# Self-stabilization in self-organized multihop wireless networks

Nathalie Mitton, Eric Fleury, Isabelle Guérin-Lassous, Sebastien Tixeuil

# Self-stabilization in self-organized Multihop Wireless Networks*

N. Mitton – E. Fleury – I. Guérin Lassous
INRIA Ares Team - CITI, INSA de Lyon
69621 Villeurbanne, France
FirstName.Name@insa-lyon.fr

S. Tixeuil
LRI - CNRS UMR 8623 & INRIA Grand Large
91405 Orsay, France
Sebastien.Tixeuil@lri.fr

## Abstract

*In large scale multihop wireless networks, flat architectures are not scalable. In order to overcome this major drawback, clusterization is introduced to support self-organization and to enable hierarchical routing. When dealing with multihop wireless networks, the robustness is a main issue due to the dynamicity of such networks. Several algorithms have been designed for the clustering process. As far as we know, very few studies check the robustness feature of their clustering protocols.*

*In this paper, we show that a clustering algorithm, that seems to present good properties of robustness, is self-stabilizing. We propose several enhancements to reduce the stabilization time and to improve stability. The use of a Directed Acyclic Graph ensures that the self-stabilizing properties always hold regardless of the underlying topology. These extra criterion are tested by simulations.*

**keywords:** multihop wireless networks, clusterization, self-stabilization, scalability, density.

## 1 Introduction

Ad hoc networks or wireless sensor networks (wireless multihop networks) are composed of devices that communicate via wireless interfaces. They require no fixed infrastructure and no 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 changes in topology and provides the connectivity between any pair of terminals. If the current wireless cards allow the communication between mobiles that are in communication range, a routing protocol is required to provide the full connectivity of the network. 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 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 [14], "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 far as we know, very few studies check the robustness feature of their clustering protocols. Moreover, when it is the case, the evaluation is driven by simulations and never by a theoretical approach.

In this article, we apply self-stabilization principles over a clustering protocol proposed in [11] and which presents good properties of robustness. With a theoretical approach, which can be applied to several clustering schemes, we show that, under some assumptions, the algorithm is self-stabilizing. We also improve the robustness by adding

*This work is supported by the FNS of the French Ministry of Research through the FRAGILE project [7] of the ACI sécurité et informatique.

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

extra-advanced features and we show that the resulting algorithm is still self-stabilizing. These properties are further validated by simulations. A brief state-of-the-art on clustering algorithms in multihop wireless networks is given in Section 2. The description of the chosen clustering algorithm is done in Section 3. In Section 4, we provide a formal analysis, showing that this algorithm is self-stabilizing. We also discuss some improvements for robustness in this section. The properties of the proposed algorithm and the different improvements are evaluated by simulations in Section 5.

## 2 State-of-the-art

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 [9]. Many techniques for clusters formation and cluster-heads selection have been proposed. All solutions aim to identify subsets of nodes within the network and to bind each of them to an unique leader. Some solutions try to gather nodes into homogeneous clusters by using either an identity criteria (*e.g.*, the lowest identity [2]) or a fixed connectivity criteria (for instance maximum degree [5, 13], $k$-hops clusters [6]) or a value computed from different metrics (as connectivity and identity criteria in max-min $d$-cluster [1], [4]). To maintain the clusters structure, most of the solutions try to keep a fixed cluster diameter [1], a fixed cluster radius [10] or a constant number of nodes in the clusters [15].

These solutions are not adapted to large multihop wireless networks. First, 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. 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? In [11], a density criteria is proposed. This metric allows to limit the exchanged traffic generated while clusters are rebuilt and the nodes' tables updated. Therefore it presents good properties of robustness. In this paper, we show that this algorithm is self-stabilizing and its robustness can be enhanced with some extra rules.

## 3 Density driven clustering algorithm

**The model.** The system is composed of a set $V$ of nodes and each node has an unique identifier. Each node $p$ can communicate with a subset $N_p \subseteq V$ of nodes determined by the range of the radio signal; $N_p$ is called the neighborhood of node $p$. Note that $p$ does not belong to $N_p$ ($p \notin N_p$). We assume that communication capability is bidirectional: $q \in N_p$ iff $p \in N_q$. Define $N_p^1 = N_p$ and for $i > 1$, $N_p^i = N_p^{i-1} \cup \{ r \mid (\exists q \in N_p^{i-1}, r \in N_q) \}$ (let's call $N_p^i$ the $i$-neighborhood of $p$). We assume that the distribution of nodes is sparse: there is some known constant $\delta$ such that for any node $p$, $|N_p| \leq \delta$. Note that a control on density can be done by adjusting their communication range and/or powering off nodes in areas that are too dense.

**The density metric criteria.** The notion of *density*, firstly introduced in [11], 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. The definition is the following:

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

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

**Cluster-heads selection and clusters formation.** Due to space limitations, we describe the heuristic process informally. The algorithm and its analysis are more detailed in [11]. 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, 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).

**Features.** This metric has been studied in [11] with both simulations and a stochastic analysis. It outlined 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 nodes mobility than other metrics, like the degree and the max-min metrics [11].

# 4 Self-stabilization

In this section we study the self-stabilizing properties of the density-driven clustering algorithm presented in [11]. We follow the same assumptions and principles as the ones given in [8]. Due to space limitations, we briefly describe these points. 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.

**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 $\tau$ (this corresponds to typical assumptions for multi-access channels [3]; the independence of $\tau$ for different frame transmissions indicates that we assume a memoryless probability distribution in a Markov model).

**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.** Some variables of nodes are designated as *shared* variables. Following the scheme presented in [8], nodes periodically transmit the values of their shared variables, based on a timed discipline. Beyond periodic retransmission, an assignment to a shared variable causes peremptory transmission: if a statement $G \rightarrow S$ assigns a shared variable, then we suppose that there is a transformation of the statement into a computation that slows execution down so that it does not exceed some desired rate, and also provides randomization to avoid collision in messages that carry the shared variable values. One technique for implementing such $G \rightarrow S$ is presented in [8]. In the remaining of the section, we assume that nodes use this scheme to learn $N_p$ and $N_p^2$. This is because the topology of the networks under considering is dynamic.

## 4.1 Constant Height DAG Construction

In the chosen clustering algorithm, as in every clustering algorithm using the node identity as last decision, 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 nodes' identifiers are unique in the network and badly distributed. In such a case, the algorithm builds only one cluster which may have a diameter as big as the diameter of the network. This may cause scalability problems because the stabilization time is likely to depend on this diameter. Moreover, it is obvious that building such a cluster is useless as we could have used the network without clusters instead. To overcome this drawback, it can be useful to give nodes smaller names (also named colors), from a constant space of names, in a way which ensures that names are locally unique. A DAG (Directed Acyclic Graph) can be constructed by using these identifiers and by orienting edges between neighbors from the higher identifier to the lower one.

Our constant height DAG construction is based on the randomized technique described in [8], but uses a much smaller name-space $\gamma$ ($|\gamma|$ is equal to $\delta^6$ in [8], while $\delta^2$ or even $\delta$ is sufficient in our case). Let $Id_p$ be a shared variable that belongs to the domain $\gamma$; variable $Id_p$ is the *name* of node $p$. Another variable is used to collect the names of neighboring nodes: $Cids_p = \{ \bowtie Id_q \mid q \in N_p \}$, where $\bowtie Id_q$ refers to the cache copy of the shared variable $Id_q$ at node $p$. Let $\mathsf{random}(S)$ choose with uniform probability some element of set $S$. Node $p$ uses the following function to compute $Id_p$:

$$\mathsf{newId}(Id_p) = \begin{cases} \boxtimes\, Id_p & \text{if } \boxtimes\, Id_p \notin Cids_p \\ \mathsf{random}(\gamma \setminus Cids_p) & \text{otherwise} \end{cases}$$

The algorithm for constant height DAG construction is the following:

$$\mathsf{N1}: \quad true \;\rightarrow\; Id_p := \mathsf{newId}(Id_p)$$

**Theorem 1** *Algorithm N1 self-stabilizes with probability* 1 *in an expected constant time to a DAG which height is at most* $|\gamma| + 1$.

The proof of this theorem is similar to the one in [8]. There are two competing motivations for tuning the parameter $\gamma$. On one hand, a large value of $|\gamma|$ decreases the expected convergence time of N1. On the other hand, a small value of $|\gamma|$ decreases the DAG's height, and thus the expected convergence time of subsequent algorithms.

### 4.2 Density-driven Clusters 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 $(d_p = d_q) \wedge (Id_q < Id_p)$. 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 $\boxtimes Id_p = Id_p$ and $\boxtimes d_p = d_p$).

We now define the clusterHead choice function:

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

The cluster-head algorithm runs as follows:

$$\mathsf{R1}: \quad true \;\rightarrow\; d_p := \mathsf{density}$$
$$\mathsf{R2}: \quad true \;\rightarrow\; \mathcal{H}(p) := \mathsf{clusterHead}$$

**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 identifiers are bounded by a constant $\gamma$. The number of edges in the neighborhood at distance one is bounded by $\delta^2$, the number of neighbors at distance one is bounded by $\delta$, so the number of possible values for the density function is at most $\delta^3$. Overall, the name-space of values in the DAG$_\prec$ is $\gamma\delta^3$, which is bounded by a constant. 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 DAG$_\prec$, and the height of the DAG$_\prec$ is constant, so the expected time for stabilization is also constant. □

### 4.3 Improving Stability

We improve the stability of the algorithm by adding some selection criterion. 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 Id (as defined in Section 4). This scheme adds stability into clusters organization by limiting clusters 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 $(d_p = d_q) \wedge (\mathcal{H}(q) = Id_q) \wedge (\mathcal{H}(p) \neq Id_p)$ or $(d_p = d_q) \wedge (\mathcal{H}(p) \neq Id_p) \wedge (\mathcal{H}(q) \neq Id_q) \wedge (Id_q < Id_p)$. In addition, the height of the new 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:

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

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.

## 5 Simulations

As mentioned in Section 4, we suppose that there exists a constant $\tau > 0$ such that the probability of a frame transmission without collision is at least $\tau$. Yet, we can suppose that in a bounded time $\Delta(\tau)$, each node is able to locally broadcast one frame and then receive all packets sent by its 1-neighbors. Such a $\Delta(\tau)$ time unit is called 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 evaluate the performance of the proposed heuristic and estimate the importance of the introduction of the DAG. Nodes are randomly deployed using a Poisson process with different intensity levels $\lambda$ ($\lambda$ corresponds to the mean number of nodes per surface unit) in a $1 \times 1$ square with various transmission ranges $R$ varying from $0.05$ to $0.1$. Each given statistic is the average over 1000 simulations. All these results are fully described in [12].

To build the DAG, each node randomly chooses a DAG Id between $0$ and $\delta^2$ where $\delta$ is the maximum node's degree in the graph as defined in Section 3. For this, each node randomly chooses a DAG Id and then compares it to its neighbors'ones. If DAG Ids are the same, the node with the smallest "normal" Id chooses another DAG Id and so on until every node has a different DAG Id than the ones of its 1-neighbors. For simulations on a grid and on a random geometry topology with $\lambda$ equal to $1000$, the number of steps required to build the DAG does not take a lot of time since it only requires two steps on average, whatever $R$ is. Therefore, building the DAG is not costly.

We measure the following criterion: number of cluster-heads per surface unit, clustering tree length (also in order to evaluate time of stabilization as they are proportional) and cluster-head eccentricity. Table 1 shows these criterion for $\lambda = 1000$ and different values of $R$. (Note that results are given f or $\lambda = 1000$ but simulations have shown that the algorithm behavior the same way for different values of $\lambda$.) We note $e(\mathcal{H}(u)/\mathcal{C}) = \max_{v \in \mathcal{C}(u)}(d(\mathcal{H}(u), v))$ 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's degree), we can note that the mean cluster-head eccentricity and tree length 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's 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 nodes distribution, a node uses very rarely the Id to choose its parent because density values are well-distributed and scarcely 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 will have the same value density and the only criteria to select a cluster-head is the Id. As the nodes' Ids are not well distributed, all nodes will finally join the same head. Table 2 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 1 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 1(a) DAG is implemented and several clusters are created. On Figure 1(b), the DAG is not implemented and only one cluster is created.

We have also tested the extra criterion given in Section 4.3 to improve the stability. Due to space limitation, we will not present all the results. We performed simulations where nodes move randomly at a randomly chosen speed during 15 minutes. We computed the percentage of cluster-heads which remained cluster-heads after each 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) this percentage is 31 % with the new rules against 25 % without. Thus, our improvements are useful in terms of cluster-head stability.

| | $R = 0.05$ | | $R = 0.08$ | | $R = 0.1$ | |
|---|---|---|---|---|---|---|
| | 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 |
| $\tilde{e}(\mathcal{H}(u)/\mathcal{C}(u))$ | 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 1. Clusters features on a random geometric graph for $\lambda = 1000$.**

| | $R = 0.05$ | | $R = 0.08$ | | $R = 0.1$ | |
|---|---|---|---|---|---|---|
| | 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 |
| $\tilde{e}(\mathcal{H}(u)/\mathcal{C}(u))$ | 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 2. Clusters characteristics on a grid for $\lambda = 1000$.**

## 6 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 [11] 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 ([5, 13]).

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. 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 *IEEE INFOCOM*, 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] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1987.

[4] 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.

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

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

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

[8] T. Herman and S. Tixeuil. A distributed tdma slot assignment algorithm for wireless sensor networks. In *Proceedings 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.

[9] 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.

[10] H.-C. Lin and Y.-H. Chu. A clustering technique for large multihop mobile wireless networks. In *IEEE VTC*, may 2000.

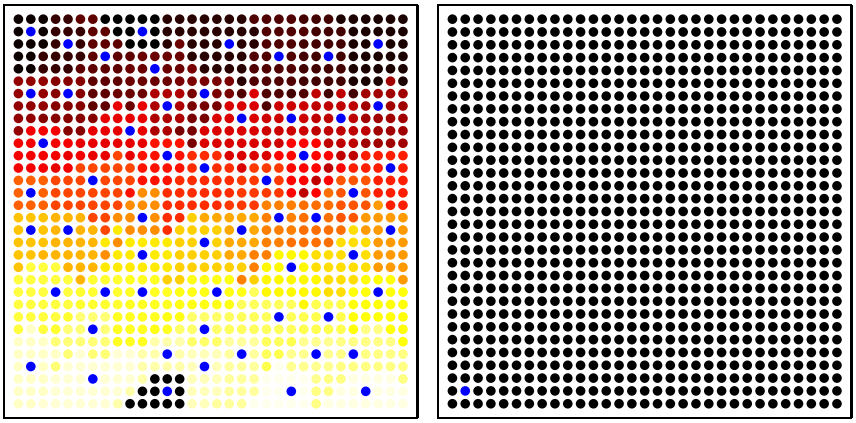
[11] N. Mitton, A. Busson, and E. Fleury. Self-organization in large scale ad hoc networks. In *3nd Med-Hoc-Net*, june 2004.

[12] N. Mitton, E. Fleury, I. Guerin-Lassous, and S. Tixeuil. Self-stabilization in self-organized multi-hops wireless networks. Research Report RR-5426, INRIA, December 2004.

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

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

[15] 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.



(a) With DAG (b) No DAG

**Figure 1. Clustering example in grid with $R = 0.05$.**