A self-organization structure for Hybrid Networks

Fabrice Theoleyre, Fabrice Valois

A Self-organization Structure for Hybrid Networks

Fabrice Theoleyre *, Fabrice Valois

ARES INRIA / CITI, INSA-Lyon, F-69621, France

Abstract

This paper focuses on the self-organization of ad hoc and hybrid networks. We propose to organize networks with two integrated virtual structures: a backbone and clusters. The backbone helps to optimize the flooding of control packets, and to offer a natural prolongation of the backbone of wired networks. Clusters help to hierarchize the network, each of which is managed logically by its clusterhead. Since MANETs are mobile, we propose distributed algorithms for both the construction and the maintenance to preserve an efficient virtual structure despite topology changes. Simulations results exhibit the robustness and persistence of the proposed virtual structure.

Key words: virtual structure, self-organization, backbone, clusters, event-driven maintenance

1 Introduction

Mobile ad hoc networks (MANet) are networks *ready to work* [1]: mobile nodes collaborate spontaneously via wireless communications, without any fixed infrastructure. Hybrid networks extend the MANET with a connection to the Internet via one or several access points. Since the last decade, several protocols were proposed to deal with routing[2], addresses allocation[3], mobility management[4], taking into account the radio link instability and the topology changes because of mobility. However, most approaches consider a MANET as a flat network: the protocol considers that all the nodes must participate jointly, in the same proportion. According to us, protocols can take advantage

* Corresponding author.
Email addresses: `fabrice.theoleyre@imag.fr` (Fabrice Theoleyre), `fabrice.valois@insa-lyon.fr` (Fabrice Valois).

of a preliminary hierarchized structure. More importantly, such a hierarchy should be shared by all the protocols (e.g. routing, addresses attribution).

First what is self-organization ? For us, self-organization allows to structure the network, to order an anarchy. Self-organization should create a structure useful for a wide variety of protocols, such that the protocols are more efficient when they are based on this hierarchy rather than on the flat network. For example, clusters allow to split the network in homogeneous areas, introducing a hierarchy useful to aggregate routes or manage locally the addresses attribution. Thus, clustering could constitute a good candidate to take part in self-organization. In the same way, backbones consist in electing a connected set of nodes to coordinate the network, to optimize for example information flooding. Consequently, backbones can also represent a self-organization structure. Backbones and clusters are often called *virtual structures*. Consequently, we will use indistinctly the terms *virtual structure* and *self-organization*. Besides, self-organization must present the following properties:

- Persistence: a structure is persistent if the number of its changes is minimized. For example, a clusterhead should remain clusterhead for a long time. A protocol can only exploit a self-organization scheme whose structure is stable, since an unstable hierarchy often requires an important overhead for protocols updates.
- Self-stabilization: protocols that maintain a self-organization structure must converge to a legal state in a finite time, whatever the initial state is. The self-stabilization property of an algorithm guarantees that it will converge in any conditions.
- Robustness: in order to handle radio topology changes, the impact on the structure properties must be minimized.
- Localized decisions: the construction and maintenance algorithms must be distributed. Furthermore, in order to reduce the global overhead and convergence delays, nodes should only use local information in making their decisions.

Virtual structures present several advantages. First, they can enhance the scalability: by creating a hierarchy, they can limit the knowledge required by a protocol. For example, the topology of a zone in the first level of the hierarchy can be first aggregated by the zone leader, and then flooded in the second level of the hierarchy. For example, all the routes toward a particular cluster can be aggregated, reducing the overhead required for updates. Moreover, a virtual structure offers a logical view of the radio topology, hiding some individual changes. For example, a node which changes its radio neighbors but which remains in the same cluster will not change the cluster topology. This stable view could be useful for upper layer protocols. Besides, virtual structures can optimize the global floodings: many protocols require a global broadcast in the network to flood either an information update or a request. Nevertheless, blind floodings in the whole network cause problems of redundancy (useless transmissions) and reliability (collisions): the so-called broadcast storm [5]. Clusters and backbones help to reduce the overhead by allowing only a subset of nodes to forward the packets. Finally, since MANETs are constituted by an heterogeneous collection of nodes, it could be relevant that the self-organization structure reflects this heterogeneity. Thus, a protocol based on this structure could allow *stronger* nodes to participate more intensively. For example, a leader will surely be a node with a low mobility and important energy or CPU capacities.

The main contribution of this article is to present an unified self-organization with an associated event-driven maintenance. The virtual structure presented here combines the assets of clusters and backbones, construction algorithms being closely integrated in order to reduce the overhead. Although many propositions deal uniquely with construction, we propose a maintenance protocol to preserve an efficient structure. Such a protocol could be adapted to maintain any Connected Dominating Set structure. Besides, the backbone presented here forms a directed tree, which could be useful for routing or localization. For example, Cellular IP could be directly adapted to such a structure, taking benefit of the tree hierarchy. Finally, we optimize the persistence of the virtual structure. Else, many changes in the hierarchy would impact a routing or localization protocol based on this self-organization.

First, we will expose the related work about self-organization structures, with the focus on backbones and clusters. Then, the section 3 will present an overview of our proposition. Section 4 will detail the backbone construction and maintenance. Section 5 will present the clustering algorithms and explain their integration with the backbone algorithms. Simulations in section 6 will expose the behavior of the proposed algorithms. Finally, section 7 will conclude the article, giving some perspectives.

2 Related work

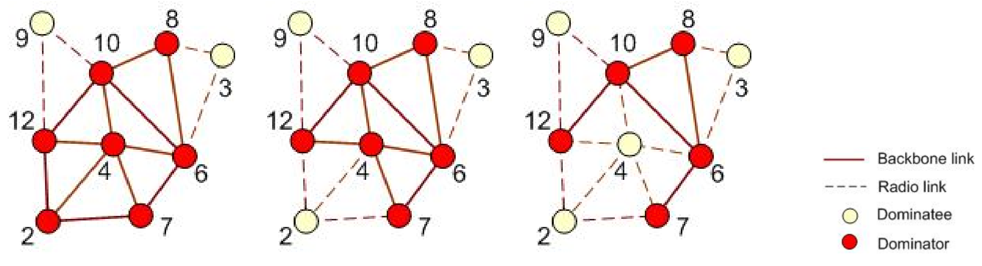
2.1 Backbone

A backbone is constituted by a connected subset of nodes. Consequently, if only backbone nodes are allowed to forward a flooded packet, the whole backbone and its neighbors will receive the information. Consequently, global floodings require less transmissions, avoiding the potential broadcast storm problem [5]. Moreover, since the backbone consists mainly of *stronger* nodes, weaker nodes participate less actively, which saves their power-energy.

A connected Dominating Set (CDS) is a set of *dominators* such that each normal node (*dominatee*) is neighbor of at least one dominator, and such that the set of dominators forms a connected set. A *Minimal Connected Dominating Set* (MCDS) is the CDS of minimum cardinality. If only the MCDS nodes are authorized to forward control packets, the amount of control packets sent for a flooding is optimal. However, since MCDS is NP-hard[6], a CDS is more relevant in a distributed environment. Moreover, problems of robustness could appear when topology changes occur: a MCDS constituting a tree, the deletion of one single edge surely disconnects the CDS. Consequently, an efficient backbone should propose a trade-off between reliability and overhead. We extended this definition to a k-CDS: the number of hops from one node to the CDS is a parameter, k.

Several heuristics were proposed to construct distributively a CDS in 2 main steps [7–10]. Firstly, a dominating set is constructed: each node must be neighbor of at least one node of the CDS. The CDS nodes are *dominators*, the other nodes being *dominatees*. A leader is present in the network and declares itself dominator, its neighbors becoming dominatees. Using an election process, the neighbors of these dominatees become dominators if they own the highest weight in their own neighborhood. The algorithm then reiterates. The weight used for the election is based either on degree[9] or identifier[7,8,10]. The second step must connect the dominating set to form a CDS. [8] proposes an iterative exploration, expanding a tree from the leader, coloring the dominatees with the highest number of dominator neighbors. [9] is very close to the previous one but uses a timer between the first and second steps. [7] proposes a more *best-effort* approach: a connected dominator invites locally other dominators to connect themselves. The leader constituting initially the single connected dominator, the structure forms finally a connected set. [10], before the creation of a dominating set, constructs a spanning tree. This spanning tree is then used to interconnect the elected dominators. However, [7–10] do not propose any algorithm to maintain the CDS, although topology changes will surely render the CDS inefficient.

[11] proposes also to start from a dominating set constructed in the same way. Each node transmitting periodically `hellos`, the dominating property can be easily maintained. Besides, dominators will send their `hellos` with a TTL of 3 hops. Thus, each dominator can maintain a path (a *virtual link*) for each dominator which is at most 3 hops far. The set of dominators with virtual links forms a connected dominating set. Since this algorithm is local, construction and maintenance algorithms are identical. However, topology changes can quickly create sub-optimal virtual links, and `hellos` from dominators can generate a significant overhead. [12] extends this concepts in constructing a k-CDS. Thus, `hellos` for the dominating set must be sent k hops far, and `hellos` for virtual links 2k+1 hops far. However, the same problems persist.



(a) Redundancy elimination [13]   (b) Rule 1 [13]   (c) Generalized rule [14]

Fig. 1. Example of the construction of a CDS with the Wu & Li algorithm

[13] presents a localized construction of a CDS. Each node broadcasts `hellos` to discover its neighborhood. Then, it executes the following rule: *A node is dominator if at least two of its neighbors are not connected with each other.* Let's consider the example in figure 1(a). Node 9 has 2 neighbors (12 and 10), and these neighbors are connected with each other. Thus, 9 is a dominatee. Node 8 has three neighbors $(3, 6, 10)$ but 3 and 10 are not connected. Consequently, 8 is a dominator. Note the CDS cardinality is substantial. Thus, the authors propose the so-called rules 1 & 2 to reduce the CDS redundancy. In the rule 1, a node u becomes a dominatee if it is covered by v, i.e. $N(u) \subseteq N(v)$ and $id(u) < id(v)$. In the rule 2, a node u becomes a dominatee if it is covered by two neighbors v and w, i.e. $N(u) \subseteq N(v) \cup N(w), id(u) < id(v), id(v) < id(w), v \in N(w)$. The dominators form finally a connected dominating set, and the authors proved that shortest paths pass through this CDS. Let be the example in the figure 1(b). Node 2 has a neighbor with an higher id, 4 and this node is covering all its neighbors (7 and 12). After applying the rule 1, 2 becomes a dominatee.

The authors extend their algorithm in [14] by eliminating more redundancy using the k-rule: *a node is covered if it has k neighbors with an higher id, connected, and covering its whole neighborhood.* This proposition can be extended in a more general form[15]: *a node is covered when a connected set of its neighbors with an higher id forms a covering of its whole neighborhood.* Let's consider the example in figure 1(c). Node 4 has a set of neighbors with an higher id $(6, 10, 12)$ which forms a connected dominating set of its own neighborhood $(2, 6, 7, 10, 12)$. Thus, 4 becomes a dominatee. However, the CDS of Wu & Li was conceived to optimize the broadcasts, not the persistence of the structure. Simulations in the section 6 exhibit many backbone changes. Thus, a hierarchical routing based on this structure could be less efficient. In the same way, [16] proposes to elect dynamically forwarders to reduce the overhead of a flooding. However, the structure being source oriented, its persistence is null.

2.2 Clusters

The clusters divide the network into several small sets of nodes, introducing a hierarchy. A cluster is usually defined by its diameter constraint: two nodes in the same cluster must be at most 2 hops far. Eventually, a leader, the clusterhead, could be elected per cluster. Clusters can be useful for MAC[17], power control[18], topology control[19], or routing[20].

[21] presents a simple algorithm largely used in the literature to form clusters. After initiating a neighborhood discovering, a node elects itself clusterhead if it has the highest identifier among all its neighbors without clusterhead. A new clusterhead advertises its decision through an `hello`, allowing its neighbors to join its cluster. [17] proposes a maintenance algorithm for this structure: each node verifies that it is at most 2 hops far from any node of the same cluster. If the diameter constraint is violated, the cluster must be splitted. Oppositely, two clusters can merge if the diameter constraint remains valid after the merging. Thus, the 2-neighborhood knowledge is required (with the radio links among the 2-neighbors). The lack of clusterheads is, according to us, a drawback for a self-organization structure, no leader being available to manage the cluster. [19] proposes to use clusters to control the topology. In a first time, clusters are constructed, with one clusterhead per cluster. Then, the clusterhead will allocate a power to each of its members in order to maintain a connectivity in and among the clusters. Moreover, robustness is improved in maintaining several paths for each pair of nodes. However, the maintenance could be problematic in a highly mobile environment: the radio ranges must be continuously updated to avoid a network partition.

ZRP [22] introduces the concept of zones, which could be related to clusters: the zone of a node is the set of its neighbors at most r hops far. Then, the authors propose a routing protocol using these zones. However, different nodes, potentially neighbors, have different zones. In other words, the existence of a zone is node-specific. The authors propose to optimize the route discovering in forwarding the requests in unicast to the nodes exactly r hops far, the border nodes. However, such a flooding is redundant since several nodes can forward the same request to the same border node. According to us, a static and common view of the hierarchy could be more efficient to self-organize the network, and optimize for example flooding or routing.

[23,24] propose to limit the number of clusters: a node is allowed to be at most k hops far from its clusterhead. In [23], a first phase allows to propagate the highest identifiers k hops far, and a second phase allows to propagate the identity of the elected clusterheads. [24] proposes to construct first a spanning tree and then cut branches when they have a height of k hops. However, the construction of a spanning tree requires a significant delay and overhead.

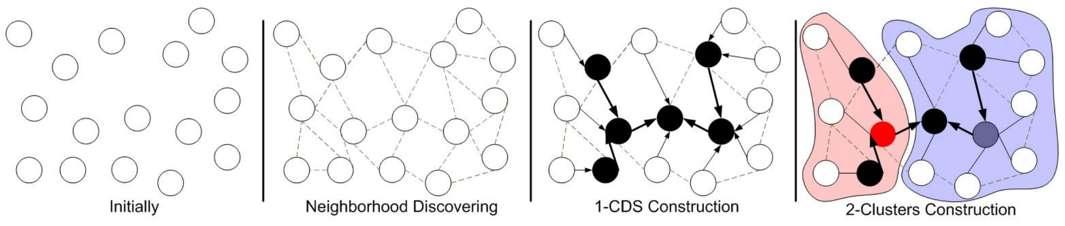


Fig. 2. Construction of the virtual structure

Moreover, neither [23] nor [24] present any maintenance procedure.

3 A virtual structure of self-organization: a general overview

We propose a self-organization scheme based on the construction of a virtual structure (fig. 2) in order to simplify later the implementation and the deployment of new services like routing, services discovering or addresses attribution. To reach such a goal, a virtual backbone is constructed. It is constituted by *strong* nodes which are responsible to collect the control traffic. The global load is reduced, decreasing the broadcast storm problem. Backbone nodes are elected according to their properties to maintain an efficient and stable backbone, presenting a reduced cardinality. Moreover, the backbone constitutes a tree, each node maintaining the identity of an unique parent. Simultaneously, the network is cut off in homogeneous areas, creating a second level of hierarchy in the network: a leader may manage its services area, organize the routing and attribute addresses. The backbone and the clusters algorithms are full integrated, reducing the overhead to construct and to maintain the virtual structure. We propose also event-driven maintenance algorithms, optimizing the persistence of the structure.

The construction proceeds in 3 phases. First, a neighborhood discovering is initiated, the knowledge of the k_{cds}-neighborhood being required for the virtual structure construction. Then, a backbone is built by a distributed election: a node is at most k_{cds} hops far from a backbone member, and the backbone forms a connected structure. Finally, based on the virtual backbone, clusterheads are elected and form clusters of radius $k_{cluster}$.

3.1 Neighborhood discovering

To construct the backbone, each node must know its k_{cds}-neighborhood. Moreover, each node must distinguish unidirectional and bidirectional links. Thus, a node sends periodically `hello` packets containing information about the source (address, weight, parent, clusterhead), the list of neighbors (address, weight, parent, clusterhead) and some control information (TTL, type of packet). Consequently, a dominator can maintain the list of its children (dominator for which it is the parent) and dominatees (dominatees which chose it as parent). If a *distance vector* approach is used, each node sends the list of its ($k_{cds}-1$)-neighbors so that each node reconstructs the list of its k$_{cds}$-neighbors. However, convergence delays could be problematic in mobile ad hoc networks. Thus, we propose to use a *state-link* approach: each node sends in its `hellos` the list of its 1-neighbors, the packets being propagated $k_{cds}-1$ hops far. This approach requires more packets, but the convergence is reached after one single transmission. Moreover, we have only a local flooding of `hellos`, restricted to $k_{cds}-1$ hops, limiting in consequence the overhead.

3.2 Stability metric

The algorithms presented further are based on elections, requiring a weight. This weight should optimize the stability of the topology. In classical approaches, the lowest identifier or the highest degree are often used. However, such a scheme is inefficient since these criteria can not reflect for example the aptitude of a node to act as a backbone member during a long time, creating a stable structure. In consequence, we propose a stability weight combining several criteria:

- Distance to an optimal degree (Δ): a node with a too high degree will constitute a bottleneck and will be exposed to more collisions, creating congestion points at the MAC layer. A node with a too low degree will serve less nodes, requiring an higher number of leaders in the network. Thus, the degree of a node should be as close as possible to an optimal degree Δ_{opt}. Let Δ_{real} be the real degree, i.e. the number of neighbors. Finally, $\Delta = |\Delta_{real} - \Delta_{opt}|$. We can note that Δ_{opt} depends on the application ($\Delta_{opt} = 8$ in the simulations).
- Mobility ($\mathcal{M}$): a node must keep its clients as long as possible. Thus, the relative mobility is much more important than the absolute mobility. Consequently the GPS is not necessary. We propose to use the changes in the neighborhood table to estimate the relative mobility. Let N_t be the set of neighbors of a node N at time t. Formally, $\mathcal{M} = \frac{|N_{t+\Delta t} \setminus N_t| + |N_t \setminus N_{t+\Delta t}|}{|N_t \cup N_{t+\Delta t}|}$. Eventually, a moving average could be computed in order to smooth the value of $\mathcal{M}$.
- Energy ($\mathcal{E}$): a backbone node or a clusterhead will spend more power energy than other nodes. A node with important reserves will be privileged. $\mathcal{E}$ represents the energy reserve of a node.

A single metric combines this criteria in a non linear manner:

$$P_{stability} = \mathcal{E}\left(\alpha \cdot (1+\Delta)^{-1} + \beta \cdot (1+\mathcal{M})^{-1}\right)$$

α and β represent weighting factors. If required, other criteria such like the processing power of computing devices could be integrated in this metric, depending on the application. In our stability metric, α is relative to the degree. Thus, it will impact the number of nodes which will be elected and the congestion at the MAC layer (as explained above). Besides, an high β will favorize the nodes which present a low relative mobility, and consequently improving the stability. Since we try to optimize the persistence of the structure instead of its cardinality, we chose during the performance evaluation $\alpha \ll \beta$.

4 Backbone construction and maintenance

The backbone is constructed before the clusters. Thus, the distance via the backbone from a node to its clusterhead can be limited, only backbone nodes can participate to the clusterhead election (optimizing the overhead) and a clusterhead is forced to be a backbone member. Moreover, the maximum distance from one node to the backbone (k_{cds}) is a parameter of our solution. In volatile environments, a small k_{cds} allows to limit the backbone disconnections. In quasi-static environments, k$_{cds}$ could be high since less topology changes occur.

4.1 Construction

We define four states for a node:

- dominator: a backbone member.
- dominatee: a client, at most k$_{cds}$ hops far from at least one dominator.
- active: a node in election, which could be elected dominator after a timeout.
- idle: a node which waits for a signal to initiate the backbone construction.

Like some other propositions, a dominating set is constructed. Then, it is interconnected to form a connected structure.

First, a Maximal Independent Set (MIS) is elected. In a MIS, no dominator is adjacent to another dominator, and no other node can be elected as dominator without violating this constraint. A MIS represents, by definition, a dominating set. A node which changes its state sends immediately an `hello` packet to notify its decision to its k$_{cds}$-neighborhood. We propose here a distributed algorithm processing in waves, described in algorithm 4.1. The leader becomes

Algorithm 4.1: PacketReception(*packet*)

```
state ← GetStateSource(packet)
MyOldState ← MyState

if (state = DOMINATOR) and (MyState = IDLE or ACTIVE)
  MyState ← DOMINATEE
  MyParent ← Source
else if (state = DOMINATEE) and (MyState = IDLE)
  MyState ← ACTIVE
  Trigger(Timer)

if IsExpired(Timer) and HighestWeight(ActiveNodes) = me
  MyState ← DOMINATOR
  MyParent ← $\emptyset$

if MyState $\neq$ MyOldState
  if MyState $\neq$ DOMINATEE
    ttl ← $k_{cds}$
  else
    ttl ← $k_{cds}+1$
  SendHelloPacket(MyState , ttl)
```

the first dominator in the network and the precedent rules are applied. In hybrid networks, the Access Point constitutes the natural leader of the network. In an ad hoc network, a leader must be elected distributively using for example the algorithm described in [25]. An example of the construction of such a dominating set with $k_{cds} = 1$ is presented in figure 3 (step 1 to 4). The leader becomes the first dominator and sends an `hello` packet. Then, its neighbors become dominatees, and the 2-neighbors of dominatees become active (step 2). Indeed, a dominatee node sends an `hello` packet with a TTL of $k_{cds}+1$ since an elected active node must be at most $2k_{cds}+1$ hops far from another dominator. After the timeout, two active nodes have a maximum weight in their neighborhood and are elected dominators (step 3), their neighbors becoming dominatees, and other nodes active. Finally, the construction of a dominating set finishes in the step 4. All state changes are advertised to neighbors via `hellos`.

Then, the dominating set must be interconnected. We chose to privilege the delay and the robustness rather than the cardinality. If the backbone is redundant, the dominators in excess will be eliminated during the maintenance. We adapt here the algorithm proposed in [7]. We can remark that a k_{cds}-CDS can be constructed from a k_{cds}-dominating set in interconnecting all the dominators $2k_{cds} + 1$ hops far. The intuition behind this idea follows. We consider two dominatees that have different parents. These dominatees are at most k_{cds}

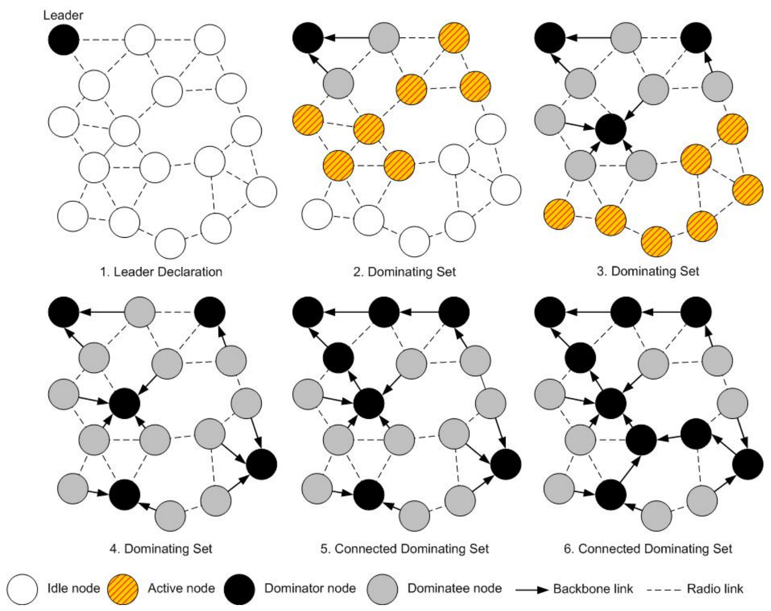


Fig. 3. Construction of the Connected Dominating Set (example with k_{cds}=1)

hops far from their own parent. Thus, the two corresponding dominators are at most 2k$_{cds}$+1 hops far. We can verify that if we interconnect all this kind of dominators, we form a connected set of dominators.

Initially, the leader constitutes the single connected dominator. It sends a `cds-invite` in broadcast with a TTL of 2k$_{cds}$+1. A dominatee forwards such a `cds-invite` with a TTL t if it already forwarded less than $max_{cds-invite}$ with a TTL superior or equal to t, $max_{cds-invite}$ being a parameter of the protocol. In a collision free environment, we should set $max_{cds-invite}$ to 1. A disconnected dominator which receives a `cds-invite` sends a `cds-accept` in unicast, along the inverse route, and sets its parent to the next intermediary node in this inverse route. Simultaneously, it sends itself a `cds-invite` to allow the connection of dominators at most 2k$_{cds}$+1 hops far. A dominatee forwarding the `cds-accept` will set its parent to the next hop, and its child to the previous hop. Finally, the previously disconnected dominator is well connected to the backbone, through a path of dominatees which became dominators. This process reiterates for each dominator, and the backbone forms finally a connected structure. Moreover, each dominator has a parent in the backbone, constituting the next hop toward the leader. In the same way, a dominatee has a parent, at most k$_{cds}$ hops far: this parent is dominator and a path exists constituted only by dominatees having chosen the same parent. The backbone forms finally a tree. An example of the CDS connection is represented in figure 3 in steps 5-6. The leader sends a `cds-invite`, allowing the dominators exactly 2 hops far to connect themselves to the leader, forcing dominatees to become dominators. A second wave of `cds-invites`, sent by these newly connected dominators, allows to connect the remaining dominators.

The backbone must remain connected and must cover all the dominatees. In consequence, we propose the following procedures.

A dominatee just verifies that it has a valid dominator P, l_P hops far: P is a dominator and a neighbor declares in its `hellos` that it is at most $l_P - 1$ hops far from P. Naturally, l_P must remain inferior to k_{cds}. If the current dominator is not valid, the dominatee searches a new valid parent. A dominatee keeps the same dominator as long as it is valid, maximizing the persistence of the backbone. When a dominatee changes its parent, it sends a gratuitous `hello` packet to advertise its decision, to flush obsolete information for its neighbors. If a dominatee does not find any valid dominator, it becomes active. An election will occur, and one or several active nodes will be elected dominators, their k_{cds}-neighbors becoming their dominatees. Finally, the new dominators will execute the maintenance reserved for dominators to reconnect the backbone.

The backbone connectivity must be maintained proactively, since otherwise there is no way for a node to detect a backbone disconnection. Consequently, the leader sends periodically `leader-hellos` in multicast, with a strictly increasing sequence id. A dominator forwards a `leader-hello` only if it comes from its parent in the backbone. Thus, a dominator which receives a `leader-hello` can legitimately consider itself connected to the leader via the backbone. A dominator adds the source of a `leader-hello` in the list of its secondary parents if the sequence id of the packet is strictly superior than the sequence id of the last `leader-hello` received from its parent. A dominator D can consider itself disconnected if it has missed several `leader-hellos` or if the parent of D is either not dominator or not yet neighbor. A disconnected dominator chooses as new parent the node with the highest weight in its secondary parent list. Then, it sends a gratuitous `hello` packet to advertise its decision, and particularly to its new parent. However, the list of secondary parents could be empty. Thus, we propose the following reactive parent discovering:

- D generates a `cds-reconnect` with the sequence id of the last `leader-hello` received from its parent. It sends the packet in broadcast with a TTL of $2k_{cds} + 1$.
- A dominatee with the parent D forwards the packet in broadcast.
- Other dominatees forward the packet in unicast toward their parent, minimizing the overhead.
- A dominator can answer if the sequence id of the last `leader-hello` received from its parent is strictly superior to the sequence id required by D. Consequently, this dominator cannot be a descendant of D. It sends a `cds-invite` in unicast on the inverse route.

- The disconnected dominator will receive the `cds-invite` and send a `cds-accept`, forcing the intermediary dominatees to become dominators.

Each dominator which receives or forwards a `cds-invite` can register the source as a secondary parent. This cache could be assimilated to a proactive maintenance: each node monitors continuously its neighborhood in order to reconnect itself without latency if the backbone becomes later disconnected. Moreover this maintenance is gratuitous: a node simply adds in its cache information about the `leader-hellos` and `cds-invites` it receives, even if it is not the final destination.

If the radio medium is heavily loaded, many collisions occur. Hence, many cds-reconnections are sent, increasing the traffic. A dominator D which tried max_{retry} reconnections becomes idle and sends a `cds-break` in multicast to its children and dominatees. If a node receives a `cds-break` and the source is its parent, it becomes idle and forwards the packet. Finally, the branch of the CDS is broken: all nodes become idle and wait an exterior solicitation for the reconstruction. A dominator which has an idle node in its k_{cds}-neighborhood table sends a `cds-invite` with the TTL k_{cds}. However, in the worst case, the idle area can be exactly $\mathrm{k}_{cds}+1$ hops far from the nearest dominator. Thus, a dominatee, neighbor of its dominator, and exactly k_{cds} hops far from an idle node, forces its dominator to send a `cds-invite` with a TTL $k_{cds}+1$. An idle node I which receives a `cds-invite` becomes active. I buffers the `cds-invite` to store the source as a secondary parent if it is elected further dominator.

We proposed mechanisms to reconnect the backbone and to maintain the dominating set property of the backbone nodes. Hence, nodes could be elected dominators to maintain an efficient backbone. However, the cardinality of the structure must remain limited, via a redundancy elimination procedure. A dominator is useless if it has no child (dominator for which it is the parent), and no dominatee exactly k_{cds} hops far. A useless dominator D sends a `cds-useless` and becomes dominatee. Its dominatees receive the `cds-useless`, forward it, and change their dominator to the parent of D. In the same way, a mechanism to limit the backbone depth, i.e. the maximal distance from one dominator to the root, can also be proposed. When a dominator D receives a `leader-hello` from one dominator P_{new} nearer from the root than its old parent, it changes its parent to P_{new}. Clearly, P_{new} could not be a descendant of D. The CDS depth is reduced, and thus the number of leaves is increased. More dominators can declare themselves useless, and the cardinality is reduced. Moreover, the depth being minimized, the number of `leader-hello` suffering from collisions is reduced.

5 Clustering

5.1 *Construction*

As explained above, only dominators take part in the construction of clusters: a dominatee joins automatically the cluster of its parent.

A dominator initiates the construction as soon as its neighborhood has a valid cds-state (either dominatee or dominator). We want to limit the distance via the backbone links from a node to its clusterhead. Thus, a virtual neighborhood discovering is initiated. A virtual neighbor is either a parent or a child in the backbone: the virtual neighborhood is a subset of the radio neighbors restricted to dominators. `cluster-hellos` containing the virtual neighbors are propagated along the backbone, with a TTL of $k_{cluster} - k_{cds}$. A dominator treats and forwards a `cluster-hello` if it comes from a virtual neighbor. Besides, a dominator stops to send `cluster-hellos` when no dominator without clusterhead is known locally.

A dominator which has the highest weight among all its $(k_{cluster} - k_{cds})$-virtual neighbors without clusterhead is elected clusterhead. It sends a gratuitous `cluster-hello` to advertise its decision. A dominator forwarding the packet chooses the source as clusterhead if the previous hop has chosen the source as clusterhead, and if the source is at most $k_{cluster} - k_{cds}$ hops far, forcing the clusters to be connected. The `cluster-hellos` are only used during the construction of clusters, maintenance algorithms being different. Since dominatees are at most k$_{cds}$ hops far from a dominator, any node is at most k$_{cluster}$ hops far from its clusterhead.

5.2 *Maintenance*

A dominatee does not take part to the cluster maintenance, keeping the clusterhead of its parent. The id of this clusterhead is extracted from the `hellos` coming from its parent.

To maintain the clusters, a dominator sends additional particular information in `hellos`: the distance of its clusterhead, and the node constituting the next hop toward it (the *relay*). A dominator D considers its clusterhead C_{old} lost if C_{old} is not yet neighbor or dominator or if C_{old} announces a different clusterhead $k_{cluster} - k_{cds}$ hops far. If the *relay* announces a new clusterhead C_{new} at most $k_{cluster} - k_{cds} - 1$ hops far, D takes this new clusterhead, and updates its distance to the clusterhead. It sends a gratuitous `hello`: other dominators receiving this packet will eventually change their clusterhead and update their obsolete information. However, if the *relay* announces a clusterhead exactly $k_{cluster} - k_{cds}$ hops far, D will try to find another relay among its virtual neighbors. This relay R must announce in its `hello` a new clusterhead C_{new} at most $k_{cluster} - k_{cds} - 1$ hops far or the clusterhead C_{old}, D being not the relay to the clusterhead for R. If no clusterhead is available, D will simply become its own clusterhead.

6 Performances evaluation

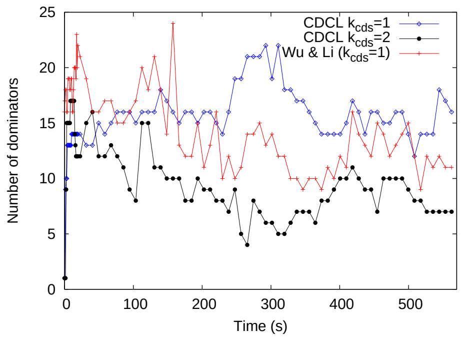
We present here our simulation results obtained from OPNET Modeler. We used the OPNET 802.11b MAC protocol with a bit rate of 11 Mb/s, a standard 300 m radio range, in DCF mode, without RTS/CTS. Each node moves according to the random waypoint mobility model, without pause time. Several simulations are executed for each set of parameters, the 95% confidence intervals being systematically plotted. We consider as generic a mobility of 5m.s^{-1}, 40 nodes, and a degree of 10. One of the node is configured statically to be the leader (like an Access Point connecting the ad hoc network to the Internet). `hello` packets are sent periodically every 4 seconds.

The results detail the relevance of the proposed virtual structure of self-organization. Particularly, the impact of k_{cds} and $\text{k}_{cluster}$ is studied. Results about algorithms convergence are presented. The impact of the degree, the mobility, and the horizontal scalability are also exposed. To evaluate the structure, we measured the cardinality, the backbone connectivity, the convergence time, the changes and the persistence. Finally, simulations allow to test the stability weight. If the $\text{k}_{cluster}$ value is not reported in the figure, $\text{k}_{cluster}$=k_{cds}.

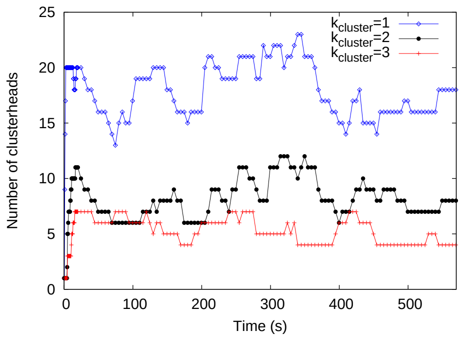
We compare the virtual structure proposed here (noted CDCL in the figures) with the CDS of Wu et al. [14], using the latest and most efficient redundancy elimination rule. [14] is a localized algorithm. Consequently, the algorithm does not need any specific maintenance algorithm. When a new information about its neighborhood is received, a node can immediately execute the Wu & Li rules in order to maintain a correct CDS. The Wu & Li algorithm is consequently executed continuously, when a change in the neighborhood occurs.

6.1 Impact of k_{cds} and $k_{cluster}$ on the virtual structure

We first simulated the algorithms with the standard parameters and for different values of k_{cds} and $\text{k}_{cluster}$. We reported the behavior of the structure during 600 seconds (fig. 4). We can first remark that the backbone cardinality decreases when the maximal authorized distance from a dominatee to the backbone (k_{cds}) increases (fig.4(a)): this constitutes an expected property. We can also observe that the number of dominators for our algorithm (with k_{cds}=1) and for Wu & Li is on average almost identical. We can remark a difference between 250 and 350 seconds, but this difference is not statistically representative, as observed later in the simulations. Besides, the number of dominators required when k_{cds}=2 is inferior to the number of dominators with Wu & Li. In the same way, if the cluster radius ($k_{cluster}$) is higher, the number of clusterheads decreases (fig. 4(b)).



(a) Impact of the maximal distance from one dominatee to the backbone (k_{cds})



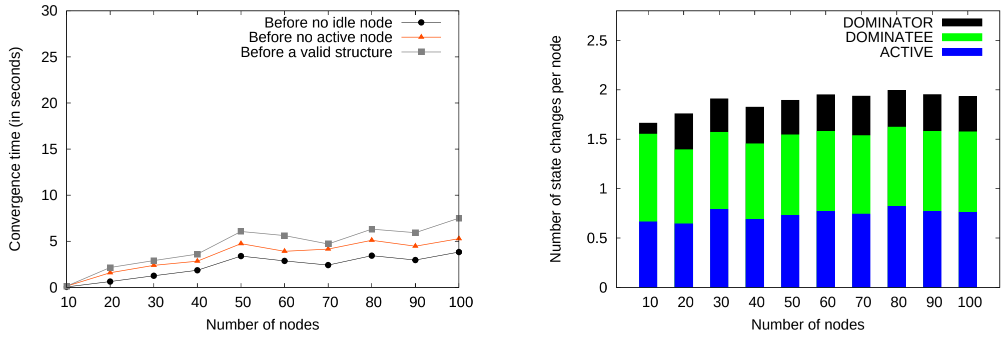
(b) Impact of the maximal distance from one node to its clusterhead ($k_{cluster}$)

Fig. 4. Behavior of the structure: 40 nodes, 5 $m.s^{-1}$, degree of 10

6.2 Convergence

Then, we investigated the behavior of the algorithms before the convergence. Indeed, the algorithms must converge quickly in order to be efficient only a few seconds after the initialization. The cluster algorithms converge very quickly, so results are not presented here. We consider that the backbone construction algorithms have converged if the dominators form a connected structure, and if each node is at most k_{cds} hops far from at least one dominator. We represented the obtained results in figure 5. The convergence time before having a valid CDS is less than 10 seconds, even with 100 nodes (fig. 5(a)). As can be observed in the figures, the time required for the elections (*no active node*) and for the dominating set interconnection (*valid structure*) is limited. Then, we measured the number of state changes before the algorithms converge, a state change being for example an idle node which becomes dominatee (fig. 5(b)). The algorithms do not oscillate, and a node changes on average only 2 times its state before being stable. More than one state change is required: for example, an idle node can become active and then dominator, or pass directly from idle to dominatee. This number of state changes is not impacted by the number of participants: the algorithms are robust. Detailed proofs of self-stabilization properties are given in [26].

Since [14] is a localized algorithm, a valid CDS is constructed when all the nodes obtain a consistent view of their neighborhood. Consequently, if no collision occurs, a node needs only two `hellos` packets to advertise all its radio links, to eliminate unidirectionnal links and to have a correct execution of the Wu & Li algorithm.



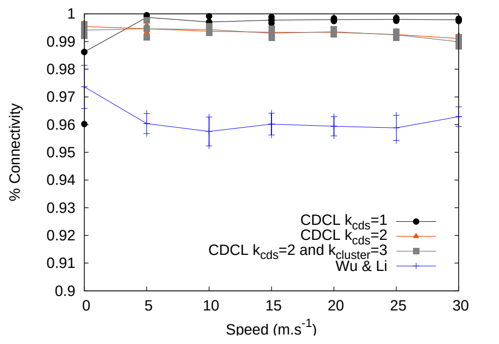
(a) Time required before convergence for each step of the construction

(b) Number of state changes per node before convergence

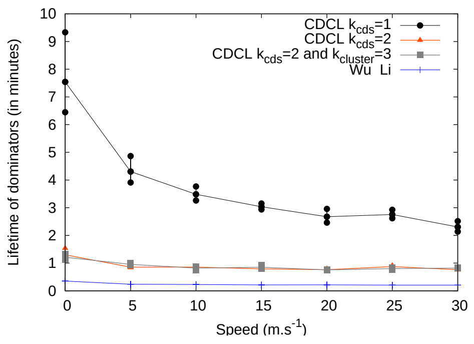
Fig. 5. Convergence of the backbone construction (dominating and connectivity properties): 40 nodes, 0 m.s^{-1}, degree of 10, k$_{cds}$=1

6.3 Mobility

Since in ad hoc networks, all nodes are mobile, we studied the impact of the mobility on the performances of the self-organization structure. Each node chooses independently a speed and a direction according to the random waypoint mobility model. Instead of varying the pause time, a node chooses a random speed between 0 and V$_{max}$, V$_{max}$ representing the x-axis (fig. 6). In a first time, we studied the backbone connectivity (fig. 6(a)). We can observe that the connectivity remains over 95%, even with very high mobilities. When k$_{cds}$=1, the backbone remains connected during a longer time: the distances for backbone reconnections and from a dominatee to its parent are reduced, faciliting the maintenance. Then, we measured the persistence of the backbone, i.e. the time during which a node keeps its state unchanged (fig. 6(b)). The persistence of the proposed backbone is much higher than the backbone constructed by Wu & Li: the dominator lifetime is 4 times higher. Wu & Li algorithm does not try to optimize the persistence, since the algorithm is memory-less. It was conceived originally to optimize the floodings, and not to consitute a good self-organization. If the dominator must attribute addresses or maintain routing tables, the Wu & Li changes could be problematic.



(a) Average backbone connectivity (average percentage of nodes which respect the CDS properties)



(b) Lifetime of a dominator (average time during which a dominator keeps its state of dominator)

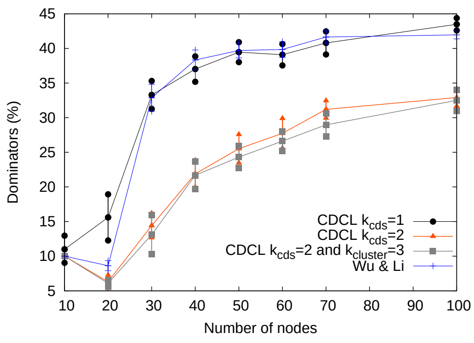
Fig. 6. Impact of the mobility: 40 nodes, degree of 10

6.4 Horizontal scalability

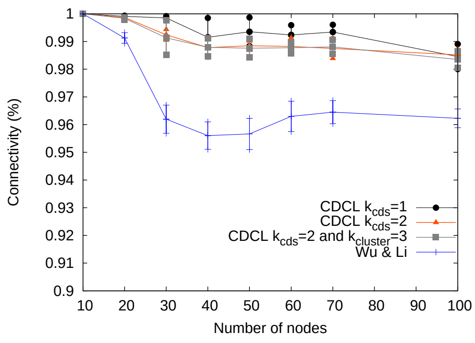
The horizontal scalability is an important criterion to evaluate a protocol: the performances must not decrease drastically when the network cardinality increases. Results are reported in figure 7. The backbone cardinality is first quantified (fig. 7(a)). The proportion of backbone members increases when the number of participants increases: the distance from the leader to the leaves of the backbone increases, and some `leader-hellos` are lost because of collisions. New backbone reconnections are required, the redundancy elimination is reduced. However, the cardinality remains acceptable even with a hundred nodes, which constitutes, according to us, a relatively important hybrid network. The backbone cardinality of Wu & Li and CDCL with k_{cds}=1 tends to be equal, as highlighted in the section 6.1. The backbone must also remain connected to be efficient (fig. 7(b)). And precisely, the connectivity is almost not impacted by the number of participants: the maintenance is robust, and allows to maintain an efficient backbone. The same remarks follow the observation of the cluster cardinality and connectivity. In consequence, results are not represented here.

6.5 Density

The average number of nodes per surface area can vary according to the applications, impacting the degree. However, a topology control [27] should be executed in order to reduce radio ranges and degrees, allowing to improve the frequency spatial reutilization and to reduce the power-energy consumption. Nevertheless, we propose here to study the impact of the node degree on the structure to confirm the robustness of the algorithms. 40 nodes are placed on the simulation area, the area size varying (fig. 8). In changing the



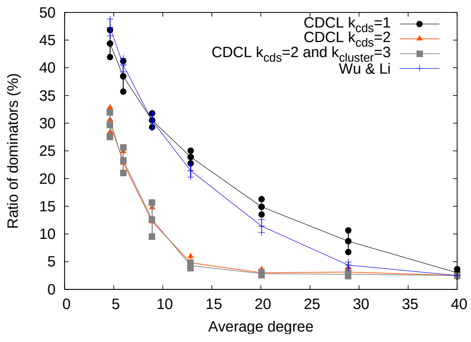
(a) Average backbone cardinality (percentage of dominators)



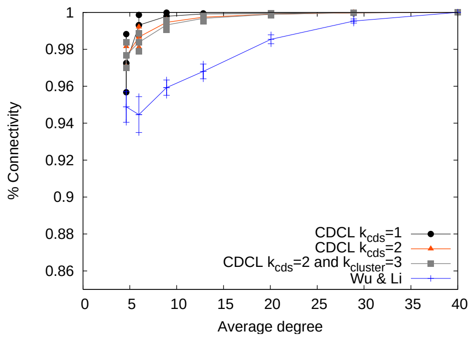
(b) Average backbone connectivity (average percentage of nodes which respect the CDS properties)

Fig. 7. Impact of the number of nodes (horizontal scalability): 5 m.s^{-1}, degree of 10

simulation area instead of the network cardinality, we separate the impact of the degree and of the horizontal scalability. We first measured the backbone cardinality (fig. 8(a)). The number of dominators decreases when the degree increases: the radio topology is much redundant. With a degree of 20, Wu & Li algorithm requires 10% less dominators than our algorithm with k$_{cds}$=1, and our algorithm with k$_{cds}$=2, 70% less. Besides, the backbone connectivity was measured (fig. 8(b)). The backbone is connected more than 96% of the time whatever the degree and the algorithms are. In sparse networks, the backbone connection decreases: possibilities for backbone reconnections are reduced. In sparse networks, the Wu & Li algorithm seems to suffer from inconsistencies because of delays of `hellos`. Since the radio topology is less redundant, the backbone is sometimes disconnected.



(a) Average backbone cardinality (average percentage of dominators)



(b) Average backbone connectivity (average percentage of nodes which respect the CDS properties)

Fig. 8. Impact of the density (average number of nodes per km^2): 40 nodes, 5 m.s^{-1}

6.6 Overhead

The control traffic generated by each node must remain limited: radio resources are scarce and must be spared. Details of the overhead generated by each node are reported in table 1. The overhead generated by Wu & Li consists only in `hellos` and was in consequence not reported here, since it corresponds to the same amount of `hellos` than CDCL. During the whole simulation, each node registers the amount of control packets sent per second, classified by packet type. We can remark that the overhead remains acceptable, even with many nodes in the network. In the same way, the `leader-hellos` present a very stable overhead: the solution to maintain the backbone connectivity is very scalable. The virtual structure of self-organization presents an higher overhead than Wu & Li but the absolute value of this overhead for both approaches seems for us negligible, justifying this approach. Moreover, our self-organization scheme presents a mixed structure of clusters and backbone forming a tree, with several levels of hierarchy. Thus, these assets are sufficient, according to us, to justify such an approach.

Finally, we can remark that the overhead to maintain the clusters is negligible: we can integrate a second level of hierarchy almost gratuitously, the cluster algorithms being very simple and integrated with the backbone structure.

Nb nodes	Hellos	CDS	Leader-Hellos	Clusters	Total
40	0.35	0.1	0.08	0.008	0.538
100	0.3	0.50	0.08	0.025	0.905

Table 1
Control traffic in packets sent per node per second: 40 nodes, 5 m.s^{-1}, degree of 10

6.7 Weight

Finally, we simulated the performances of a simple power-energy-saving solution to observe the impact of different weights on the dominators and clusterheads elections. A dominatee is allowed to sleep according to a probability which is inversely proportional to the number of its neighbors owning a lower weight. However, a node with a too low degree (<6) is not allowed to sleep so that a backbone reconnection is not disturbed. Besides, we simulated an heterogeneous network in introducing a node with a very low initial power-energy. Results are represented in the table 2. The backbone connectivity decreases since some dominatees sleeping, the backbone reconnections require more time. However, the clusters keep the same connectivity. The combined metric presents the lowest backbone connectivity, compared to the other metrics. However, the average sleeping time of the nodes is much higher. When a weight takes into account the residual energy of a node, the weakest nodes can efficiently save their energy, distributing the load to stronger nodes. Moreover, a weakest node sleeps on average 100% more with the combined metric. Thus, a weight based on energy seems to allow the weakest nodes to participate less actively, taking into account the heterogenity of an ad hoc network. Consequently, a combined metric integrating several criteria seems for us promising.

Weight	Backbone	Clusters	Sleeping time	
	connectivity	connectivity	Average	weakest node
Degree	93,6	99,1	70,7	65,2
Id	93,6	99	69,5	40
Mobility	94,3	99,2	71,2	59,5
Stability	93	99,2	80,4	128

Table 2
Power-Energy saving solution performances: 40 nodes, 5 m.s^{-1}, degree of 10

7 Conclusion and perspectives

We proposed here a self-organization structure useful in MANet and hybrid networks, based on both a virtual backbone and clusters. We presented distributed algorithms for both the construction and the maintenance since all terminals are mobile. The backbone can help to optimize the flooding in creating a subset of forwarding nodes, reducing the load on backbone clients, potentially weaker. Moreover, a first level of hierarchy is introduced in distinguishing backbone nodes from *normal* nodes. The backbone forms a tree, which could be useful for routing or localization protocols. Besides, clusters allow to introduce a second level of hierarchy, grouping nodes geographically close and electing one leader per cluster. Clusters can hide topology changes (a node can change its radio neighbors without changing the cluster topology) and could be useful to attribute addresses or for a hierarchical routing. Moreover, the cost (in time, control packets) to construct and maintain the clusters is almost negligible, justifying these additional *gratuitous* levels of hierarchy. Besides, information is shared among the backbone and clusters algorithms in order to reduce the overhead. Finally, the structure exhibits a good persistence, largely superior to the algorithm of Wu & Li. The persistence is for us a key property for a self-organization structure to be usable by a routing or localization protocol.

A routing protocol could advantageously take benefit from such a stability. Thus, we are currently testing a hierarchical routing protocol, which combines several classical routing protocols. This protocol exhibits interesting preliminary performances. Besides, the backbone forms an extension of classical wired backbones. Thus, localization and paging solutions may be proposed to offer a spontaneous multihops Internet connection to mobile clients. In the future, this self-organization structure will constitute a framework for protocols to extend their scalability and robustness.

References

[1] G. Mario, From battlefields to urban grids: New research challenges in ad hoc wireless networks, Pervasive and mobile computing 1 (1) (2005) 77–93.

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

[3] M. Fazio, M.and Villari, A. Puliafito, Ip address autoconfiguration in ad hoc networks: Design, implementation and measurements, Computer Networks 50 (7) (2006) 898–920.

[4] G. Pei, M. Gerla, Mobility management for hierarchical wireless networks, Mobile Networks and Applications 6 (2001) 331–337.

[5] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, in: International Conference on Mobile Computing and Networking (MOBICOM), ACM, Seattle, USA, 1999.

[6] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, D. J. Rosenkrantz, Simple heuristics for unit disk graphs, Networks 25 (1995) 59–68.

[7] K. M. Alzoubi, P.-J. Wan, O. Frieder, Distributed heuristics for connected dominating set in wireless ad hoc networks, IEEE ComSoc/KICS Journal of Communications and Networks, Special Issue on Innovations in Ad Hoc Mobile Pervasive Networks 4 (1) (2002) 22–29.

[8] M. Cardei, X. Cheng, X. Cheng, D.-Z. Du, Connected domination in ad hoc wireless networks, in: International Conference on Computer Science and Informatics (CSI), North Carolina, USA, 2002.

[9] S. Butenko, X. Cheng, D.-Z. Du, P. M. Pardalos, On the construction of virtual backbone for ad hoc wireless networks, in: Cooperative Control: Models, Applications and Algorithms, Vol. 1, Kluwer Academic Publishers, 2003, Ch. 3, pp. 43–54.

[10] P.-j. Wan, K. M. Alzoubi, O. Frieder, Distributed construction of connected dominating set in wireless ad hoc networks, Mobile Networks and Applications 9 (2) (2004) 141–149.

[11] P. Sinha, R. Sivakumar, V. Bharghavan, CEDAR: a core-extraction distributed ad hoc routing algorithm, in: INFOCOM, IEEE, New York, USA, 1999.

[12] B. Liang, Z. J. Haas, Virtual backbone generation and maintenance in ad hoc network mobility management, in: INFOCOM, IEEE, Tel-Aviv, Israel, 2000.

[13] J. Wu, H. Li, Dominating-set-based routing in ad hoc wireless networks, in: International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL'M), ACM, Seattle, USA, 1999.

[14] J. Wu, An enhanced approach to determine a small forward node set based on multipoint relays, in: Vehicular Technology Conference (VTC Fall), IEEE, Orlando, USA, 2003.

[15] J. Carle, D. Simplot-Ryl, Energy efficient area monitoring by sensor networks, IEEE Computer Magazine 37 (2) (2004) 40–46.

[16] I. Stojmenovic, M. Seddigh, J. Zunic, Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks, IEEE Transactions on Parallel and Distributed Systems 12 (12) (2001) 14–25.

[17] C. R. Lin, M. Gerla, Adaptive clustering for mobile wireless networks, IEEE Journal of Selected Areas in Communications 15 (7) (1997) 1265–1275.

[18] V. Kawadia, P. R. Kumar, Power control and clustering in ad hoc networks, in: INFOCOM, IEEE, San Francisco, USA, 2003.

[19] C.-C. Shen, C. Srisathapornphat, R. Liu, Z. Huang, C. Jaikaeo, E. L. Lloyd, CLTC: A cluster-based topology control framework for ad hoc networks, IEEE Transactions on Mobile Computing 3 (1) (2004) 18–32.

[20] M. Jiang, J. Li, Y. C. Tay, Cluster based routing protocol (CBRP), Internet draft version 01, IETF (July 1999).

[21] M. Gerla, J. T.-C. Tsai, Multicluster, mobile, multimedia radio network, Wireless Networks 1 (3) (1995) 255–265.

[22] M. R. Pearlman, Z. J. Haas, Determining the optimal configuration of the zone routing protocol, IEEE Journal on Selected Areas in Communications 17 (8) (1999) 1395–1414.

[23] A. Amis, R. Prakash, T. Vuong, D. Huynh, Max-min d-cluster formation in wireless ad hoc networks, in: INFOCOM, IEEE, Tel-Aviv, Israel, 1999.

[24] Y. Fernandess, D. Malkhi, K-clustering in wireless ad hoc networks, in: International Workshop on Principles of Mobile Computing (POMC), ACM Press, Toulouse, France, 2002.

[25] N. Malpani, J. L. Welch, N. Vaidya, Leader election algorithms for mobile ad hoc networks, in: workshop on Discrete algorithms and methods for mobile computing and communications (DIALM), ACM, New York, USA, 2000, pp. 96–103.

[26] F. Theoleyre, F. Valois, About the self-stabilization of a virtual topology for self-organization in ad hoc networks, Research Report 5650, INRIA (August 2005).

[27] J. Cartigny, F. Ingelrest, D. Simplot-Ryl, I. Stojmenovic, Localized LMST and RNG based minimum-energy broadcast protocols in ad hoc networks, Ad Hoc Networks 3 (1) (2005) 1–16.