About the Self-Stabilization of a Virtual Topology for Self-Organization in Ad Hoc Networks

Fabrice Theoleyre, Fabrice Valois

About the self-stabilization of a virtual topology for self-organization in ad hoc networks

Fabrice Theoleyre and Fabrice Valois

CITI, INRIA ARES, INSA Lyon
21 av Jean Capelle, 69621 Villeurbanne Cedex, France
{fabrice.theoleyre,fabrice.valois}@insa-lyon.fr

Abstract. Ad hoc networks are spontaneous wireless networks without any wired infrastructure, composed of mobile terminals. We assume that nodes must collaborate to set up an efficient network, such a collaboration requiring a self-organization in the network. We proposed a virtual structure to organize the network: the backbone is a connected structure helping to optimize the control traffic flooding. Clusters form services area, hierarchizing the network, electing one leader per cluster. Since the ad hoc topology is volatile, the self-stabilization of the algorithms is vital. The algorithms for both the construction and the maintenance are analytically studied to prove the self-stabilization of the proposed self-organization. Thus, the virtual structure is efficient and very scalable, a local topology change impacting only locally the virtual structure. Finally, simulations investigate the behavior and the performances of the virtual structure.

1 Introduction

MANet (Mobile Ad hoc NETworks) are spontaneous topologies of mobile nodes where each of them collaborate in order to give services like routing, localization, etc. It can be used to offer a spontaneous network infrastructure. Each terminal can communicate via wireless links without preconditioned fixed infrastructure. The network must function autonomously, without any human intervention. To send packets from a source to a destination, either the destination is in the radio range of the source or intermediaries nodes must help to forward the packets. To reach such a goal, the nodes must collaborate and exchange control information to set up routes in the network. Indeed, each node is both client and router. Because of the nodes mobility, radio links are created and deleted continuously leading to topology changes. And finally, routes are volatile. So, self-adaptation of the network to the dynamicity is a major issue of MANet. Ad hoc networks can be connected to the Internet, via a dedicated device, the wireless access point (AP), gateway from the ad hoc network to the wired world. Such networks are often called *hybrid networks* constituting *wireless multihops cellular networks*.

In our point of view, self-organization can answer to the above key problems. Self-organization deals with virtual topologies in order to simplify ad hoc topologies. For example, virtual topologies can be constituted by a backbone

[10], or a combination of a backbone and clusters [8]. The goal is to offer control on the MANet. According to us, virtual topologies allow scalability (in creating a hierarchy useful for example for routing protocols), facilitate integration of MANET in wired or cellular networks (with a virtual backbone), hide topology changes (in creating a stable macroscopic view), take heterogeneity into account (in distributing unfairly the load in the network). The mobility represents a key challenge in MANet. As each node is mobile, many radio links appear and disappear, occurring many topology changes. Virtual structures must remain efficient along the time. Hence, it must be continuously maintained, such that structural constraints hold. The structure must reconstruct or repair itself with a minimal delay. Such a property conduct to the self-stabilization properties.

In this paper, we focus on the demonstration of self-stabilization properties of the virtual structure described in [8]. This article makes two main contributions to the understanding of ad hoc self-organized virtual structures. First, it proves theoretically self-stabilized properties of the virtual structure. Secondly, it proposes an evaluation of convergence time of the algorithms through simulations.

Next, we will expose related work about self-organized virtual structures in ad hoc networks. Section 3 presents the distributed algorithm of the studied virtual structure. Section 4 presents the notations and complexity results. Section 5 presents an analytical study of the self-stabilized properties for the backbone, and section 6 is dedicated to the clusters. Results of simulations are given in section 7. Finally, we conclude this work and give some perspectives.

2 Related work

Clusters Clustering consists in grouping nodes geographically close. A clusterhead is often elected per cluster, managing its services area. Each node must be $k_{cluster}$ hops far at most from its clusterhead. Let $N_k(u)$ be the k-neighborhood of u, i.e. the set of nodes at most k hops far from u.

[6] is the most used algorithm to construct clusters. In the first step, each node initiates a neighborhood discovering. According to this information, each node decides to become clusterhead or not. The decision is propagated in the neighborhood so that each node which has not chosen any clusterhead yet takes the source as its new clusterhead. The decision could be based on several metrics (node identifier (id), mobility, location...). The authors propose to reconstruct the cluster if the diameter constraint of 3 hops is violated.

Backbones A backbone could be well modeled with a *Minimum Connected Dominating Set* (MCDS): each node must be neighbor of at least one node of the MCDS which is a connected structure with a minimal cardinality. The construction of an MCDS is a NP-hard problem.

Many articles propose to construct a CDS in 2 steps. First, a dominating set (DS) is created, where each node is neighbor of a node in the DS. Secondly, the DS is interconnected to form a connected structure. Usually, 4 nodes

states exist: dominator(in the CDS)/dominatee(not in the CDS)/active(in election)/idle(waits for the construction). In [2, 3, 1], a leader declares itself dominator and broadcasts its decision: its neighbors become its dominatees. The neighbor of dominatees become active. The active nodes with the highest weight in their neighborhood become dominators, and the process keeps on. Then, the DS must be interconnected. [3] proposes an iterative exploration requiring an important overhead and delay. [1] proposes a best effort approach, sending `invite` packets.

To the best of our knowledge, only [10] proposes a localized algorithm. A node is elected as a CDS member if it has 2 disconnected neighbors. Rules for a redundancy elimination are proposed: *a node with a set of 2 connected neighbors of higher id which cover its whole neighborhood becomes dominatee, else it becomes dominator*. This rule could be extended as: *a node with a set of neighbors of higher id forming a CDS and covering its whole neighborhood becomes dominatee, else it becomes dominator*. These rules create a CDS.

Self-Stabilization Self stabilization was first defined by Dijkstra [4]: a system is self-stabilizing when "regardless of its initial state, it is guaranteed to arrive at a legitimate state in a finite number of steps." [7] presents bases of the self-stabilization in the fault tolerance domain. In ad hoc networks, topology changes occur frequently, and could be modeled as temporary faults. In consequence, the self-stabilization properties of an algorithm are essential in the ad hoc networks. Recently, [5] studied a multicast protocol in ad hoc networks. To the best of our knowledge, no prior work was done to study the self-stabilization properties of the Connected Dominating Sets structures in ad hoc networks.

3 The Virtual Topology for Self-Organization

We proposed in [8] a virtual topology for self-organization. This topology helps to structure the network, to optimize floodings, to create a hierarchy... It is constituted by a backbone and clusters. First, a k-neighborhood discovering is initiated. Then, the algorithm constructs a k_{cds}-CDS. Finally, some dominators are elected as clusterheads such than $k_{cluster}$-clusters are formed.

3.1 Backbone

Construction The following nodes states exist: dominator / dominatee / active (in election) / idle (initial state). The Access Point (AP) acts as leader and becomes the first dominator. It propagates its new state k_{cds} hops far using an `hello` packet. The following rules are applied when a node receives an `hello` to construct a k_{cds}-Dominating Set:

1. An active or idle node which receives an `hello` from a dominator D, k_{cds} hops far, becomes dominatee and chooses D as parent

2. An idle node which receives an `hello` from a dominatee D, k_{cds} hops far, becomes active and triggers a timer of $\Delta_{election}$ seconds. $\Delta_{election}$ is the maximal round-trip-time to a farthest k_{cds}-neighbor.
3. After the timer expiration, a node which owns the highest weight among its active k_{cds}-neighbors becomes dominator. We can remark that a dominator has no parent during this phase.

The interconnection is inspired from [1]: the leader sends a `cds-invite`, with a TTL $2 \cdot k_{cds} + 1$. A dominator without parent chooses the source as new parent and sends a `cds-join` along the inverse route. Each intermediary dominatee becomes dominator and sets its parent as the next hop in the route. A dominator which sent a `cds-join` can send a `cds-invite` for other dominators in its $(2k_{cds}+1)$-neighborhood. The dominators form finally a k_{cds}-CDS structure.

Maintenance A node sends periodically `hellos` containing its id, weight, cds-state, parent in the CDS and ids of its 1-neighbors. `hellos` being forwarded k_{cds} hops along, each node has a complete knowledge of its k_{cds}-neighborhood. Hence, each dominatee can verify that its parent is still valid: it is at most k_{cds} hops far, is dominator, and there exists a dominatee neighbor having the same parent and being nearer of this parent (to force connectivity of the cds-dominance area).

The backbone must remain connected. Hence, the AP sends periodically `ap-hellos`, forwarded only by dominators. If a node misses several `ap-hellos` from its parent in the backbone, it considers itself disconnected and engages a backbone reconnection. It sends a `cds-request` in broadcast with a TTL of $2k_{cds}+1$. At least one connected dominator is at most $2k_{cds}+1$ hops far. It will reply with a `cds-reply` following the inverse route. Finally, the disconnected dominator sends a `cds-accept` to force intermediaries to become dominators.

To avoid a constant growth in the size of the backbone, we propose a mechanism to eliminate redundancy. A dominator is *useless* if it has no dominatee at exactly k_{cds} hops and no dominator for which it is a parent. An useless dominators sends a `useless-advertisement` forcing all its children in the backbone to choose its parent as new parent.

If many reconnections occur in the backbone, the load on the radio medium could be important. Hence, many collisions occur, disturbing the reconnection process. A dominator which tries many unsuccessful `cds-reconnect` sends a `break` in broadcast and takes the *idle* state. When a node receives a `break` from its parent, it becomes *idle* and forwards the message. Finally, the whole branch becomes *idle*. A connected dominator neighbor of the idle area will trigger the reconstruction, acting like the construction.

3.2 Clusters

Construction As the backbone was constructed during the first phase, we use naturally it for the cluster construction. Only dominators participate to the election, reducing the overhead. Moreover, a clusterhead is forced to be dominator: a clusterhead will use further the backbone to optimize the floodings.

During the construction, each dominator begins to send periodically `cluster-hellos` when all its neighborhood has either the dominator or the dominatee state. `cluster-hellos` contain the address of the source and its weight. Theses packets are forwarded $k_{cluster} - k_{cds}$ hops along, uniquely by virtual neighbors. A virtual neighbor of N is either a parent of N in the CDS, or a child (a node for which N is a parent). A `cluster-hello` is forwarded only if it comes from a parent or a child in the CDS. A node is elected clusterhead if it has the highest weight among all its $k_{cluster} - k_{cds}$-virtual neighbors without clusterhead. An elected clusterhead sends a gratuitous `cluster-hello` to advertise its decision. A dominator without clusterhead chooses the source of the `cluster-hello` as clusterhead if the previous hop has also chosen this clusterhead, and if the clusterhead is at most $k_{cluster} - k_{cds}$ hops far. Such a condition forces the construction of connected clusters. Since dominatees are at most k_{cds} hops far from their parent, the algorithm constructs clusters of radius $k_{cluster}$.

Maintenance `cluster-hellos` are not yet required for the maintenance. However, each node adds in its `hellos` its clusterhead, the relay and the distance toward it. Hence, each dominator can easily verify that its clusterhead is valid, i.e. a virtual neighbor has the same clusterhead and is at most $k_{cluster} - k_{cds} - 1$ hops far from its clusterhead.

If a node A loses its clusterhead C_{old}, i.e. C_{old} is no more valid, it searches a new candidate: a node is a virtual neighbors and announces a clusterhead at most $k_{cluster} - k_{cds} - 1$ hops far. When a node changes its clusterhead, it sends immediately a gratuitous `hello` to force other nodes to change potentially their own clusterhead. In this way, the convergence delay is reduced.

We propose a procedure to eliminate redundancy. If a clusterhead has no virtual neighbor having chosen it as clusterhead, the node is an useless clusterhead. Since a cluster is connected, no other node has a fortiori chosen it as clusterhead. A useless clusterhead tries to find a new valid clusterhead and become client.

4 Preliminaries

To study the ad hoc networks, we use the graph theory: a node in the network is represented by a vertex, and there exists one edge from one vertex to another iif there exists a radio link between the two nodes. Since we use only bidirectional links, we study undirected graphs. We note G(V,E) the graph, V being the set of vertices and E the set of edges. We assume that the graph is connected. We use the following notations:

- n: the cardinality of the network ($= |V|$)
- D: the set of dominators: $|D|$ is the CDS cardinality
- $N_k(u)$: the k-neighborhood of u
- $\Delta_k(u)$: the number of k-neighbors ($\Delta_k(u) = |N_k(u)|$), i.e. the number of nodes at most k hops far. By convention, $\Delta_1(u) = \Delta(u)$
- $\Delta'_k(u)$: the number of k-virtual-neighbors. A virtual neighbor of N is either the parent or a child of N the CDS. We can remark that $\Delta'_k(u) \leq \Delta_k(u)$

- $w(u)$: the weight of the node u
- $d(u, v)$: the distance in hops from u to v
- h_T: the maximal distance from one node to the root of T (the *height* of T)
- $dominator(u)$: is the parent of a dominatee u. $dominator(u) \in N_{k_{cds}}(u)$
- $parent(u)$: is the parent of a dominator u. $parent(u) \in N(u)$

5 Backbone Self-Stabilization

Ad hoc networks presenting a volatile topology, the virtual structure must adapt itself to changes. We present here and in the following section results about self-stabilization of the virtual structure presented in section 3. The construction algorithms converge in a finite time. In the same way, the maintenance algorithms form a valid virtual structure if the number of topology changes (edge/vertex addition or deletion) is finite and sufficiently inter-spaced. We assume that the graph associated to the ad hoc network is connected. If during the construction, not enough time is sufficient to let the structure converge because of unknown reason, the algorithm will converge during the maintenance step. More details are given in the long version of this article[9].

Hypothesis 1 *We assume that the radio topology is stable after a list of changes, constituted by a sum of elementary topology change (vertex/edge deletion/addition). The inter-changes time is sufficient to let the algorithm converge.*

We propose here to demonstrate that the construction algorithm provides a k_{cds}-Connected Dominating Set (CDS). We prove first that the backbone forms a k_{cds}-Dominating Set (DS), then a connected structure, being moreover a tree. Same proofs are given for the maintenance.

5.1 Construction

Creation of a k_{cds}-Dominating Set

Theorem 1. *The algorithm of the first phase terminates and forms a k_{cds}-DS*

Lemma 1. *Every vertex has either the dominator or the dominatee state at the end of the first step.*

Proof. Let separate the problem in 2 cases:

• Let assume that an idle vertex I exists, and that there exists another not-idle vertex N in the connected component including N. Let $c = \langle I, c_1, c_2, ..., c_k, N\rangle$ be a path from I to N. All the k_{cds}-neighbors of I are idle, else I would have change its state. Thus, $\{c_j\}_{j\in[1..k_{cds}]}$ are idle. In the same way, the recurrence formula is: $\forall i, \{c_{i\cdot k_{cds}+j}\}_{j\in[1..k_{cds}]}$ *idle* $\Rightarrow \{c_{(i+1)\cdot k_{cds}+j}\}_{j\in[1..k_{cds}]}$ *idle*. In consequence, N must be idle. The connected component is only constituted by idle vertices. However, at least the leader is not idle. This leads to a contradiction.

• Let assume that a vertex N is active. If a k_{cds}-neighbor is dominator, N would be dominatee. In the same way, if all the k_{cds}-neighbors are dominatees,

N would be dominator. If N is the active node of highest weight in its k_{cds}-neighborhood, then N is elected dominator after $\Delta_{election}$ time at most. So, there exists A_1, active, at most k_{cds} hops far and with an higher weight than N.

Let A_k be the graph so that its vertices are the active vertices of G during the k^{th} round, and so that there exists an edge from a vertex a_i to a vertex a_j if and only if $w(a_i) < w(a_j)$. A_k is acyclic and has a finite cardinality, inferior or equal to n. The second property is trivial, let demonstrate the first property. Let $c = \langle c_0, c_1, ..., c_k \rangle$ be a cycle in A_k. An edge exists from c_i to c_{i+1}, i.e. $w(c_i) < w(c_{i+1})$ with $i \in [1..k-1]$. Transitively, $w(c_0) < w(c_k)$. However, c is a cycle: the edge (c_k,c_0) exists and $w(c_k) < w(c_0)$, this leads to a contradiction.

The graph A_k contains at least a sink a_k, i.e. a vertex has a null outer degree. After $\Delta_{election}$ seconds, a_k will be elected and become dominator, its k_{cds}-neighbors becoming its dominatees. Let I_k be the set of idle vertices in G during the k^{th} round. During the round k, at least one vertex a_k becomes dominator. So, $a_k \notin A_{k+1} \cup I_{k+1}$. The k_{cds}-neighbors of a_k in $A_k \cup I_k$ become its dominatees. Simultaneously, some vertices are extracted from I_k and added to A_{k+1}. So $|I_k| + |A_k| \geq |I_{k+1}| + |A_{k+1}| + |\{a_k\}|$. In consequence: $|A_n| = |I_n| = 0$. In consequence, the algorithm will converge at the end of the first phase to a graph with no active vertex.

Lemma 2. *Every vertex is at most k_{cds} hops far from a dominator, or is itself a dominator, i.e. the graph of dominators forms a k_{cds}-DS.*

Proof. The proof comes directly from the lemma 1: at the end of the first phase, only dominatees and dominators exist: a dominatee changes its state because a dominator is at most k_{cds} hops far (by construction) and a vertex elected dominator remains dominator.

Formation of a k_{cds}-CDS

Theorem 2. *The set of dominators forms at the end of the construction a connected set of k_{cds}-dominating, i.e. a k_{cds}-CDS.*

Property 1. Let c be a path between 2 dominators D_1 and D_k. c follows the property 1 if it is composed by a set of i dominators, interspaced consecutively from each other by at most $2 \cdot k_{cds}$ dominatees: $\exists c = \langle D_1, d_1, ..., d_j, D_2, d_{j+1}, ..., D_i \rangle$ such that d_l are dominatees, and such that $d_c(D_i, D_{i+1}) \leq 2 \cdot k_{cds} + 1$.

Lemma 3. *A path c exists at the end of the first phase of the algorithm which follows the property 1, binding each dominator to the leader $\mathcal{L}$.*

Proof. Let D_k be the set of dominators elected during or before the k round. $D_0 = \{\mathcal{L}\}$. D_0 comprises only one dominator following trivially the property 1.

Let assume that D_k follows the property 1. At the end of the $k-1^{th}$ round, a set S_{k-1} of vertices was elected dominators, such that $S_{k-1} \cup D_{k-1} = D_k$ and $S_{k-1} \cap D_{k-1} = \emptyset$. A node N of S_{k-1} is active during the k-1^{th} round before being elected at the end of the round. Let $c_1 =< N, a_1, ...a_i, d >$ be the path

from N to the nearest dominatee d during the round $k-1$. N being active, by construction, $|c_1| \leq k_{cds}+1$. The $\{a_l\}$ are by definition not dominatees, and are by construction at most k_{cds} hops far from d, a dominatee. In consequence, $\{a_l\}$ are active. Since N will be elected dominator, $\{a_l\}$ will become its dominatees at the end of the round. Let $c_2 =< d, d_1, ...d_i, D >$ be the path from the dominatee d to its parent D. By definition, $D \in D_k$, $|c_2| \leq k_{cds}$, and d_l are dominatees. Since $D \in D_k$, let $c_3 =< D, ..., \mathcal{L} >$ be the path from D to the leader. c_3 follows the property 1. Clearly, the path concatenation $c_1.c_2.c_3$ follows the property 1 at the end of the first phase of the algorithm.

Lemma 4. *If the property 1 is respected at the end of the first phase, the algorithm will construct a connected k_{cds}-DS.*

Proof. Let $\mathcal{D}_i$ be the set of dominators such that for each dominator D from $\mathcal{D}_i$, the path c from D to the leader, following the property 1 has at most i dominators. $\mathcal{D}_0 = \{\mathcal{L}\}$. $\mathcal{D}_0$ forms a trivial connected k_{cds}-DS applied to the vertices dominated by $\mathcal{D}_0$. It will send, according to the construction algorithm, a `join-invite` with a TTL=$2 \cdot k_{cds} + 1$.

Let assume that the set $\mathcal{D}_i$ forms a connected k_{cds}-DS. Let a dominator $u \in \mathcal{D}_{i+1}$, and c be the path from u to the leader $\mathcal{L}$, respecting the property 1. $c = \langle u, v_1, ..., v_k, \mathcal{L} \rangle$. From the lemma 3, there exists a dominator v_i from c, at most $2k_{cds}+1$ hops far from u since c respects the property 1. v_i has a path $c' \subset c$ respecting the property 1. Moreover, $v_i \in \mathcal{D}_i$. Thus, v_i will send a `join-invite` with a TTL=$2k_{cds}+1$. u will receive the `join-invite`, and will connect itself to $\mathcal{D}_i$. In consequence, $\mathcal{D}_{i+1}$ forms a connected k_{cds}-DS.

Formation of a tree

Definition 1. *Let the CDS $\mathcal{G}_{CDS}$ containing all the vertices of G, and such that an edge exists from a vertex u to a vertex v iif v is the parent of u if u is a dominator, or iif v is the relay toward its dominator if u is a dominatee.*

Theorem 3. *$\mathcal{G}_{CDS}$ is a tree.*

Proof. According to the previous definition of $\mathcal{D}_i$, $\mathcal{D}_0 = \{\mathcal{L}\}$ is a trivial tree, formed by a singleton. Let assume that $\mathcal{D}_i$ forms a tree. $\mathcal{D}_i$ has $|\mathcal{D}_i - 1|$ edges. Let $u \in \mathcal{D}_{i+1}/\mathcal{D}_i$. u will interconnect itself to the CDS thanks to a `join-invite` sent by a dominator from $\mathcal{D}_i$. Let v be this dominator. The path $c = \langle u, u_1, ..., u_k, v \rangle$ has only dominatees, else u choosing the nearest dominator, will not interconnect itself to v. In consequence, dominatees will become dominators. We add to $\mathcal{D}_i$ a branch of k dominatees and one dominator, with k edges from a dominatee to its new parent, and an edge from u to its new parent. Thus, $\mathcal{D}_i \cup \{u\} \cup \{u_i\}_{i \in [1..k]}$ has $|\mathcal{D}_i| - 1 + 1 + k$ edges, i.e. $\left|\mathcal{D}_i \cup \{u\} \cup \{u_i\}_{i \in [1..k]}\right| - 1$ edges. In consequence, $\mathcal{D}_{i+1}$ is a tree.

Let d_i the set of dominatees at at most i hops from their father. When a vertex d_0 to $\mathcal{D}$, the vertex and the edge toward its parent is added. Then, $d_0 \cup \mathcal{D}$ remains a tree.

Let $d_i \cup \mathcal{D}$ be a tree. Let $u \in d_{i+1}$ be a dominatee. u chooses a parent and a relay r toward this parent. r is one hop nearer from its parent, by construction. Thus, $r \in d_i$. Only one vertex and one edge are added. $d_i \cup \mathcal{D}$ is a tree. A dominatee being at most k_{cds} hops far from its dominator, $\bigcup_{i \in [1..k_{cds}]} d_i \cup \mathcal{D} = G$. In conclusion, the CDS forms a tree.

5.2 Maintenance

Dominating Set

Theorem 4. *A dominatee has always a dominator, at most k_{cds} hops far, i.e. the CDS forms a k_{cds}-DS.*

Proof. Dominatees with a dominator neighbor choose it as parent. This dominator is valid. Let assume that the set of dominatees at most i hops far from their parent have a valid parent. A dominatee at most $i+1$ hops far from its parent has chosen it since it is at most k_{cds} hops far, through another dominatee having chosen the same dominator, but at i hops, with $i < k_{cds}$. Thus, since the parent of dominatees at most i hops far from their parent is valid, each dominatee chooses a valid parent.

A dominatee can have no dominator candidate for reconnection in its neighborhood table, i.e. no neighbor exists having chosen a dominator at most k_{cds}-1 hops far. Such a dominatee becomes active. An active vertex becomes dominatee iif it finds a valid dominator as parent. Active vertices becoming dominators execute the maintenance reserved for dominators. Thus, each dominatee has a dominator at most k_{cds} hops far, and this dominator is reachable through a dominatee with the same dominator, one hop nearer from its parent.

Connectivity

Theorem 5. *The set of dominators forms a tree.*

Lemma 5. *The set of dominators remains a (connected) tree when the radio topology is stable.*

Proof. Let assume that the topology is stable. Each dominator receives an `ap-hello`, maintaining the source as parent. Let $\mathcal{D}_i$ the set of dominators, i hops far via other dominators from the leader, the root of the CDS. $\mathcal{D}_i$ is supposed connected. The vertices of $\mathcal{D}_{i+1}/\mathcal{D}_i$ choose a parent in $\mathcal{D}_i$ since they receive the `ap-hello` from their parent, and so they are one hop farther from the leader. Thus, $\mathcal{D}_{i+1}$ is connected.

Let assume $\mathcal{D}_i$ has no cycle, E_i be the set of edges of $\mathcal{D}_i$, and V_i be the set of its vertices.We can establish that $|E_i| = |V_i - 1|$. For each vertex of $\mathcal{D}_{i+1}/\mathcal{D}_i$, we add one vertex in E_i and one edge in V_i. So :

$$|E_{i+1}| = |V_i| - 1 + [|V_{i+1}| - |V_i|] = |V_{i+1}| - 1$$

Thus $\mathcal{D}_{i+1}$ is connected, without any cycle.

Definition 2. *We consider a dominator* u connected *iif there exists an ascendant path directed from* u *to the leader* $\mathcal{L}$*, where the first edge is* $(u, parent(u))$*, and then constituted by the ascendant path* $\langle parent(u), ..., \mathcal{L}\rangle$*.*

Lemma 6. *When a dominator of a branch of the CDS reconnects itself, all its ascendants and descendants reconnect themself.*

Proof. If a dominator u reconnects itself, then there exists a valid path $\langle u, ..., \mathcal{L}\rangle$ to the leader. Besides, a descendant or an ascendant v of u has by definition a path $\langle u, ..., v\rangle$. Thus, v has a path $\langle u, ..., v\rangle \cup \langle u, ..., \mathcal{L}\rangle$ to the leader. However, all dominators must perhaps change their parent to have a valid path to the leader.

Lemma 7. *When all dominator of a branch are disconnected, at least one dominator will reconnect itself.*

Proof. Every topology change could be decomposed by an elementary addition/deletion of edges. The addition of an edge in the graph cannot generate a disconnection in the CDS. Let assume that the edge (u, x) was deleted. After a finite time Δ_t, the whole branch, i.e. the descendants of u, will consider itself disconnected. A dominator considers itself disconnected when it missed all `ap-hellos` during Δ_t. Δ_t depends from the interval between two `ap-hellos` and the number of acceptable missed `ap-hellos`. Let v be a dominator descendant of u. u will not forward any `ap-hello` with an id superior to l, id of the last `ap-hello` forwarded before the edge (u, x) broke. Thus, the child of u cannot forward any `ap-hello` with an id superior to l. Recursively, v can neither receive nor forward any `ap-hello`. The dominators of the branch of root u consider themself disconnected, and try to reconnect themself via a dominator forwarding an `ap-hello` with an id superior to l.

At least one dominator finalizes its reconnection, and no cycle is created in the CDS, i.e. v cannot choose to reconnect itself to a descendant of u. Effectively, v asks for an `ap-hello` id higher than the last `ap-hello` forwarded by any descendant of u, as explained above. Let $\mathcal{D}$ be the set of descendant dominators of u, and their dominatees (the disconnected part). $\mathcal{D}$ is a connected component. Let $C = G/\mathcal{D}$. C is also a connected component: let $c \in L$ be a descendant of $\mathcal{L}$. $c \in C$ is by definition not descendant of u. Thus $u \notin \langle c, ..., \mathcal{L}\rangle$, in other words $\langle u, x\rangle$ and $\langle c, ..., \mathcal{L}\rangle$ are disjoint.

Let $\mathcal{N}$ be the set of vertices in C, neighbors of $\mathcal{A}$. $\mathcal{N} \neq \emptyset$: let $u \in \mathcal{D}$. The graph is assumed connected. Thus, a path $p = \langle u, u_1, ..., \mathcal{L}\rangle$ exists with $u \in \mathcal{D}$ and $\mathcal{L} \in C$. $u_i \in p$ exists such that $u_i \in C$ and $u_{i-1} \in \mathcal{D} \cap p$. By definition of $\mathcal{N}$, $u_i \in \mathcal{N}$. Moreover, $dominator(u_{i-1})$ is a dominator of the disconnected branch since u_{i-1} is in $\mathcal{D}$. $dominator(u_i)$ is connected, and is in C. $d\,(dominant(u_{i-1}), dominator(u_i)) \leq 2 \cdot k_{cds} + 1$. Thus, a dominator at most $2k_{cds}+1$ hops far from a connected dominator exists in the disconnected branch.

Finally, $dominator(u_{i-1})$ will reconnect itself to $dominator(u_i)$ thanks to a `cds-reconnect` with a TTL=$2k_{cds} + 1$. According to the lemma 6, each dominator of $\mathcal{D}$ will reconnect itself and choose a new valid parent with an `ap-hello`.

In consequence, we can conclude:

Theorem 6. *When an edge deletion implicates a disconnection in the CDS, the CDS will reconstruct itself and a valid CDS will be created.*

Lemma 8. *If a break of the CDS occurs, the branch is broken and then rebuilt.*

Proof. The idle zone is a connected component of the graph. Since we consider the events as discrete, the set of not idle nodes forms also a connected component, comprising the leader $\mathcal{L}$. Let u be a vertex not idle, neighbor of the idle area. Such a vertex exists for the same reason as the lemma 7. Let i be in the idle zone, and neighbor of u. Two cases exist. If u is a dominator, it will reconnect itself in a finite time according to the theorem 6. If u is a connected dominator, it will send a `cds-invite` with a TTL=$k_{cds}+1$. Clearly, i will receive this packet. If u is a dominatee, $dominator(u)$ will be in a finite time a connected dominator for the same reason as above. For the same reason as above, $dominator(u)$ will send a `cds-invite` with a TTL=$k_{cds}+1$ and i will receive it.

A `cds-invite` received by the idle node i triggers the reconstruction of the idle branch. i becomes the leader of the zone. The reconstruction leader i reconnects itself to $dominator(u)$ in sending a `cds-accept`. $dominator(u)$ being by definition connected itself to the leader, i is transitively connected. Following a proof similar to theorem 2, a CDS is reconstructed.

6 Cluster Self-Stabilization

6.1 Construction

Theorem 7. *The set of clusterheads constructs a $k_{cluster}$-dominating set, i.e. a $k_{cluster}$-clustering.*

Proof. If a dominator is not a clusterhead, then it chooses a dominator-clusterhead according to the process of neighborhood discovering on the CDS topology. `cluster hellos` are forwarded along the CDS-links, at most $k_{cluster}-k_{cds}$ hops far. So a dominator chooses a clusterhead at most $k_{cluster}-k_{cds}<k_{cluster}$ hops far. Moreover, a dominator chooses as clusterhead the source of a `cluster hellos` only if the previous hop chose also the source as clusterhead. Hence, the cluster is connected.

A dominatee has the same clusterhead as its dominator. Moreover, according to the lemma 2, it is at most k_{cds} hops far from its dominator, itself $k_{cluster}-k_{cds}$ hops far from its clusterhead. Transitively, a dominatee is at most $(k_{cluster}-k_{cds})+k_{cds}=k_{cluster}$ hops far from its clusterhead. Since a dominatee is connected to its dominator through a path containing at most k_{cds} dominatees having chosen the same dominator, the cluster is connected.

According to the lemma 1, each vertex is either dominator or dominatee. In consequence, any vertex has a clusterhead, at most $k_{cluster}$ hops far.

6.2 Maintenance

Theorem 8. *The maintenance algorithm maintains a set of clusterheads forming a $k_{cluster}$-dominating set of $\mathcal{G}_{CDS}$.*

Lemma 9. *Dominators are at most $k_{cluster}$ hops far from their clusterhead.*

Proof. Let $\mathcal{G}'_{CDS}$ be the set of dominators having chosen the vertex C as clusterhead. There exists one edge in $\mathcal{G}'_{CDS}$ from u to v if v is the relay toward the clusterhead for u. We can remark that such edges and vertices own to $\mathcal{G}_{CDS}$. If v is at most H hops far from its clusterhead, u is at most $H+1$ hops far from its own clusterhead in $\mathcal{G}'_{CDS}$ and also in $\mathcal{G}_{CDS}$.

$\mathcal{G}'_{CDS}$ is a tree, i.e. no cycle exists in $\mathcal{G}'_{CDS}$. Let assume the existence of a cycle $\langle u_1, ..., u_k \rangle$. u_i with $i \in [1..k]$ is H_i hops far from C. u_1 is the relay toward the clusterhead of u_k. So $H_k = H_1 + 1$. In the same way, u_{j+1} being the relay of u_j for $j \in [1..k-1]$, $H_j = H_{j+1} + 1 \Rightarrow H_j < H_{j+1}$. Thus, $H_k = H_1 + 1$ and $H_1 < H_k$, this leads to a contradiction. $\mathcal{G}'_{CDS}$ is a tree, a dominator chooses a relay one hop nearer of the clusterhead C.

Let D_i be the set of dominators which set the field *distance to clusterhead* to i in their `hellos`. Any dominator of D_i chooses by construction a relay in D_{i-1}. Let assume that the vertices in D_{i-1} are $i-1$ hops far from C. Thus, the vertices of D_i are i hops far from C. Moreover, $D_0 = C$ and C is 0 hops far from itself. Finally, a dominator is allowed to choose a relay only if this relay is at most $k_{cluster} - kcds$ hops far from its clusterhead. Thus, $D_{k_{cluster}-k_{cds}} = \emptyset$. A dominator has either a clusterhead at most $k_{cluster} - k_{cds}$ hops far, or becomes its own clusterhead.

Lemma 10. *Dominatees are at most $k_{cluster}$ hops far from their clusterhead.*

Proof. This result holds for the same reasons as in the theorem 7.

7 Performance evaluation

We simulate our solution with OPNET Modeler 8.1, using the WIFI standard model (300m radio range). The default parameters are 40 nodes and a degree of 10. The 95% confidence intervals are reported on the figures.

General performances Figure 1 presents the general performances of the CDS, without mobility. The cardinality is stable and scalable according to the number of nodes. The connectivity is not 100% since packet collisions may occur. However, it remains over 99.5%. Algorithms for both the CDS and the clusters seem present a good horizontal scalability, i.e. according to the cardinality.

Convergence of the construction algorithm We investigate the convergence time of the algorithm for the CDS construction. The clusters are always well-constructed before the end of the CDS construction. In consequence, the clusters are robust and don't represent the more sensitive part of the virtual structure. Thus, no

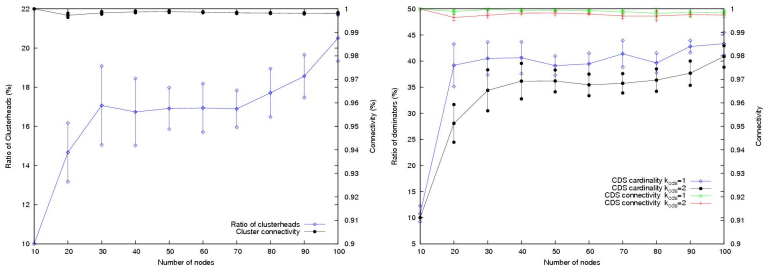

Fig. 1. Impact of the number of nodes

simulation result about the convergence of clusters are given here, the convergence being too fast.

Approximately 5 seconds are needed to have no idle node in the networks with k_{cds}=1 and $k_{cluster}$=2 (fig. 2). Two supplementary seconds are necessary for the election, i.e. no active node remains in the network. Finally, less than 10 seconds are necessary to have a CDS *largely connected* or *strictly connected*. *Strictly connected* means that the tree relation (node→parent) creates a valid Connected Dominating Set. For a *largely connected* CDS, we take into account the redundant mesh structure of the CDS (edges between each backbone neighbor and each dominatee with the same parent). The construction algorithms, executed in parallel for the first and second phases seem efficient: they converge quickly, forming in a few seconds an operational and self-organized ad hoc network. Results are little higher but similar for k_{cds}=2 and $k_{cluster}$=3.

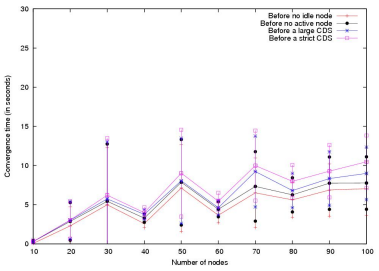

Fig. 2. Convergence Time for a CDS (k_{cds}=1 / $k_{cluster}$=2)

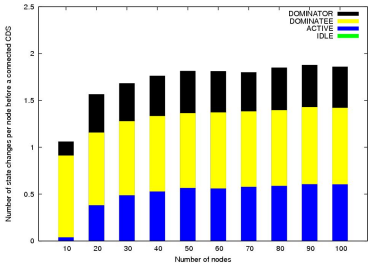

Fig. 3. Ratio of the number of cds state changes and the number of nodes before having a connected CDS (k_{cds}=1 / $k_{cluster}$=2)

In figure 3 is represented the number of state changes to have a valid CDS with k_{cds}=1 and $k_{cluster}$=2. More precisely, we measure the number of times a node changes its state before having a valid CDS. For example, with 100 nodes, 70% of the nodes become active, 80% dominatee and 35% dominators. During

the first phase, among the active nodes, some nodes are elected dominators, and some other become dominatees. In the second phase, some dominatees become dominators to have a connected structure. In conclusion, a node changes its cds-state in average 2 times so that the structure becomes valid. Moreover, the number of changes per node is stable according to the number of participants.

Finally, the behavior of the structure was studied during the time (fig. 4). With a network of 50 nodes, k_{cds}=1 and $k_{cluster}$=2, idle and active nodes are only present during the construction part, in the very first seconds. Dominators are elected but, being redundant, they become dominatees after a few seconds. During the maintenance, the structure is very stable: the number of dominators and dominatees is almost the same during all the simulation. Slight variations appear because of packet collisions: some `hello` or `ap-hello` packets are lost. The nodes *believe* that a topology change occurs in the neighborhood.

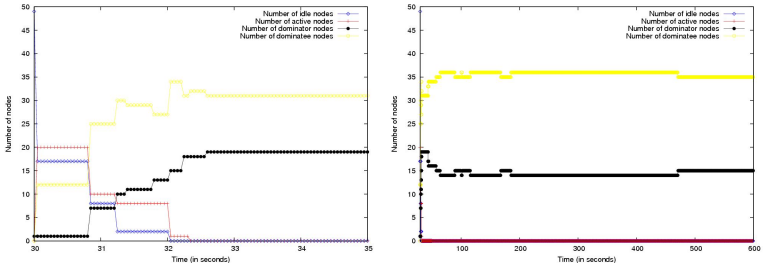

Fig. 4. Number of idle/active/dominator/dominatee nodes during a 600s simulation (k_{cds}=1 / $k_{cluster}$=2 / 50 nodes)

Temporary failure We simulate a temporary failure: a dominatee becomes arbitrarily dominator, or a dominator becomes arbitrarily dominatee (fig. 5). This simulates a node failure (after for example a power-off). We can remark that the convergence time is inferior to 3s for a dominator if the CDS is required to be strictly connected. If the CDS must be only largely connected (which is the case for flooding applications), the convergence time is inferior to 1.2 seconds. The convergence time is longer when k_{cds}=3 because of reconnection complexity.

8 Conclusion

In this article, we propose the construction and the maintenance of a virtual structure for the self-organization of ad hoc networks. A backbone helps to collect the traffic control and to distribute it efficiently in the network. Clusters create a hierarchical organization of the ad hoc networks, clusterheads managing their cluster, i.e. their services area. The construction algorithms are proven to construct a Connected Dominating Set and a clustering scheme in any ad

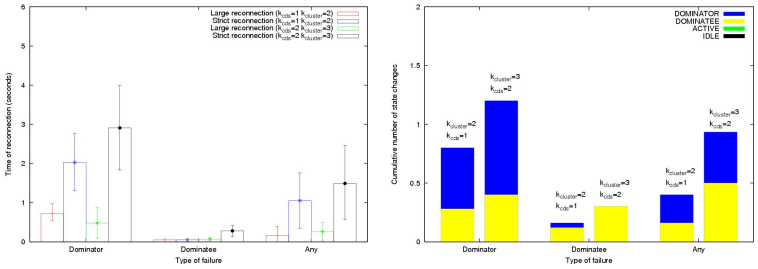

Fig. 5. Reconnection Time and number of changes after a temporary failure

hoc network. The complexity in messages is reduced, which represents a required property in wireless networks. Moreover, ad hoc networks present a very volatile topology. In consequence, maintenance is vital. The proposed algorithms are proven to be self-stabilizing if topology changes occur in the network. Both the construction and the maintenance are time-bounded. The virtual structure is studied trough simulations. The cardinality and connectivity of the structures remain very stable. Moreover, the structure is constructed efficiently and quickly. When a temporary failure occurs, the maintenance algorithms reconnect the structure in a very small delay, and only a few nodes are impacted by changes. A local topology change impacts only locally the structure. This explains the good scalability of the virtual structure. The proposed virtual structure for self-organization is proven to be self-stabilized. In consequence, such a scheme is very flexible and totally parameterizable. It constitutes a genuine framework to deploy efficiently new services in ad hoc networks: routing could be deployed on this self-organization, taking into account the natural scalability of the virtual structure.

References

1. K. Alzoubi, P.-J. Wan, and O. Frieder. New distributed algorithm for connected dominating set in wireless ad hoc networks. In *Hawaii International Conference on System Sciences (HICSS)*, Big Island, USA, January 2002. IEEE.
2. S. Butenko, X. Cheng, D.-Z. Du, and P. M. Pardalos. On the construction of virtual backbone for ad hoc wireless networks. In *Cooperative Control: Models, Applications and Algorithms*, pages 43–54. Kluwer Academic Publishers, 2003.
3. M. Cardei, X. Cheng, X. Cheng, and D.-Z. Du. Connected domination in ad hoc wireless networks. In *International Conference on Computer Science and Informatics (CSI)*, North Carolina, USA, March 2002.
4. E. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, November 1974.
5. S. K. S. Gupta and P. K. Srimani. Self-stabilizing multicast protocols for ad hoc networks. *Journal of Parallel and Distributed Computing*, 63(1):87–96, 2003.
6. C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7):1265–1275, 1997.

7. M. Schneider. Sef-stabilization. *ACM Computing Surveys*, 25(1):45–67, March 1993.
8. F. Theoleyre and F. Valois. A virtual structure for mobility management in hybrid networks. In *Wireless Communications and Networking Conference (WCNC)*, volume 5 of *1*, pages 1035–1040, Atlanta, USA, March 2004. IEEE.
9. F. Theoleyre and F. Valois. About the self-stabilization of a virtual topology for self-organization in ad hoc networks. Research Report, INRIA, August 2005.
10. J. Wu and F. Dai. Distributed dominant pruning in ad hoc wireless networks. In *International Conference on Communications (ICC)*, pages 353–357, Anchorage, USA, May 2003. IEEE.