (One color per cluster). Cluster-heads appear in blue. On Figure 9(a), the DAG is implemented and several clusters are created. On Figure 9(b), the DAG is not implemented and only one cluster is created.

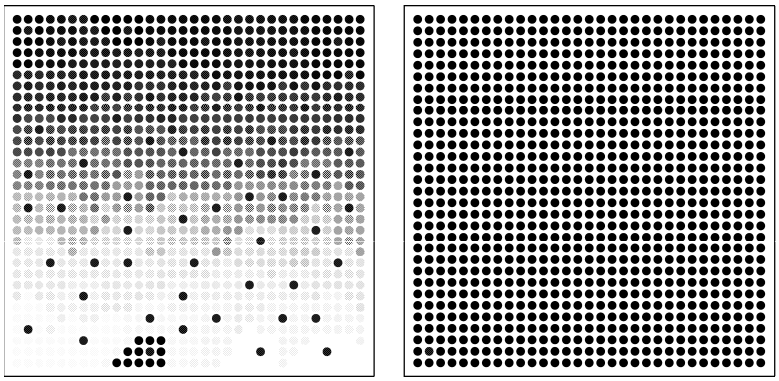
	$R = 0.05$		$R = 0.08$		$R = 0.1$	
	With DAG	No DAG	With DAG	No DAG	With DAG	No DAG
# clusters	61.0	61.4	19.2	19.5	11.7	11.7
Eccentricity of a node	2.6	2.6	3.1	3.1	3.2	3.2
average tree length	2.7	2.7	3.3	3.3	3.5	3.5

Table 3: Cluster features on a random geometric graph for $\lambda = 1000$.

We have also tested the extra criteria given in Section 5.4 to improve the stability. We performed simulations where nodes move randomly at a randomly chosen speed during 15 minutes. We computed the percentage of cluster-heads which remain cluster-heads after each

	$R = 0.05$		$R = 0.08$		$R = 0.1$	
	With DAG	No DAG	With DAG	No DAG	With DAG	No DAG
# clusters	52.8	1.0	29.3	1.0	18.5	1.0
Eccentricity of a node	3.4	29.1	4.1	19.1	3.6	6.5
average tree length	3.7	83.4	4.7	100.5	4.5	32.1

Table 4: Clusters characteristics on a grid for $\lambda = 1000$.



(a) With DAG (b) No DAG

Figure 9: Clustering example in grid with $R = 0.05$.

2 seconds. For a node mobility between 0 to $1.6m/s$ (for pedestrians) the percentage of cluster-heads reelection is about 82% with our improvement rules and 78% without. For a mobility from 0 to $10m/s$ (for cars in a very anarchic world) this percentage is 31% with the new rules against 25% without. Thus, our improvements are useful in terms of cluster-head stability.

8 Conclusion

In this paper, we have addressed several issues concerning the self-stabilization on the clustering process in multihop wireless network. We have proved that the clustering algorithm based on the density criteria, defined in [34] is self-stabilizing. We have proposed different enhancements to reduce the stabilization time and to improve stability of the cluster-heads. Note that our contribution regarding the self-stabilization could be applied to several clustering metrics as for instance the node's degree [9, 44].

We use for this a coloring protocol. Distance-k coloring is a useful mechanism for multihop wireless networks. In [18], distance-3 coloring was used to construct a TDMA schedule and in [39], 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 [7] to anonymous networks (**all** 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 multihop wireless networks are low, which allows such a coloring to be used in order to reduce the stabilization time of a clustering algorithm.

In the future, several possible extensions of this work are open to investigation. It could be interesting to derive sharp bounds on the stabilization as a function of the mobility, *e.g.*, speed of the nodes, mobility model, frequency of links failure, etc. Based on these bounds, we also plan to study hierarchical self-stabilizing algorithms. The underlying idea is to decompose the algorithm in macro steps and to authorize to execute the macro step k if and only if the macro step $k-1$ is stable. In our case, if the neighborhood of a node p is not stable, *i.e.*, if node p is too mobile and/or its neighbors are evolving too fast, then the node p is not allowed to perform the next steps and can not compete for clusters organization. Finally, we also want to consider energy constraints in the stabilization algorithm and we are investigating energy-efficient organization algorithms.

References

[1] A. Amis, R. Prakash, T. Vuong, and D. Huynh. Max-Min D-cluster formation in wireless ad hoc networks. In *Proc.* 19^{th} *Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM).* IEEE Press, March 2000.

[2] P. Basu, N. Khan, and T. Little. A mobility based metric for clustering in mobile *ad hoc* networks. In *DCS Workshop*, 2001.

[3] S. Bernard, S. Devismes, K. Paroux, M. Potop-Butucaru, and S. Tixeuil. Probabilistic self-stabilizing vertex coloring in unidirectional anonymous networks. In *Proceedings of ICDCN 2010*, Lecture Notes in Computer Science, Kolkata, India, January 2010. Springer Berlin / Heidelberg.

[4] S. Bernard, S. Devismes, M. Gradinariu Potop-Butucaru, and S. Tixeuil. Optimal deterministic self-stabilizing vertex coloring in unidirectional anonymous networks. In *Proc. IEEE International Conference on Parallel and Distributed Processing Systems (IPDPS)*, Rome, Italy, May 2009.

[5] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, 1987.

[6] M. Chatterjee, S. K. Das, and D. Turgut. WCA: A Weight based distributed Clustering Algorithm for mobile ad hoc networks. *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, 5(2):193–204, April 2002.

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

[8] G. Chelius, E. Fleury, and L. Toutain. No admistration protocol (nap) for ipv6 router auto-configuration. *International Journal on Internet Protocol and Technology*, 1(2):101–108, September 2005.

[9] G. Chen and I. Stojmenovic. Clustering and routing in mobile wireless networks. Technical Report TR-99-05, SITE, June 1999.

[10] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3):165–177, 1990.

[11] P. Danturi, M. Nesterenko, and S. Tixeuil. Self-stabilizing philosophers with generic conflicts. *ACM Transactions of Adaptive and Autonomous Systems (TAAS)*, 4(1), January 2009.

[12] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, 1974.

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

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

[15] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *IEEE Conference on Computational Complexity*, pages 278–287, 1996.

[16] Y. Fernandess and D. Malkhi. k-clustering in wireless ad hoc networks. In *2nd ACM PMC workshop*, 2002.

[17] M. Gradinariu and S. Tixeuil. Self-stabilizing vertex coloring of arbitrary graphs. In *Proc. International Conference on Principles of Distributed Systems (OPODIS)*, pages 55–70, Paris, France, December 2000.

[18] T. Herman and S. Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. In *Proc. of the First Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors'2004)*, number 3121 in Lecture Notes in Computer Science, pages 45–58, Turku, Finland, July 2004. Springer-Verlag.

[19] T.C. Hou and T.J. Tsai. An access-based clustering protocol for multihop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 19(7):1201–1210, 2001.

[20] M. Ikeda, S. Kamei, and H. Kakugawa. A space-optimal self-stabilizing algorithm for the maximal independent set problem. In *Proc. Third International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pages 70–74, Kanazawa, Japan, September 2002.

[21] C. Johnen and S. Tixeuil. Route preserving stabilization. In Shing-Tsaan Huang and Ted Herman, editors, *Self-Stabilizing Systems*, volume 2704 of *Lecture Notes in Computer Science*, pages 184–198. Springer, 2003.

[22] M. Karchmer and J. Naor. A fast parallel algorithm to color a graph with δ colors. *Journal of Algorithms*, 9(1):83–91, 1988.

[23] K. Kothapalli, C. Scheideler, M. Onus, and A. W. Richa. Constant density spanners for wireless ad-hoc networks. In P. B. Gibbons and P. G. Spirakis, editors, *Proc. of the 17th Annual ACM Symposium on Parallel Algorithms (SPAA)*, pages 116–125, Las Vegas, Nevada, USA, July 2005.

[24] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster based approach for routing in dynamic networks. In *ACM SIGCOMM*, pages 49–65, April 1997.

[25] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. End-to-end packet-scheduling in wireless ad-hoc networks. In *SODA '04: Proc. of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1021–1030, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[26] H.-C. Lin and Y.-H. Chu. A clustering technique for large multihop mobile wireless networks. In *IEEE Proc. of Vehicular Technologies Conference VTC*, May 2000.

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

[28] F. Manne, M. Mjelde, L. Pilard, and S. Tixeuil. A self-stabilizing 2/3-approximation algorithm for the maximum matching problem. In Sandeep S. Kulkarni and André Schiper, editors, *Proc. of Stabilization, Safety, and Security of Distributed Systems, 10th International Symposium (SSS 2008)*, Lecture Notes in Computer Science, Detroit, USA, November 2008. Springer-Verlag Berlin Heidelberg.

[29] F. Manne, M. Mjelde, L. Pilard, and S. Tixeuil. A new self-stabilizing maximal matching algorithm. *Theoretical Computer Science (TCS)*, March 2009.

[30] T. Masuzawa and S. Tixeuil. Bounding the impact of unbounded attacks in stabilization. In A. K. Datta and M. Gradinariu, editors, *Proc. of Stabilization, Safety, and Security of Distributed Systems, 8th International Symposium, (SSS)*, volume 4280 of *Lecture Notes in Computer Science*, pages 440–453, Dallas, TX, USA, November 2006. Springer.

[31] T. Masuzawa and S. Tixeuil. Stabilizing link-coloration of arbitrary networks with unbounded byzantine faults. *International Journal of Principles and Applications of Information Science and Technology (PAIST)*, 1(1):1–13, December 2007.

[32] T. Masuzawa and S. Tixeuil. On bootstrapping topology knowledge in anonymous networks. *ACM Transactions on Adaptive and Autonomous Systems (TAAS)*, 4(1), January 2009.

[33] A. MCDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 17(8), August 1999.

[34] N. Mitton, A. Busson, and E. Fleury. Self-organization in large scale ad hoc networks. In *Proc. of The Third Annual Mediterranean Ad Hoc Networking Workshop, MED-HOC-NET*, Bodrum, Turkey, June 2004.

[35] N. Mitton, A. Busson, and E. Fleury. Efficient Broadcasting in Self-Organizing Sensor Networks. *International Journal of Distributed Sensor Networks (IJDSN)*, 2(2), February 2006.

[36] N. Mitton and E. Fleury. Distributed node location in clustered multi-hop wireless networks. In *Technologies for Advanced Heterogeneous Networks: First Asian Internet Engineering Conference, (AINTEC'05)*, volume 3837 / 2005, pages 112 – 127, Bangkok, Thailand, December 2005.

[37] N. Mitton and E. Fleury. Efficient broadcasting in self-organizing multi-hop wireless network. In *4th International Conference on AD-HOC Networks & Wireless (Ad Hoc Now'05)*, Cancun, Mexico, October 2005.

[38] N. Mitton, E. Fleury, I. Guérin-Lassous, B. Séricola, and S. Tixeuil. On fast randomized colorings in sensor networks. In *Proceedings of ICPADS 2006*, pages 31–38. IEEE Press, July 2006.

[39] N. Mitton, E. Fleury, I. Guérin-Lassous, and S. Tixeuil. Self-stabilization in self-organized wireless multihop networks. In *Proc. of the* 2^{nd} *International Workshop on Wireless Ad Hoc Networking WWAN*. IEEE Press, 2005.

[40] N. Mitton, K. Paroux, B. Sericola, and S. Tixeuil. Ascending runs in dependent uniformly distributed random variables: Application to wireless networks. *Methodology and Computing in Applied Probability*, 2009. To appear.

[41] T. Moscibroda and R. Wattenhofer. Coloring unstructured radio networks. In P. B. Gibbons and P. G. Spirakis, editors, *Proc. of the 17th Annual ACM Symposium on Parallel Algorithms (SPAA)*, pages 39–48, New York, NY, USA, 2005. ACM Press.

[42] M. Nesterenko and A. Arora. Dining philosophers that tolerate malicious crashes. In *ICDCS*, pages 191–198, 2002.

[43] M. Nesterenko and A. Arora. Tolerance to unbounded byzantine faults. In *21st Symposium on Reliable Distributed Systems (SRDS 2002)*, page 22. IEEE Computer Society, 2002.

[44] N. Nikaein, H. Labiod, and C. Bonnet. DDR-distributed dynamic routing algorithm for mobile ad hoc networks. In *Proc. 1st ACM international symposium on mobile ad hoc routing and computing MOBICOM*, Boston, MA, USA, November, 20th 2000. ACM.

[45] R. Rajaraman. Topology control and routing in ad hoc networks: a survey. *ACM SIGACT News*, 33(2):60–73, June 2002.

[46] R. Ramanathan and M. Steenstrup. Hierarchically-organized, multihop mobile wireless networks for quality-of-service support. *Mobile networks and applications*, 3:101–119, June 1998.

[47] R. Riedi, P. Druschel, Y. C. Hu, D. B. Johnson, and R. Baraniuk. SAFARI: A self-organizing hierarchical architecture for scalable ad hoc networking networking. Research report TR04-433, Rice University, February 2006.

[48] S. Tixeuil. *Wireless Ad Hoc and Sensor Networks*, chapter Fault-tolerant distributed algorithms for scalable systems. ISTE, October 2007. ISBN: 978 1 905209 86.

[49] S. Tixeuil. *Algorithms and Theory of Computation Handbook, Second Edition*, chapter Self-stabilizing Algorithms. Chapman & Hall/CRC Applied Algorithms and Data Structures. Taylor & Francis, November 2009.

[50] G. Venkataraman, S. Emmanuel, and S. Thambipillai. Size-restricted cluster formation and cluster maintenance technique for mobile ad hoc networks. *International Journal of Network Management*, 17(2):171–104, March 2007.

[51] H. Balakrishnan W. R. Heinzelman, A. Chandrakasan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of the 33rd International Conference on System Sciences (HICSS)*, January 2000.

[52] J. Yu and P. Chong. A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys*, 7(1):32–48, 2005.

[53] 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: Time slot

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 [53], 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 [53]), 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 W, $\mathbb{E}[X]$ also is a constant. Therefore, in average, a node is able to read all shared variables in its neighborhood within constant time.