Robustness and Reliability for Virtual Topologies in Wireless Multihop Access Networks

Fabrice Valois, Fabrice Theoleyre

Robustness and Reliability for Virtual Topologies in Wireless Multihop Access Networks

Fabrice Theoleyre, Fabrice Valois
CITI - INRIA ARES, INSA Lyon
21, Avenue Jean Capelle, 69621 Villeurbanne Cedex, France
Email: fabrice.theoleyre, fabrice.valois@insa-lyon.fr

***Abstract*— Mobile ad hoc networks (MANet) are a spontaneous collection of mobile terminals. Each node must collaborate in order to structure information exchange. An hybrid network is a MANet connected to Internet via an Access Point (AP). We propose to organize MANet and hybrid networks through a virtual topology. We consider a virtual topology as a hierarchical organization based on the integration of both backbone and clusters. Construction and maintenance procedures of such a virtual topology are detailed and deal with robustness and reliability issues. We present a proactive gratuitous maintenance for our backbone and a new maintenance algorithm for clusters presenting a reduced overhead. Moreover, this improved solution allows to integrate multiple APs in hybrid networks, deleting the previous single point of failure. A method to interconnect backbones is described, which is useful for many applications.**

I. Introduction

MANet (Mobile Ad Hoc Networks) are spontaneous wireless networks. Several terminals choose to collaborate with each other in order to purpose wireless multihops communications between any pair of nodes. The nodes must also support network functionalities like routing, localization or security. An *hybrid network* is a MANet connected to Internet through AP.

Ad hoc networks are essentially studied according to the routing problem. In these networks, all nodes are both clients and routers. Communications could also be multihops: some intermediaries nodes must be chosen to route packets from a source to a destination. Therefore, routing is to find the efficient forwarders, the optimal *path*. There exist mainly the proactive and the reactive approaches. In proactive one, a node have the whole knowledge of the network topology. Thus, it can directly compute optimal routes. The periodical flooding of topology packets creates an important overhead. Nevertheless the end-to-end delay and the latency are minimal, and new routes are instantaneously reported. Oppositely, the reactive solution, also called *on-demand*, exchanges control information only when a node wants to contact a new destination, or when the precedent route is broken. Hence, overhead is reduced but delays increase since the source must wait the round-trip time of the route request. Moreover, broadcasts in ad hoc networks present problem of redundancy and reliability [10], called the *broadcast storm*. Hybrid solutions try to combine assets of both proactive and reactive, like ZRP [11].

Ad hoc networks gather multiple specificities. Nodes are all independent and mobile, causing a very versatile environment. The network must also adapt itself to such dynamicity, reconstructing routes with an acceptable overhead. Besides, nodes have different capacities and behaviors. Some nodes will move quickly like cars, other slowly like pedestrians or itinerant users. We think that network organization must also reflect such an heterogeneity. In the same way, power energy saving is a key problem in MANet. Nodes must save their battery as much as possible but, some nodes can have much energy than others. It would be interesting to force nodes with an important energy level to act as network managers, and other nodes to be clients and to save their power energy in sleeping. Indeed, sleeping is the only efficient method to limit energy consumption [5]. Additionally, solutions must be scalable according to network cardinality: number of participants must grow without a brutal drop of performances. Other major functionalities remain to be proposed like security, localization, node addressing scheme…

We have proposed in [16] to construct and maintain both a backbone and clusters. The connected backbone allows to hierarchize MANet by creating dominator nodes (backbone members) which represent stronger nodes. Other terminals act as clients and can save their power energy using a sleeping mode. Thus, this backbone reflects the network heterogeneity. Moreover, this backbone allows an efficient flooding: control in-

formation could be flooded only to backbone nodes, not reaching clients. The number of transmissions is reduced and the load on weakest nodes is decreased. Moreover, in an hybrid network, the AP is the root of the backbone. It can also directly disseminate efficiently control information and route traffic between wireless and wired networks. In order to reduce the backbone cardinality, the distance between the dominators and their clients is parameterizable. We also construct $k_{cluster}$-clusters on our backbone. These clusters allow to implement hierarchical routing, with route requests flooded only to the backbone members, from a clusterhead to another. The distance from a node to its clusterhead being flexible, the cluster number is, as well as backbone cardinality, parameterizable.

However, an hybrid network creates a very versatile environment. It is also vital to maintain both backbone and clusters. We must maximize the structure connectivity, but minimize the overhead. In this study, we propose to improve our precedent algorithms for both backbone and clusters in order to reduce the overhead and improve the reactivity, i.e. we add some proactive features to collect more information for future backbone reconstruction. Our precedent solution was taking into account only one AP to wired world creating a single point of failure. Hence, we propose a solution integrating several APs, one backbone being constructed per AP. Nevertheless, these backbones are not connected, which might present some problems to flood information when path followed by flooding is an important property. We propose also a solution to interconnect these backbones, via *dominatee-connectors*.

The paper is organized as follows. First, related works on both backbones and clusters are explained in part II, with existing algorithms for both the construction and the maintenance. Part III is dedicated to the introduction of our algorithms for both the construction and the maintenance of the backbone and the cluster structures. These algorithms are an optimized version of our precedent work, improving the structure robustness and minimizing the overhead. In part IV, we introduce a solution integrating several APs, deleting the reliability problem for the interconnection with the wired world. Moreover, this solution integrates an interconnection of all built backbones. Then, in part V, we present the simulation results about the cardinality and connectivity structures, the delivery rate to wired network, the performances of floodings. Finally, part VI concludes about robust virtual structures and our contribution.

II. Related Work

Graph Theory is well suited to model MANet where a terminal is represented by a vertex, an edge existing between two vertices if and only if there exists a radio link between the two corresponding nodes. If we consider omnidirectional antennas identical for all nodes, the radio range is circular and constant. MANet could also be represented by a *Unit-Disk-Graph*: two vertices A and B have a common edge if and only if the two circles of radius 1, centered on A and B, intersect. *Unit-Disk-Graph* are well known graphs and many topology constructions use such a model.

A. Backbones

Multiple structures could model a backbone in MANet. The *Minimum Spanning Tree* (MST) is the most known backbone in the wired world. A MST is a tree connecting all nodes, minimizing the total cost of all edges used to constitute the backbone. There exist several algorithms for construction, like [7]. But MST has major drawbacks for a backbone: all nodes are backbone members and maintenance is not trivial. *K-Tree Cores* could model a backbone in graphs. A K-Tree Core is a tree, with k leaves, minimizing the average distance between any node and the nearest node of the K-Tree Core. [15] proposes a distributed algorithm for the construction of such a structure. This algorithm is divided into two major steps: the construction of a spanning tree, and the selection of the k adequate leaves. These *dominating leaves* are chosen according to a metric representing the distance saved when this leaf is selected, reported hierarchically from the leaves to the root. Such a structure represents well a backbone, and some nodes can act as backbone clients. Moreover, the number of leaves being flexible, the backbone cardinality is equally parameterizable. However, we think that k-Tree Cores presents important construction overhead and delays, and maintenance algorithms for these structures could be problematic. Moreover, the backbone members are not chosen according to robustness: this could generate some problems of structure instability.

Minimal Connected Dominating Set (MCDS) could model backbones in graphs. A MCDS is constituted by *dominator* nodes. The set of dominators is connected, and all other nodes, called *dominatees*, are neighbor of at least one dominator. Moreover, the cardinality of the MCDS must be minimal. MCDS is also a good representation of a backbone and present several assets:

- cardinality minimization;

- opposition dominator/dominatee reflecting behavior and capacity heterogeneity;
- election of dominator nodes based on adaptable weight;
- connected structure.

There exist several propositions to construct MCDS-approximations in graphs. Algorithms can often be divided into two parts ([1], [3], [4], [8], [14]). If a leader exists, it will initiate the backbone construction; else it could be elected. In the first step, dominating set is constructed: some nodes must be elected so that each dominatee is neighbor of at least one dominator. The node which owns the highest *weight* in its neighborhood elects itself as *dominator*, and all its neighbors choose it as their dominator father, becoming dominatees. The weight for elections is generally based either on node's degree ([3], [4], [8], [14]) or on identifier ([1]). The second step consists in the dominators interconnection, with a cardinality minimization. Such an interconnection is not trivial. ([3], [4]) propose to explore iteratively each dominatee, to choose the best candidates to become dominators instead of dominatees. The leader chooses to explore its dominatee neighbor d which owns the highest number of dominator neighbors. d becomes dominator, and forwards the exploration message to a dominator-neighbor D_1. The isolated dominators, neighbors of d, choose it as father. Then, D_1 is the new explorer. The procedure goes on until there exist no dominatee having an isolated dominator neighbor. [1] interconnects with an easier algorithm. Initially, only the leader is connected. It sends invitation packets to its 3-neighborhood to invite other dominators to connect themselves. Each unconnected dominator replies with a packet, forcing intermediate nodes to become dominators. This method forces more dominatees to become dominators, because of lack of particular optimization, but presents a lower construction delay.

CEDAR ([14]) proposes to interconnect dominators using virtual links. Each dominator sends `hello` packets to its 3-neighborhood. By this way, each dominator knows its virtual neighbors (dominators at most 3 hops far away), and can form a backbone, with potential loops. Packets which are flooded in backbone are encapsulated and then forwarded by dominatees forming the virtual links. The backbone doesn't form a tree and the cardinality is not optimal. Moreover, virtual links could become sub-optimal in a mobile environment, and we think that overhead of these `hello` flooded to 3-neighborhood could be important.

[8] proposes to create databases for mobility management. These databases are geographically distributed, and could approach the construction of a *Connected Dominating Set* (CDS). There exist 3 states: panic, normal and samaritan. Initially, all nodes are in panic mode. Each node floods to its r-neighborhood its number of panic or samaritan neighbors. The nodes with a local maximum are elected as database. Its r-neighbors become normal if they have no panic neighbor, else they become samaritan. The process stops when no more panic node exists. Each database sends `hello` to 2r+1 hops to interconnect itself to other databases, forming a backbone. The authors propose a maintenance method, which is rarely approached due to its complexity. The nodes without any database at r hops, become panic nodes and the process acts as in construction. The authors propose to merge databases which are less than d hops far away. The problem is to fine tune the parameter d. Connectivity is maintained with `hello` packets.

Many of these algorithms construct 1-MCDS and are not-well suited for k_{cds}-CDS construction. For example, the exploration method could present high construction delays. We think a k_{cds}-CDS is more suited to model a backbone: the distance between one node and the backbone being parameterizable, the cardinality is flexible according to the environment and application. Less nodes must participate to network management, more nodes are not essential and can sleep. More important, the maintenance of this structure is vital in a mobile environment. Nevertheless, only few propositions present a maintenance procedure. In [14], each dominator maintains virtual links but connections could be suboptimal and overhead important, `hellos` being flooded to $2k_{cds}$+1 hops. We think that these construction and maintenance must optimize robustness, connectivity, overhead... and not only minimize dominators cardinality. Therefore, our proposition [16] presents a solution in this way. But the present article presents important improvements for robustness, reliability and adaptability.

B. Clusters

Many articles propose to construct clusters to provide quality of service, hierarchically routing... The clustering consists of separating nodes in subsets to create homogeneous zones with a local leader called clusterhead. The maximum distance between any node and its clusterhead, the *radius* of the cluster, is usually a constant and is denoted as $k_{cluster}$.

Clustering is mainly based on an election. A clusterhead is a node which has the highest id ([9], [13]) or lowest mobility [12] inside its neighborhood. All its

neighbors without clusterhead choose the new elected node as chief. For maintenance, we can force the strongest node to remain clusterhead. Another approach is to optimize stability [9] in electing new nodes only when a node loses its clusterhead due to mobility or link breakdowns.

[2] constructs k-clusters, k being the maximal distance between a node and its clusterhead. In the first phase, during k rounds, each node forwards the highest id received during the previous round. The second phase, during k rounds too, propagates the lowest id heard at the end of the first phase. It is necessary to guarantee the cluster connectivity and to inform some nodes that they were elected. [6] constructs k-clusters but k being here the maximal number of members per cluster. They construct a tree and prune branches in order to limit the number of nodes per branch. The root of this branch becomes clusterhead. This method could be attractive in limiting the cluster cardinality, but no maintenance is given.

III. Topology Construction and Maintenance

We will precise here robustness improvements for our precedent algorithms. We want to construct a robust, adaptive, dynamic infrastructure in order to privilege strongest nodes, and give them a *federator* role. Each node starts to discover its k_{cds}-neighborhood. Next, we construct a k_{cds}-CDS, rooted at the AP, representing our backbone. We prefer to optimize robustness rather minimizing cardinality. We just tend to have less dominators, without guarantees. In parallel, we construct cluster of nodes on the backbone. The backbone and clusters are fully integrated, reducing overhead, sharing information for both construction and maintenance.

A. Construction

1) Neighborhood Discovering: Our virtual topology requires a k-neighborhood knowledge. Each node broadcasts `hello` packets with a TTL set to k. All receivers maintain an entry in their neighborhood table, decrement the TTL, and forward the packet if the TTL is not null.

2) Backbone construction: For the backbone construction, there exist 4 states :

- idle: node in initialization mode. It waits for an exterior solicitation for construction;
- dominatee: backbone client, having a dominator less than k_{cds} hops far away;
- active: node in election process to become dominator;
- dominator: backbone member.

When a node switches its state, it sends immediately a `state-message` to its k_{cds}-neighborhood. In an hybrid network, the AP can act as a natural leader. It will become the first dominator, and send its new state. The first step corresponds to the creation of a dominating set: each node must have at least one dominator at most k_{cds} hops far away. The strength of a node is represented by the *stability weight* described in [16]. The following rules are applied:

1) an idle/active node receiving a state message from a dominator becomes dominatee, and chooses the sender as father;
2) an idle node receiving a state message from a dominatee becomes active;
3) an active node becomes dominator when it owns during τ time the highest weight of its k_{CDS}-neighborhood of active nodes. τ must be sufficient to let the k_{CDS}-neighbors declare their potential new state. It has no father in this phase of CDS construction.

The second step of this algorithm consists in interconnecting all these dominators. Initially, only the leader is a connected dominator. A dominator recently connected sends a `join-message` with a TTL set to $2k_{cds}+1$. Indeed, because dominatees are at most k_{cds} hops far away from their dominator, the dominators can form a connected structure if we connect dominators $2k_{cds}+1$ hops far away. Dominatees forward a `join-message` with $TTL = t$ if they didn't forward more than x `join-messages` with $TTL \geq t$ ($x = 1$ in collision-free environments). Our algorithms are distributed and asynchronous. Hence, we force end of the first step in delaying all `join-messages` until the node is either dominatee or dominator.

An isolated dominator which receives a `join-message` sends a `join-reply`, following the inverse route. Each node in the path becomes dominator and sets its father to the next hop. The process reiterates until each node has a father. A dominator maintains the identity of its father, dominatees and sons (a son being a dominator which chose it as father).

3) Clusters Construction: Clusters and CDS structures are fully integrated. Only dominators participate to clusters construction: dominatees have automatically the clusterhead of their dominator. Hence, we reduce overhead for construction and force clusterheads to be backbone members. Clusterheads can also further directly flood information on backbone.

Each dominator must discover its ($k_{cluster}$-k_{cds})-virtual-neighborhood. A virtual neighbor is a son or a

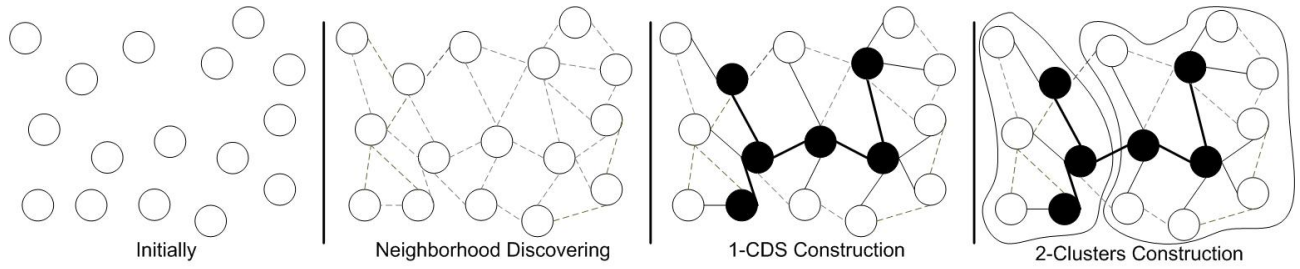

Fig. 1. Backbone and Clusters Construction

father in the backbone. It will send a `cluster-hello`, forwarded uniquely by virtual neighbors. The local strongest node of ($k_{cluster}$-k_{cds})-virtual-neighborhood is elected as clusterhead, and advertises its decision using a gratuitous `cluster-hello`. A dominator without clusterhead which receives such a message from a node which had chosen it as clusterhead, chooses the source as clusterhead. Thus, we force cluster connectivity. Because dominatees are at most k_{cds} hops far away from their dominator, we really construct $k_{cluster}$-clusters.

B. Maintenance

1) Backbone: Each node maintains continuously the information (identity, state, weight) about its father, and of eventual sons and dominatees inside its k_{cds}-neighbors. We have separated maintenance for dominatees and dominators :

a) Dominatees: A dominatee which loses its primary father (no one of the last x `hellos` received), will take the strongest among its *secondary fathers*. The strongest node is the node with the highest weight. Any dominator in the neighborhood-table, at most k_{cds} hops far away, is a secondary father.

If the node has no secondary father, it becomes active, and an election occurs like during construction, to choose the node(s) which must become dominator. These new dominators will then execute the maintenance reserved to dominators.

b) Dominators: The backbone must preserve its connectivity. Thus, the AP sends periodical `ap-hellos` with a strictly increasing *ap-hello id*. These packets are only forwarded by dominators when it comes from their father. Hence, a dominator can consider itself connected if it received this `ap-hello`. We consider a dominator disconnected if it missed the last x `ap-hellos`.

An isolated dominator must find a new father. If no *secondary-father* exists, it will engage the following procedure.This procedure combines broadcast and unicast in order to reduce overhead:

1) D sends in broadcast a `reconnect-request` with the last *ap-hello id* seen.
   The dominatees of D forward the packet in broadcast ; other dominatees forward the packet in unicast toward their dominator D_i.
2) D_i sends a `reconnect-reply` if it has an higher *ap-hello id*. The `reconnect-reply` follows the inverse route in unicast.
3) When D receives a `reconnect-reply`, it adds the next hop in the path as secondary father.

In order to improve maintenance, all dominators will store `ap-hellos`, received during the last x seconds, which own a strictly higher *ap-hello id* than the last `ap-hello` received from its father. The source is also a *secondary-father*: it is already alive and connected to the AP. If such a dominator was a descendant, the *ap-hello id* would have been equal. When a dominator is isolated, it chooses the secondary father with the highest weight. It advertises its new father directly if it comes from an `ap-hello`, else with a `reconnect-advert`. The `reconnect-advert` allows to force intermediate dominatees to become dominators in order to have a connected path between D and the source of the `reconnect-reply`. We add also here a proactive maintenance approach to our precedent algorithm. The reconnection can potentially occur without any overhead.

A dominator which has tried y unsuccessful reconnections will break its branch. It sends a `break-message` toward its sons, and reinitializes its state to idle. Its sons and dominatees reinitialize themselves in the same way. This branch forms also an idle area, waiting for an exterior signal for reconstruction. A connected dominator which owns an idle neighbor will send a `join-message`, acting as signal. The reconstruction is then similar to construction. Because an idle area can be exactly k_{cds}+1 hops far away from a dominator, a dominatee neighbor of this dominator and owning an idle node at k_{cds} hops will inform its dominator, forcing

it to send a `join-message`.

Our algorithm elects dominators. Hence, the backbone cardinality will increase if none antagonist process exists. A dominator with no dominatee at exactly k_{cds} hops, and no son, is useless. A useless dominator becomes dominatee, and sends a `useless-message`, forcing its dominatees to choose its father as new dominator.

2) Clusters: Only dominators participate to the maintenance. They choose one clusterhead and maintain the identity of the corresponding relay (their father or one of their sons), intermediary to contact their clusterhead. `hellos` contain the clusterhead identifier (C) of the source and the hop count toward it (H). When a `hello` comes from the relay, the dominator can also update its H variable, and check that it is already connected to its clusterhead C via other dominators. Such a procedure is possible because the backbone forms a tree and has no loop. In order to assure a quicker convergence, a node which changes its H or C variable sends immediately a gratuitous `hello`.

A node becoming dominator, which remarks that all its neighbors have already chosen a clusterhead, will directly execute the maintenance procedure, without cluster construction process. This can improve (re)construction delays when a node appears or an area is in reconstruction.

A dominator D_1 is disconnected if its relay is gone, dead, or it changed its state, or if it advertises another clusterhead, or if $H = k_{cluster} - k_{cds}$. If a dominator is disconnected, it sends a `cluster-reconnect-request` to its *virtual-neighborhood*. A *virtual-neighbor* D_2 can reply if it has a clusterhead at most $k_{cluster}$-k_{CDS}-1 hops far away and D_1 is not the relay of D_2 or the clusterhead of D_1 and D_2 are different. After $\max_{Cluster-reconnect}$ (=5 in simulations) unsuccessful attempts, an isolated dominator becomes its own clusterhead. A node which becomes clusterhead sends a gratuitous `hello` instantaneously to improve convergence.

A cluster is connected. Hence, a clusterhead is useless if no *virtual-neighbor* chooses it as clusterhead. Such a useless clusterhead will try to find another clusterhead to become *normal* dominator. The clusterhead of one of its virtual-neighbor can suit if it is less than $k_{cluster} - k_{cds}$ hops far away (computed with the H field).

IV. Multiple Gateways Integration

A. Multiple APs

Our precedent work took into account only one AP. Thus, it was constituting a single point of failure toward Internet. This solution being not acceptable, we present here an improvement allowing to integrate multiple APs. In order to further manage mobility, the AP must be a backbone member. Each AP must also be leader, and we want to construct a backbone per AP. We will logically cut off the hybrid network in zones according to APs.

For construction, we have several leaders. An isolated dominator could also have the choice of backbone to join in receiving different `join-messages` from different backbones. But, there is no fundamental difference. Since each node has only one father, we will construct one backbone per AP, with dominatees on backbones boundaries.

It is necessary for a node to identify its effective AP. Hence, the AP adds an *id-ap* field in its `ap-hellos`. By this way, each dominator knows directly its AP. Moreover, we add in `hellos` the *id-ap* field, so that each dominatee knows its AP too. The number of APs present in the hybrid network could be relatively small, the field could also own only a few bits. This overhead is hence low.

The maintenance for dominators presents some minor changes. On reception of an `ap-hello`, the source is a secondary father if the *ap-hello id* is higher than the id of the last `ap-hello` received, or if the *id-ap* is different: the dominator can reconnect itself to another backbone, and can realize a *soft handover* between 2 different APs. In the same way, an isolated dominator wanting to reconnect itself sends a `cds-reconnect` with the last *ap-hello id* received. Every connected dominator with an higher *ap-hello id* or a different *ap-id* can answer. A branch can reconnect itself to a new backbone, and run an handover for all its descendants. `cds-reconnect` and `cds-reply` must also contain an *id-ap* field.

The construction and maintenance for clusters are identical because the algorithms take into account only virtual-neighbors, which consists in dominators of the same backbone.

B. Backbones Interconnection

1) Motivations: We build several backbones. APs can communicate via wired network, but the backbones are not interconnected via wireless nodes. For a backbone flooding, the message will be broadcasted to the backbone. When the packet reaches the AP, it will be forwarded to other APs and backbones, via the wired network. For many applications, such a flooding is efficient and doesn't present some additional overhead. But when the followed path for flooding is important, separated backbones could present important disadvantages.

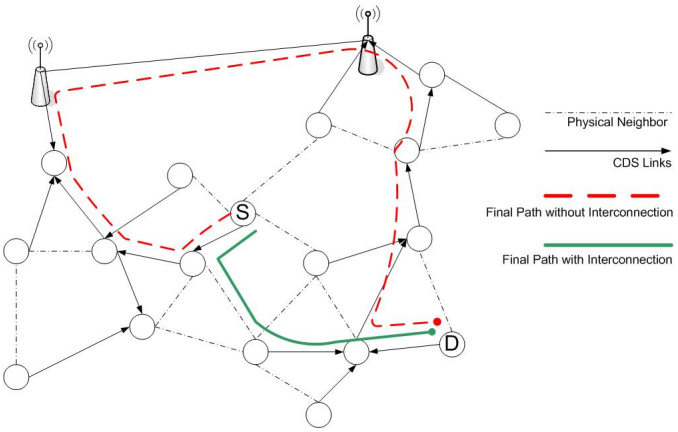

Fig. 2. Backbone Interconnection Asset for Routing

We suggest the following example: a source S sends a route request to its dominator, which will flood the backbone to find the destination. The route request will accumulate addresses of intermediate nodes during flooding. If the destination is in a different backbone, the path will in all cases pass through APs, even if the destination is not so far. The route is also clearly suboptimal. In fig. 2, the route with no interconnection (– –) is longer than the route with backbone interconnection (——).

2) Proposition: We must elect some dominatees (the *dominatee-connectors*) for the interconnection. However, a backbone flooding must limit the number of *dominatee-connectors*, because dominatees are potentially weak nodes. A dominatee must also wait the approbation of its dominator or its clusterhead, able to choose efficient connectors, because they have a more macroscopic view. However, if the clusterhead takes the decision, an high amount of data must be exchanged several hops along between clusterhead, dominatees and dominators in whole cluster. Thus, we have chosen the dominators as *dominatee-connectors* managers.

A dominatee knows that it represents a potential *dominatee-connector* in monitoring the *id-ap* declared by its neighbors in `hellos`. If this *id-ap* is different from its own *id-ap*, it must warn its dominator.

3) Backbone Flooding: We propose the following scheme for a backbone flooding. First, we implement a local routing table: each node knows the next hop toward each k_{cds}-neighbor. This information presents no additional overhead: S broadcasts already `hellos`. Each neighbor forwards this packet if the TTL is not null, inscribing its address, R, into the *Relay* field (already present in the packet). Finally, when a node receives the `hello`, it can choose R as relay toward S. This routing table could be integrated to the neighborhood table.

1 bit	4 bits	32 bits
Set Flag 1	AP Neighbor 1	Cluster ID 1
	...	
Set Flag N	AP Neighbor N	Cluster ID N

TABLE I
GATEWAY FIELDS IN `HELLO`

For a future routing solution, it would be interesting for a node N to know the cluster local topology. Such a topology consists in the list L_C of clusters neighbors of the cluster of N, and the list L_{GW} of neighbors of each of these clusters (such nodes are usually called gateways). With this information, a hierarchical routing solution could be set without high overhead. The route will present the list of cluster intermediaries instead of *normal nodes*. Each node in one cluster knows the neighboring cluster, and at least one gateway for each of these clusters. It can also forward the packet to the suited gateway, which will do the same thing if the destination is not yet reached. A gateway between 2 clusters must also advertise such an information to other members of its neighborhood. We choose to combine the information about cluster neighborhood and about *dominatee-connectors*.

Each node accumulates in `hellos` the list of clusterhead different from its own clusterhead declared by its 1-neighbors with the *id-ap* associated. Fields corresponding to gateways information in `hellos` have also the format described in tab. I. The field *Set Flag i* indicates if the i^{th} *cluster-neighbor* exists.

Each dominator can also register on `hello` reception the identity of its *dominatee-connectors*, and the relay toward this dominatee. The following steps of the algorithm are executed locally by each node to provide backbone flooding using interconnections (see Fig. 3 for a graphic illustration):

1) a) The dominator D_1 must flood a packet.
   b) D_1 searches for the list L_C of its connectors. It eliminates doubles in L_C, when 2 *dominatee-connectors* connect the same backbone, identified by *id-ap*. Then, D_1 searches the relays L_R toward each of the nodes in L_C.
   c) D_1 sends a packet to the multicast backbone address and L_R (with final destination L_C).
2) When a dominator D_2 receives the packet

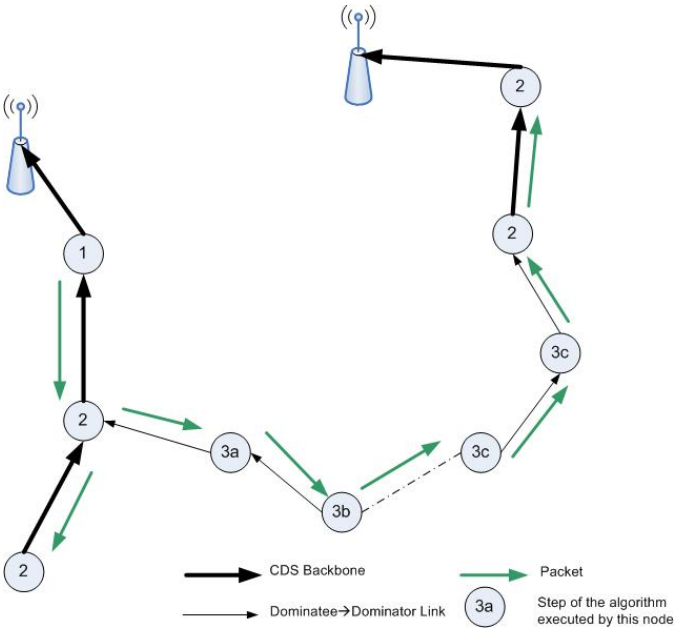

Fig. 3. Description of Flooding Method for interconnected Backbones

from D_1 it executes the same procedure as D_1.

3) a) When a dominatee recognizes its address as relay, but is not the final destination and has not the required *id-ap*, it forwards the packet toward the final destination (known with the neighborhood table).
   b) When a dominatee recognizes its address as final destination but has not the required *id-ap*, it forwards the packet to one of its neighbors having the required *id-ap*.
   c) When a dominatee has the required *id-ap*, it forwards the packet toward its dominator.

A dominator uses only one radio transmission to forward the packet to its virtual neighbors on backbone, and to all its *dominatee-connectors*. By this way, transmissions are reduced to minimum. Moreover, we can set up a mechanism of acknowledgments for backbone. Since the transmission is a flooding, all dominators must retransmit the packet. When a dominator N receives a packet with the backbone flooding multicast address, it adds in its *retransmission table* $n_{retransmission}$ entries corresponding to the packets that must be further forwarded by its virtual-neighbors. Each entry comprises the *packet id* and the *source address* of packet to route. Since each virtual-neighbor must forward the packet, each son and father of N has an entry in *retransmission table*. So $n_{retransmission} = nb_{sons} + 1$. When further, N receives the packet coming from an entry of the *retransmission table*, it deletes this entry. After a timeout $t_{retransmission}$, N retransmits not acknowledged packets older than $t_{retransmission}$ seconds. This passive acknowledgment mechanism has no overhead, except for the necessary retransmissions. The acknowledgments along the *dominatee-connectors* are implicit too, except for the explicit unicast acknowledgment sent by the final destination. We consider that such a mechanism could be efficient for reliable flooding.

Type	Intervall
`Hellos`	4
`Ap-Hellos`	1
`Clusterhead-Hellos`	1

TABLE II
TIME BETWEEN TWO HELLOS (IN SECONDS)

V. PERFORMANCE EVALUATION

A. Simulation

We used Opnet Modeler 8.1 to simulate the behavior of our solution. The nodes have a radio range of 300m and use the IEEE802.11b model of Opnet Modeler. They move on a rectangular surface according to the boundless mobility model, with an initial random position. APs only are fixed. We study the connection time, cardinality and persistence of our structures, according to mobility and network cardinality. The delivery ratio in unicast (ratio of number of packets sent and packets received) to the AP was measured to confirm CDS connection rate. The delivery ratio in multicast for a backbone flooding is the ratio of dominators which received the packet and the number of backbone members. All results are averaged on several simulations of 10 minutes, with only one flexible parameter by simulation. We consider as generic a speed of $5m.s^{-1}$, a backbone radius of 2, a cluster radius of 3, a network cardinality of 30 nodes, and a density of 8. Table II presents intervals between different `hellos`.

B. Results

1) Illustration: Figure 4 illustrates the structures after 2 minutes of maintenance. There are 7 dominators and 3 clusterheads. These dominators own an higher average weight than dominatees. We have a backbone, rooted at the Access Point, with identifier *1*, and this backbone is connected. In parallel, 2 of the clusterheads have several

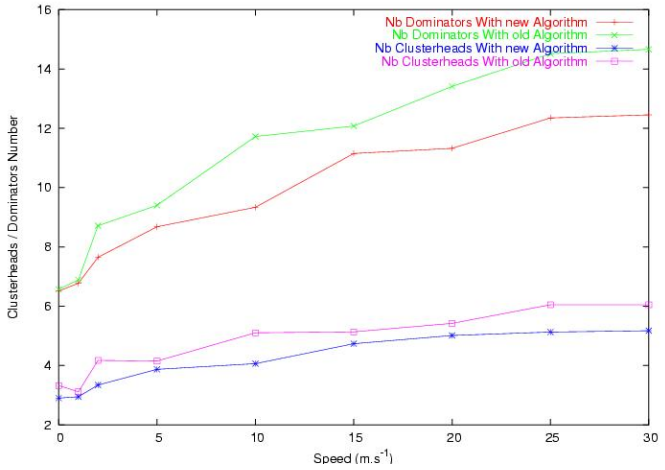

Fig. 5. Cardinality of virtual topology with new and old algorithms

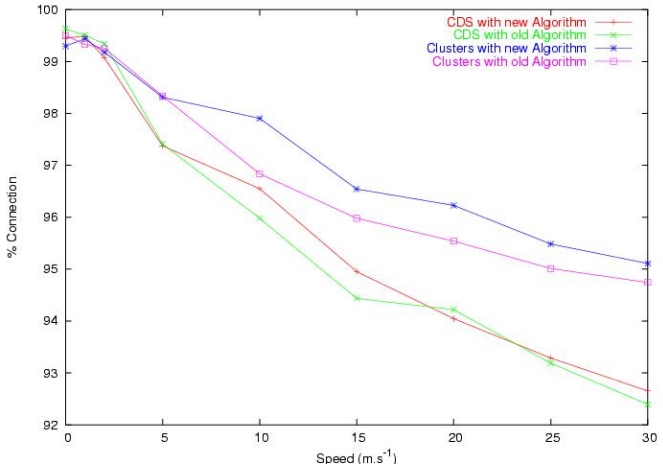

Fig. 6. Connectivity Percentage of virtual topology with new and old algorithms

clients. The other clusterhead serves only itself, the maintenance will force it to search for a new clusterhead and to become a *normal* node. We can check that we have a cluster radius of 3 hops.

2) Impact of Mobility: In this section, we study behavior of both our backbone and clusters according to nodes mobility. We compare our precedent algorithms from [16] with the algorithms presented here. First, we have less dominators and clusterheads with the new algorithms (fig.5). The structures seem more stable, and less dominators and clusterheads are necessary to maintain connectivity. We have more nodes acting as client. This may be interesting for power energy saving of dominatees, for mobility management (with less mobility areas)...

Then, we observe connectivity of our structures (fig.6). For the backbone, a dominatee considers itself connected if it has a valid father, information being obtained with `hellos`. A dominator is connected if it has received one of the last three `ap-hellos`. For clusters, a dominator considers itself connected if it has a valid relay, the relay being the intermediary toward its clusterhead (information extracted from `hellos`). The connectivity of the backbone is similar for both algorithms with a small advantage for the algorithm with proactive maintenance with speeds up to $5m.s^{-1}$. Indeed, the proactive maintenance is interesting only when the environment is volatile and several reconstructions occur simultaneously, nodes sniffing the reconnection-information from other nodes. The cluster connection increase is significative for speeds superior to $5m.s^{-1}$. Finally, we simulated a connection between a node and the Internet (also the AP). The delivery rate is the ratio between the packets sent and

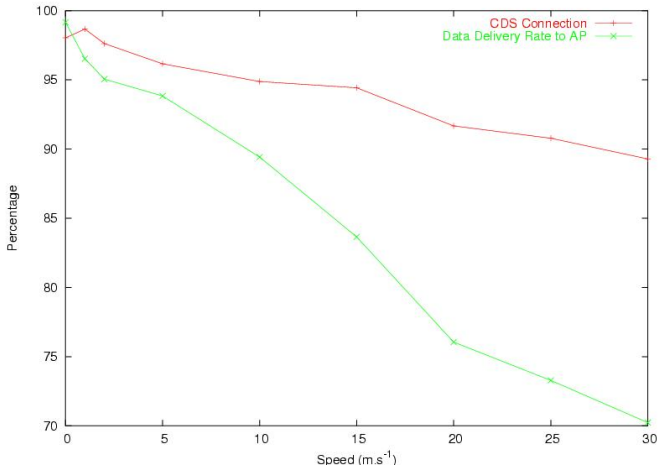

Fig. 7. Comparison between CDS Connection and Delivery Rate

the packets received. This delivery rate is lower than backbone connection for relatively high mobilities (cf. fig 7). We think that this is due to some collisions. However, the delivery rate is near 95% for speed of 5m.^{-1}.

We observe persistence of our clusterheads (fig.8), i.e. the number of clusterhead which change. With our new algorithm for cluster maintenance, the persistence increases due to the secondary fathers. A clusterhead remains clusterhead during 1,5 minutes with a speed of $10m.s^{-1}$. Our new algorithm outperforms the previous solution.

We study number of cluster-reconnections of both algorithms (fig.9). Our new algorithm is more stable, less reconnections occurring. The overhead is also reduced.

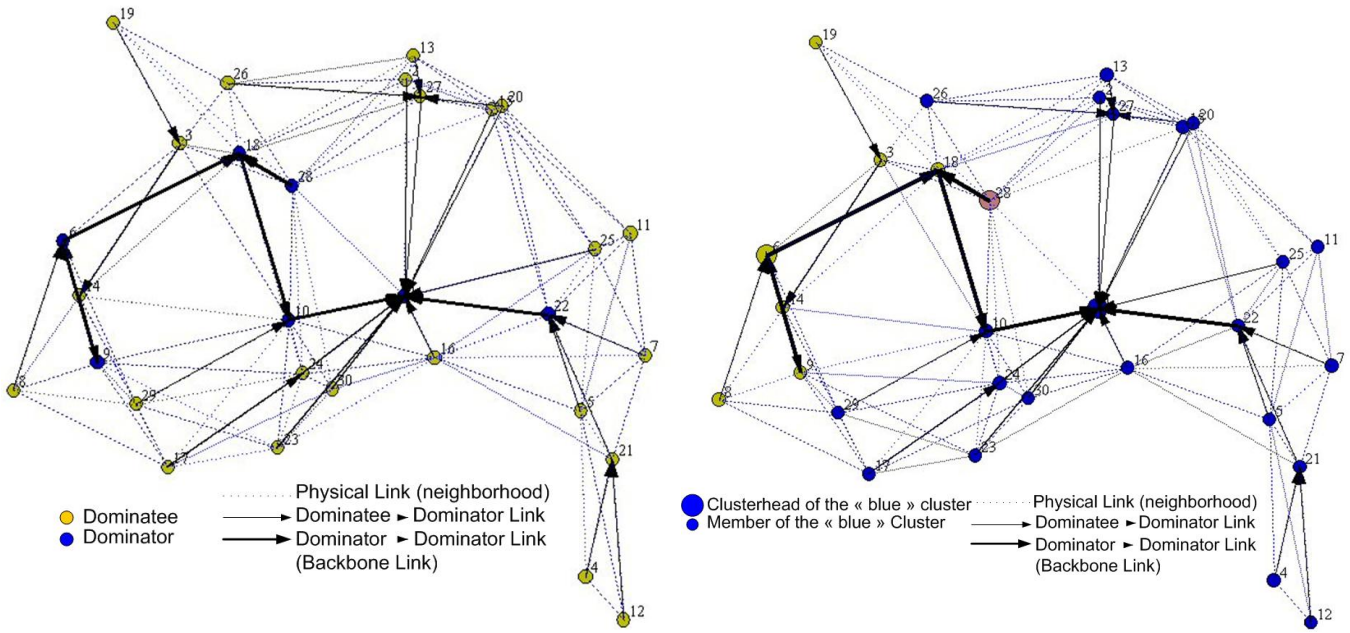

Fig. 4. Illustration of virtual topology construction

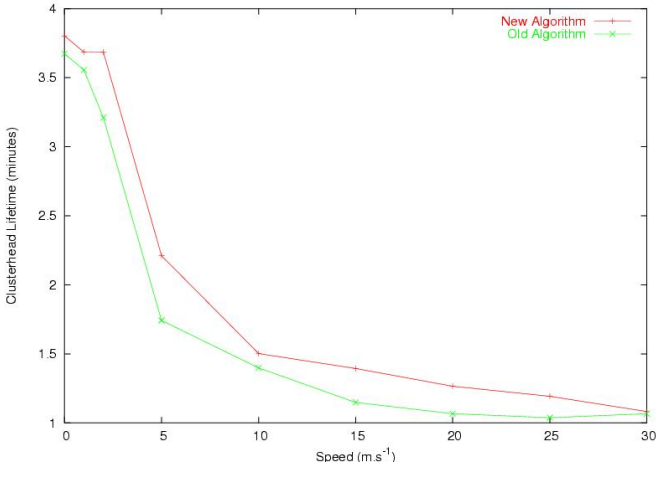

Fig. 8. Clusterhead Persistence with new and old algorithms

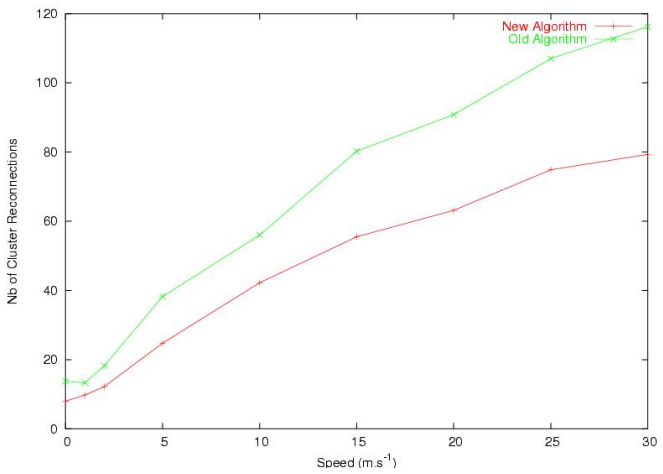

Fig. 9. Number of Cluster Reconnection Requests with new and old algorithms

Table III presents the overheads according to the different types of control packets, and corroborate this remark. The new algorithm for CDS maintenance presents a little less overhead, but perhaps not significative. However, the new cluster maintenance presents much less overhead (50%). Dominatees don't participate to cluster maintenance, so no overhead is induced for them.

3) Impact of number of nodes: In this section, we study impact of the network cardinality. When the number of participants grows, we have more clusterheads and dominators. Indeed, the network diameter increases

	New Algorithm		Old Algorithm	
	Dominator	Dominatee	Dominator	Dominatee
Hellos	2.38	2.08	2.37	2.01
CDS	0.35	0.20	0.37	0.20
Clusters	0.29	0	0.48	0
Total	3.02	2.29	3.22	2.31

TABLE III
CONTROL PACKETS SENT OR FORWARDED (IN PACKETS PER SECOND)

as degree is constant, the network is also less stable with more collisions, longer routes: we need more dominators and also clusterheads. This percentage of dominators is more stable when the number of participants exceeds a threshold. For the same reasons, the connection of our structures decreases while the number of nodes increases. However, even with 60 nodes, the CDS connection remains over 93%, and cluster connection over 95%. The differences between both algorithms are not significative, the speed of $5m.s^{-1}$ being perhaps not enough important to exerg a significant difference. The proactive maintenance approach is not decisive in such an environment.

4) Backbones Interconnection: In these simulations, we proceed with 40 mobile clients and 2 APs. We evaluate the performances of our backbones interconnection. First, we send one packet which must be distributed to all dominators, i.e. a *backbone flooding*. The acknowledgment mechanism described in section IV-B.3 was implemented. Then, the delivery rate of such a flooding is the percentage of dominators which receive this packet (Fig. 11). This delivery rate decreases when mobility increases, but remains over 88% whatever speed. In the same way, we record overhead for this flooding in Fig. 12. The overhead for dominators is largely superior than the overhead for dominatees. It increases naturally with mobility since many losses and retransmissions occur.

Finally, we study the number of interconnections between backbones. We count an interconnection when a dominator considers one of its dominatee as *dominatee-connector* toward another backbone. A virtual link between 2 backbones counts also for 2 interconnections (one for each extremity). We have 5 to 6 *dominatee-connectors*. When mobility is high, number of connectors decreases. We think that the network is more volatile, less candidates have the time to declare their role.

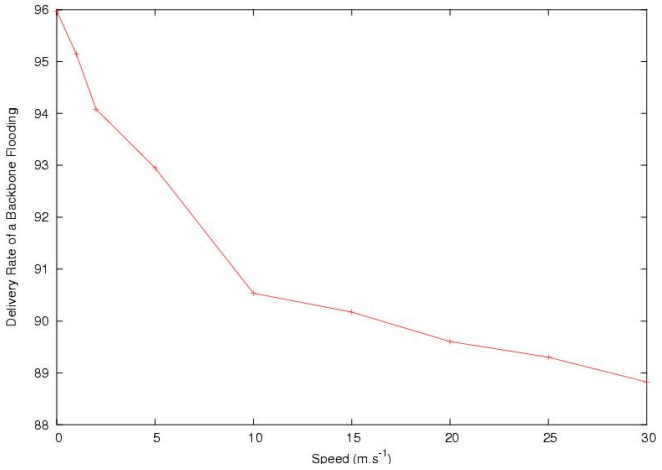

Fig. 11. Delivery rate for Backbone Floodings

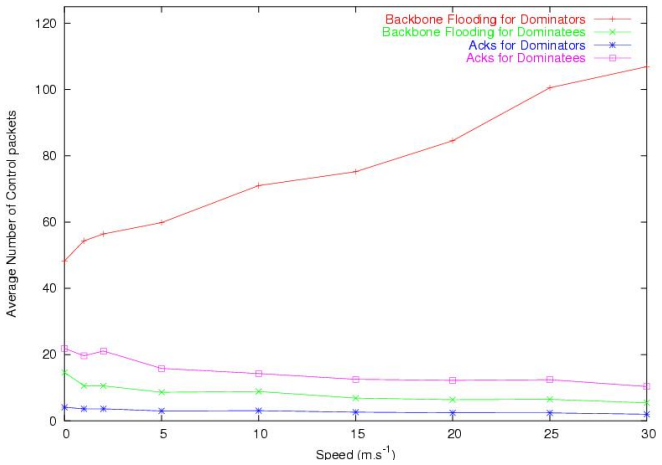

Fig. 12. Overhead for one Backbone Flooding

VI. Conclusion and Future Work

We propose and extend a virtual topology which is a combination of both backbone and clusters. We propose algorithms for both the construction and the maintenance, which is a key point in a dynamic environment. The structures are robust and present a reduced overhead. They allow to hide physical changes to higher levels. Virtual topology can also constitute a complete framework to develop further new services. The backbone can already organize efficiently a backbone flooding, but in the future, we could implement services like routing, localization or mobility management.

The structures present a reduced and parameterizable cardinality. Moreover, the structures are robust according to mobility, since connection rate and delivery rate are high even with high speeds. We improve the precedent algorithms in creating a proactive approach for CDS reconnection, and in changing the maintenance for clusters, in order to reduce overhead. The new algorithms are also more robust, which is a key point in a so versatile environment. It would be interesting to further investigate analytically the performances of our backbone and clusters construction and maintenance.

We propose a mechanism allowing to integrate several APs in the hybrid network, constructing one backbone per AP. Besides, we suggest a method to interconnect the different backbones for efficient flooding when the path is important. The delivery rate for such a backbone

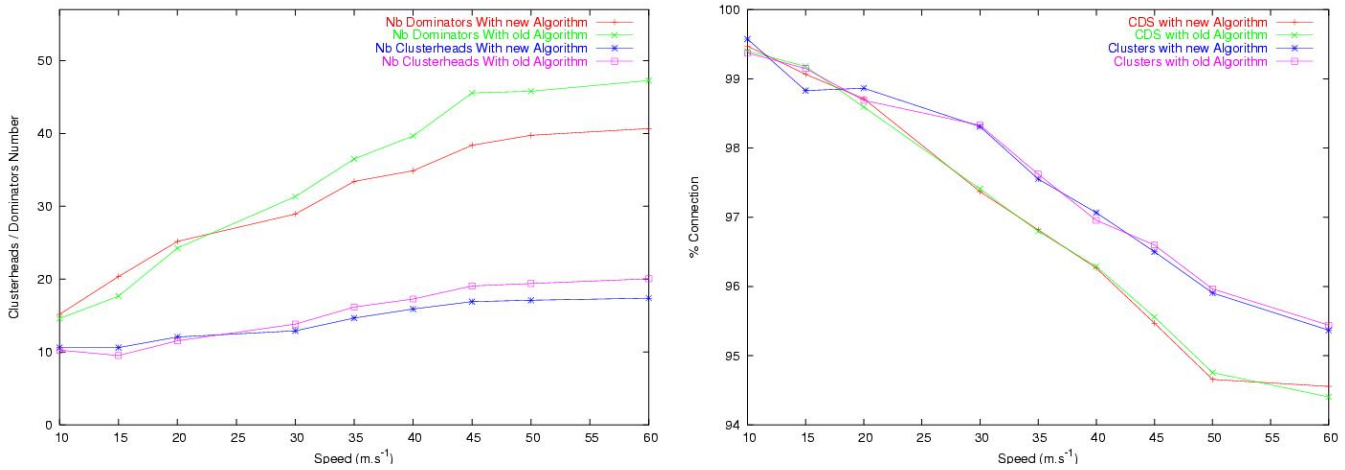

Fig. 10. Impact of Number of participants

flooding presents important performances, with a low number of transmissions and number of impacted nodes. Virtual topology appears to be a key issue in hybrid network to create a prolongation of cellular wireless networks. We must also study some additional features. Routing protocols and their impact on such a structure should be studied.

REFERENCES

[1] K. M. Alzoubi, P.-J. Wan, and 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):22–29, march 2002.

[2] Alan Amis, Ravi Prakash, Thai Vuong, and Dung Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings of IEEE INFOCOM*, pages 32–41, Tel-Aviv, Israel, March 1999. IEEE.

[3] Sergiy Butenko, Xiuzhen Cheng, Ding-Zhu Du, and Panos M. Pardalos. On the construction of virtual backbone for ad hoc wireless networks. In *Cooperative Control: Models, Applications and Algorithms*, volume 1 of *Cooperative Systems*, chapter 3, pages 43–54. Kluwer Academic Publishers, January 2003.

[4] Mihaela Cardei, Xiaoyan Cheng, Xiuzhen Cheng, and Ding-Zhu Du. Connected domination in ad hoc wireless networks. In *International Conference on Computer Science and Informatics*, North Carolina, USA, March 2002.

[5] Laura Marie Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *INFOCOM*, Anchorage, USA, April 2001. IEEE.

[6] Yaacov Fernandess and Dahlia Malkhi. K-clustering in wireless ad hoc networks. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 31–37, Toulouse, France, October 2002. ACM Press.

[7] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 5(1):66–77, January 1983.

[8] B. Liang and Z. J. Haas. Virtual backbone generation and maintenance in ad hoc network mobility management. In *INFOCOM'2000*, Tel-Aviv, Israel, March 2000. IEEE.

[9] Chunhung Richard Lin and Mario Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7):1265–1275, 1997.

[10] S.Y. Ni, Y.C. Tseng, Y.S. Chen, and J.P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Mobile Computing and Networking (MobiCom'99)*, pages 151–162, Seattle, USA, August 1999. ACM.

[11] Marc R. Pearlman and Zygmunt J. Haas. Determining the optimal configuration of the zone routing protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1395–1414, June 1999.

[12] Basu Prithwish, Khan Naved, and Thomas D. C. Little. A mobility based metric for clustering in mobile ad hoc networks. In ACM, editor, *International Conference on Mobile Computing and Networking*, pages 129–140, Roma, Italy, July 2001. ACM.

[13] Ahmed Safwat and Hossam Hassanein. Infrastructure-based routing in wireless mobile ad hoc networks. *The Journal of Computer Communications*, 25(3):210–224, 2002.

[14] Prasun Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan. Cedar: a core-extraction distributed ad hoc routing algorithm. In *INFOCOM'99*, pages 202–209, New York, USA, March 1999. IEEE.

[15] Saurabh Srivastava and R. K. Ghosh. Cluster based routing using a k-tree core backbone for mobile ad hoc networks. In *Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 14–23. ACM Press, 2002.

[16] Fabrice Theoleyre and Fabrice Valois. A virtual structure for mobility management in hybrid networks. In *Wireless Communications and Networking Conference (WCNC)*, Atlanta, USA, March 2004. IEEE.