HAL open science

A virtual structure for hybrid networks

Fabrice Theoleyre, Fabrice Valois

To cite this version:

Fabrice Theoleyre, Fabrice Valois. A virtual structure for hybrid networks. Wireless Communications and Networking Conference. WCNC, Mar 2004, Atlanta, United States. pp.1040-1045. hal-00371396

HAL Id: hal-00371396

https://hal.science/hal-00371396v1

Submitted on 27 Mar 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Virtual Structure for Hybrid Networks

Fabrice Theoleyre, Fabrice Valois
CITI-INRIA INSA Lyon
21, Avenue Jean Capelle, 69621 Villeurbanne, France
firstname.lastname@insa-lyon.fr

***Abstract*— Hybrid networks are heterogeneous networks merging both wireless and ad hoc nodes and where the interconnection to IP world is an important topic through gateways called AP (access point). Indeed, each node can be contacted and can contact another node in Internet. To reach that, architectures to support mobility management will be studied. The solutions inspired by wired networks are not particularly suited to hybrid networks. We propose to use a virtual dynamic infrastructure including both backbone and clusters. A backbone is suited to spare energy, optimize control traffic diffusion and hierarchize participants. The clusters are intended to create services areas and to handle particularly the mobility management. We present algorithms to both construct and maintain such structure. This dynamic topology is robust according to mobility, and is well suited to implement mobility management and localization procedure. Finally, the number of backbone members and clusters are completely parameterizable according to the environment.**

I. Introduction

Mobile Ad hoc networks (also named MANET) could be defined by *spontaneous wireless networks*, in which neither wireless nor wired infrastructures exists. All communicating objects are mobile and organize themselves to set up an efficient network. Hybrid networks are ad hoc networks connected to Internet, thanks to APs. The key issue in such network is the routing problem. The routing (proactive [7] and reactive [9]) is based on high amount of broadcasted control traffic flooding the network. But, in MANET, broadcasts constitute the *broadcast storm problem* [12]: redundancy of transmissions and reliability problems due to collisions. Hence, we propose to structure hybrid networks using a distributed approach with a virtual structure.

We propose to first create a virtual backbone [10]. Some nodes in the network will be elected to serve as masters. This election could take into account batteries longevity, mobility, etc., these parameters being adaptable. The creation of a dynamic backbone have some advantages. First, the backbone can spare control bandwidth for broadcast packets, all packets being sent only to backbone members. Moreover, we can add wireless-routers which are automatically integrated in the hybrid network. Finally, the backbone can help us to manage nodes mobility within wireless hybrid network. Using virtual backbone, we propose further to create services areas thanks to clusters. We can hence create a hierarchy in the hybrid network. This can help clusterhead deliver a service like mobility management and addresses attribution. Moreover, clusters introduce stability in the hybrid network by hiding some properties of the dynamic environment to higher levels. Both backbone and clusters are built and maintained simultaneously to reduce overhead and combine benefits.

Next, we will expose related work of creation of virtual structures in hybrid networks. Section III will expose our solution merging both backbone and clusters, with their construction and maintenance procedures, rarely described due to their complexity. Section IV will present simulations results: the stability of our structure, the persistence of masters, the impact of mobility and degree are discussed. Finally, section V will expose some perspectives.

II. Related Work

A. Backbone

A MANET can be modeled by a graph where the vertices are the communicating objects, and the edges the links between two nodes in communication. Backbone in such model could be represented by a *k_{mcds}-Minimal Connected Dominating Set* (k_{mcds}-MCDS). In this structure, it exists a maximal distance, k_{mcds}, between any node of the graph and the MCDS. The MCDS members are named *dominators* and the other nodes *dominatees*. The *dominators* give a connected structure. The cardinality of dominators must be minimalized. Many articles deal with the backbone construction using a 1-MCDS approximation. The construction can often be divided in two steps ([4], [6], [10]). If a leader exists, it initiates the construction, otherwise it could be elected. The first step elects some nodes as dominator, their neighbors becoming their dominatees. The election is often based on the lowest-id or on the highest-degree. The second step is the interconnection of these dominators, some dominatees becoming dominators. [6] explores each dominatee and choose firstly the dominatee which has the highest number of dominator neighbors. In [1], each connected dominator sends broadcasted invitations in order to invite other dominators to connect themselves via it, the leader being initially the only connected dominator. However, these algorithms propose a 1-MCDS construction and are not well suited to k_{mcds}-MCDS. The exploration method, for instance, requires a high delay before the end of MCDS construction. Furthermore, we think that a k_{mcds}-MCDS is more suitable in order to have less dominators in the network, and to allow more nodes to spare their energy in reducing their participation in the network control. In addition, the maintenance is a crucial procedure in a dynamic environment. The backbone members must be constantly updated to have a stable, efficient backbone. We think that a backbone should be generally constructed once, and maintained all the rest of the

time. Nevertheless, no article proposes a maintenance procedure for MCDS in MANET context. [10] introduces a maintenance, but the dominators are not connected to each other, the overhead generated is important (dominators can be (2k+1)-hops far), and virtual connection between dominators could be suboptimal when dominators move.

B. Clusters

Many articles propose to construct clusters in ad hoc network to allow quality of service or to provide hierarchical routing. Clustering is cutting the network in zones, with a master, named *clusterhead*, for each zone. It exists in a $k_{cluster}$-*cluster* a maximal distance, $k_{cluster}$, between one node and its clusterhead. Clustering tends generally to minimize the global number of clusters. The construction of clusters is usually based on an election. The node which is the *strongest* of its $k_{cluster}$-neighborhood during a round elects itself as *clusterhead*. This election can use the degree, the address [11], [13], a mobility metric [3]... Different approaches exist for the 1-cluster maintenance. The first is to always maintain as clusterhead the strongest node of a cluster, implicating many changes. The second [11] modifies cluster composition only when clusterhead is too far or down, with the creation of new clusters. There exists other propositions to construct $k_{cluster}$-clusters. In [2], the construction is in 2 phases. The first propagates the highest identifiers, and the second informs the clusterheads that they have been chosen. The authors of [8] propose to construct a tree and to prune the branches when they have $k_{cluster}$ members. Here, $k_{cluster}$-*clusters* means that a zone must contain at most $k_{cluster}$ members. But no maintenance, in both methods is explained.

III. Proposition

A. Motivations

We propose to construct a structure which combines the assets of both backbones and clusters (fig. 1). First, we construct a k_{mcds}-MCDS approximation, which represents our backbone. In fact, we don't try to minimize the cardinality of dominants: we try rather to limit this number. By this way, we construct more precisely a CDS. This virtual structure allows to better handle control traffic: nodes not in the backbone can spare their energy, the control information is flooded only to backbone's nodes. Secondly, we will integrate in this backbone a structure of $k_{cluster}$-clusters. These clusters will serve as services areas like mobility management (cluster can represent a localization) or addresses assignment. Moreover, we propose a maintenance procedure for these two merged structures, limiting the induced overhead.

B. Metric

A key issue for such structure is to be as stable as possible. Hence, we introduced a combined metric that we mean representative of the virtual structure stability. This stability weight depends in order of importance on:

- persistence: to force a master to be master as long as possible, and also support stability;
- relative mobility: to support nodes with a stable neighborhood, which can better and longer fulfill their role;
- degree: not too small to restrict collisions and enough important to minimize number of masters;
- energy: to penalize nodes with too low energy reserve which can soon die.

Metrics for virtual topology will be studied in depth in a future article, in order to simulate the behavior in different environments and to prove such a combined metric can be efficient.

C. Backbone

1) Construction: We propose to construct a CDS of variable diameter, k_{cds}. The construction of our backbone can be divided in two parts. The first step forces all nodes to choose a dominator. The AP of the hybrid network represents the natural leader and initiates the construction, becoming dominator. There exist 4 states:

- *dominator*: backbone member;
- *dominatee*: backbone client;
- *active*: in election state;
- *idle*: in initialization.

For neighborhood discovering, a node sends initially and periodically an `hello` with its identifier, state (*idle, active, dominatee, dominator*) and weight, propagated to k hops. Each node knows also its k-neighborhood. When a node changes its state, it advertises it to its neighborhood with a gratuitous `hello`. The next rules are applied:

- An *idle* node which receives an `hello` from a *dominator* becomes its *dominatee*, fixing the *dominator* as its father;
- An *idle* node which receives an `hello` from a *dominatee* becomes *active*;
- An *active* node which has the highest weight in its k_{cds}-neighborhood of active nodes during τ time becomes *dominator*.

This process is asynchronous, active nodes must also wait for the end of one step before becoming perhaps *dominator*. All candidates must have the time to declare their potential new state. The state message of an *active* node can be propagated during $k_{cds} - 1$ hops to prevent others that it participates to the election. Also, τ depends on the propagation delay of a message during $2 \cdot (k_{cds} - 1)$ hops. For the backbone's construction, a node must know its k_{cds}-neighborhood, so $k \geq k_{cds}$. The k-neighborhood discovering could be in the long term integrated to a routing protocol, cooperating to share the amount of information and reducing the overhead.

The second step is the interconnection of dominators. We connect the backbone thanks to the AP, acting as a gateway to Internet: the AP represents also our leader in the interconnection phase. Initially, only the leader is a connected dominator. A connected dominator sends a `join-message` to other non connected dominators, with a TTL of $2k_{cds}+1$ as in [1]. Indeed, a dominator is maximum $2k_{cds} + 1$ hops far from another dominator. A non connected dominator will answer with a `join-reply`, which follows automatically the inverse route,

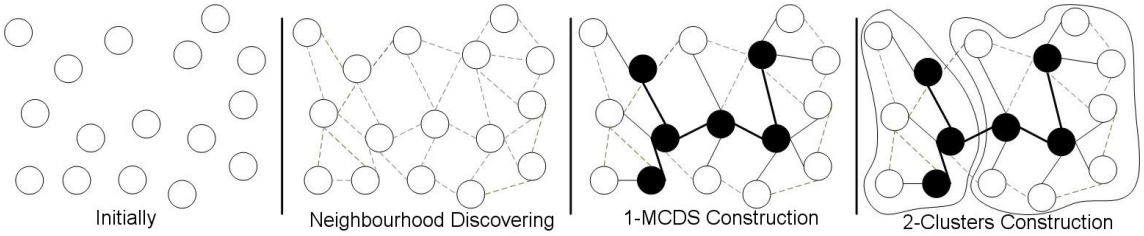

Fig. 1. Backbone and clusters combination

forcing all intermediary nodes to become backbone members. Each dominator maintains the identity of its father which is the next dominator in the CDS toward the root. The father of a dominator is necessarily one of its 1-neighbor in order to reflect the CDS connection. These new connected dominators will finally send a `join message`, and the process reiterates until total connection of hybrid network. Maintenance procedure begins as soon as construction is locally finished. Finally, each node owning a father, the k_{cds}-CDS is also a model of tree, with a leader as its root.

A node must know for the construction its k_{cds}-neighborhood before changing its active state into dominator state. But the neighborhood knowledge presents a cost. In consequence, we fixed the neighborhood discovering to k=k_{cds}.

2) Maintenance: The maintenance could be sum up in two elementary rules: each node must all the time own one and only one dominator, and the set of dominators must be connected. We also propose periodically `hellos` packets from the AP, with an unique incremental *ap-hello id*. Such `ap-hellos` packets are relayed by dominators, so a backbone member knows if it is already connected. In parallel, each dominator, like other nodes, sends `hellos` packets to advertise its presence.

We have separated maintenance procedures for dominatees and for dominators. A dominatee *d* which lost its dominator searches a new dominator in its k_{cds}-neighborhood-table. If a candidate exists, *d* chooses it as dominator. If it exists many candidates, it chooses the node with the strongest weight. If no candidate is present in the neighborhood-table, *d* becomes active. A new election occurs, like in construction mode: it avoids multiple neighbors, which lost their dominator, to become dominator simultaneously.

The maintenance process for dominators is more complex due to the necessity to maintain the connectivity of our CDS approximation:

1) A dominator *D* not yet connected (*x* `ap-hellos` missed) sends a `reconnect-request` with the id of the last `ap-hello` seen. If D receives further a `reconnect-reply`, D will act like with a `join-message`;
2) Only connected dominators can answer to a `reconnect-request` with a `reconnect-reply`. A connected dominator is a node which has received a newer `ap-hello` (higher *ap-hello id*);
3) A dominator D after *y* successive useless `reconnect-request` breaks its branch of k_{CDS}-CDS. All its sons and descendants receive its `break-message` and go to *idle state*, as in initialization. A reconstruction will follow.

After a `break-message`, an area is formed with only nodes in *idle* state, waiting for an exterior solicitation for reconstruction, as in construction mode. The next rules are also applied in order to reconstruct this zone:

1) If a connected dominator has an *idle* neighbor in its k_{cds}-neighborhood, it sends a `join-message` to initiate the area reconstruction, acting as a new temporary leader.
2) A dominatee neighbor of its dominator which has an exactly k_{cds} hops far *idle* neighbor advertises its dominator that it must send a `join-message` for the reconstruction.

We must create an antagonist procedure in order to avoid a constant increasing backbone. A dominator which has no dominatee at exactly k_{cds} hops and no son is useless: its father can serve this node and all its dominatees. This dominator sends also a `useless-message` forcing all its dominatees to choose its father as new dominator, the address of its father being contained in `hello packets`.

Hence, the maintenance allows all nodes to have a dominator, and a connected dominator set. We have also a virtual dynamic backbone, as stable as possible, using for its construction and maintenance a stability metric. This maintenance requires to know the k_{cds}-neighborhood, as in the construction process. The neighborhood discovering is also set up to k_{cds} hops during both construction and maintenance.

D. Clustering

1) Construction: The clusters are integrated to the backbone structure. Only backbone members participate to the construction of clusters in order to reduce the overhead during both construction and maintenance. Additionally, clusterheads are necessarily backbone members which is an advantage to further exchange control packets with other clusterheads via the backbone. Dominatees don't participate to the election and take automatically the clusterhead of their dominator. Each backbone member discovers its backbone member neighbors, creating a temporary $(k_{cluster} - k_{cds})$-cds-neighbors table, a *cds-neighbor* being its father or son in the k_{cds}-CDS. Thus, a 1-neighbor is not obligatory a *cds-neighbor*.

A dominatee is, by definition, k_{cds} far from its dominator. The radius of our clusters is also $k_{cluster}$. The `cluster-hellos` packets have the format of normal `hellos packets`, but can't be integrated in other *hello-messages* because of backbone utilization. This $(k_{cluster} - k_{cds})$-neighborhood discovering is also independent from the *normal* neighborhood-discovering. However, this type of packet is only used in the cluster construction, the overhead is also not too increased.

Each node maintains, during the cluster construction phase, the table of its $(k_{cluster} - k_{cds})$-cds-neighbors, and the list of these neighbors which have no clusterhead. For each round, if a node *N* has, during *t* time (message propagation time), the highest weight in this list, *N* is elected as clusterhead. All its $k_{cluster}$-neighbors having no clusterhead choose *N* as their new clusterhead. The process reiterates until each node in the hybrid network has a clusterhead. Maintenance process begins as soon as construction is locally finished.

2) Maintenance: The maintenance is a key point for both backbone and clusters. The maintenance must try to minimize the changes of cluster composition. We have introduced a `clusterhead-hello` packet, with the same role as for `ap-hellos` packet. The dominatees don't participate to the maintenance.

A backbone member which is always connected to its clusterhead (at least one of the last *x* `clusterhead-hellos` received) does nothing. A backbone member *B* which is not yet connected (none of the last *x* `clusterhead-hellos` received) broadcasts a `reconnect-request` on the virtual backbone with the id of the last `clusterhead-hello` seen. If *B* receives further a `reconnect-reply`, it chooses the source as new clusterhead. We force cluster connectivity with two rules: firstly, a node of the same cluster of the source of `reconnect-request` which has more recent information (higher *clusterhead-hello id*) sends a `reconnect-reply` else it relays the `reconnect-request` to other members of its own cluster; secondly, a node of a different cluster as source, but a neighbor of the source can send a `reconnect-reply` if it received one of the last *x* `clusterhead-hellos`.

We must propose a mechanism avoiding an increasing number of clusterheads, as in backbone maintenance for dominators. $C_{useless}$ is an useless clusterhead if none of its 1-neighbors declares $C_{useless}$ as their clusterhead. Because of cluster connectivity, no other nodes in the network can have chosen $C_{useless}$ as clusterhead. An useless clusterhead which receives a `clusterhead-hello` chooses the source as new clusterhead, losing its role of *master*.

IV. Performance Evaluation

A. Simulation

We used Opnet Modeler 8.1 to simulate the behavior of our solution. The radio level is represented by the standard IEEE 802.11b of Opnet Modeler. The random waypoint model was used to simulate mobility of each node [5]. A node moves on an area of 1900m $\times$ 1900m with a 300m radio range. We estimate a node disconnected if it missed the 3 last `ap-hellos` or `clusterhead-hellos`. Thus, we tend to over-estimate the disconnection time. We assume there is 30 nodes, a speed of $5m.s^{-1}$, a degree of 10, k_{cds}=2 and $k_{cluster}$=4, only one parameter being changed by simulation. The `ap-hellos` and `cluster-hellos` packets are sent every 2 seconds, and the `hellos` packet every 5 seconds.

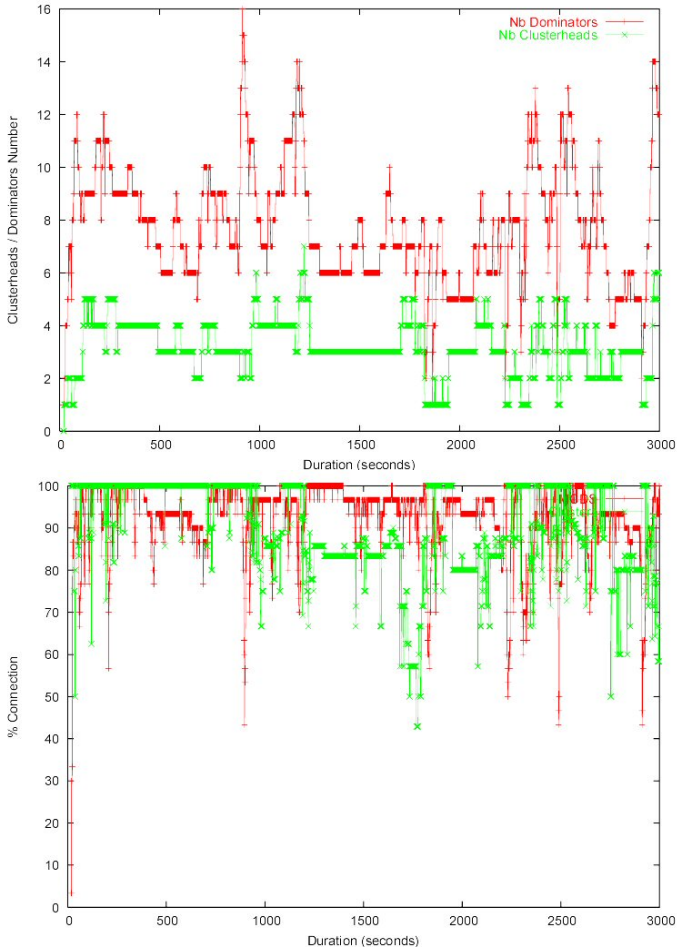

Fig. 2. Performances according to duration

B. Results

We simulated the influence of degree and number of participants, but these results are not presented because of the lack of place.

1) Duration: First, we observe the behavior of our structure during 1 hour. A dominator keeps its role during about 2.4 minutes, and a node changes of dominator every 2.8 minutes. A clusterhead remains clusterhead during about 1.4 minutes, and a node changes its clusterhead after 2 minutes. This duration is small to spare the clusterheads energy, but seems sufficient to allow clusterheads to serve as masters. There exists on average 8.6 dominators and 3.3 clusterheads. The nodes with higher weight tend to be chosen as dominators: with an average weight of 57, a dominator has a weight of 71 and a dominator with

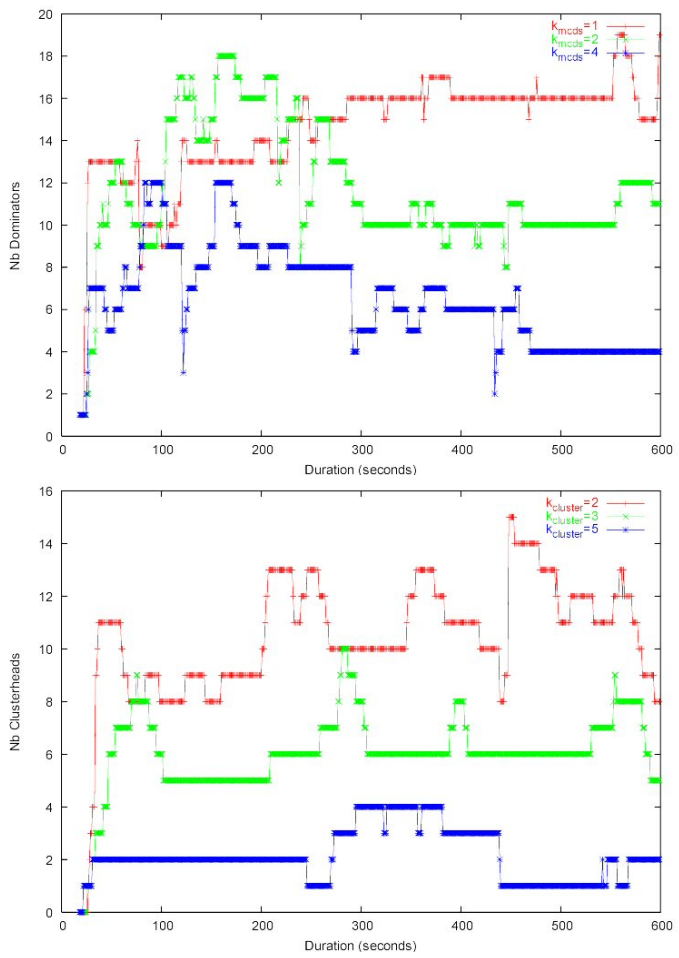

Fig. 3. Performances according to k_{cds} and $k_{cluster}$

dominatees a weight of 115. The CDS is relatively stable, the breaks corresponding to CDS reconstruction. The connection time are high (92.7% to CDS, 93.9% to clusterhead). In the future, there will exist data traffic, and the dominators will use these packets to maximize the connection time, having more information about their neighbors and ancestors.

2) CDS and Clusters diameters: We simulated the behavior of our solutions when backbone and cluster diameters vary. We observe that the backbone cardinality decreases when k_{cds} increases (fig. 3). Many nodes can spare their energy, but it exists sufficient backbone members to distribute the load. We observed during the simulations that the connection percentage falls when k_{cds} is too high. Indeed, many collisions occurred and the reconnection is more difficult, due to the average distance between two dominators. We observe that there exists less clusterheads when $k_{cluster}$ increases (fig. 3). We can notice the stability of clusterheads, which is an important property.

3) Mobility: We simulated the influence of mobility on the backbone and clusters. The speed varied between 0 and $35m.s^{-1}$. The cardinality of backbone set is relatively stable , the variations being acceptable. The clusterhead set cardinality is almost constant. For a high mobility of $15m.s^{-1}$, there

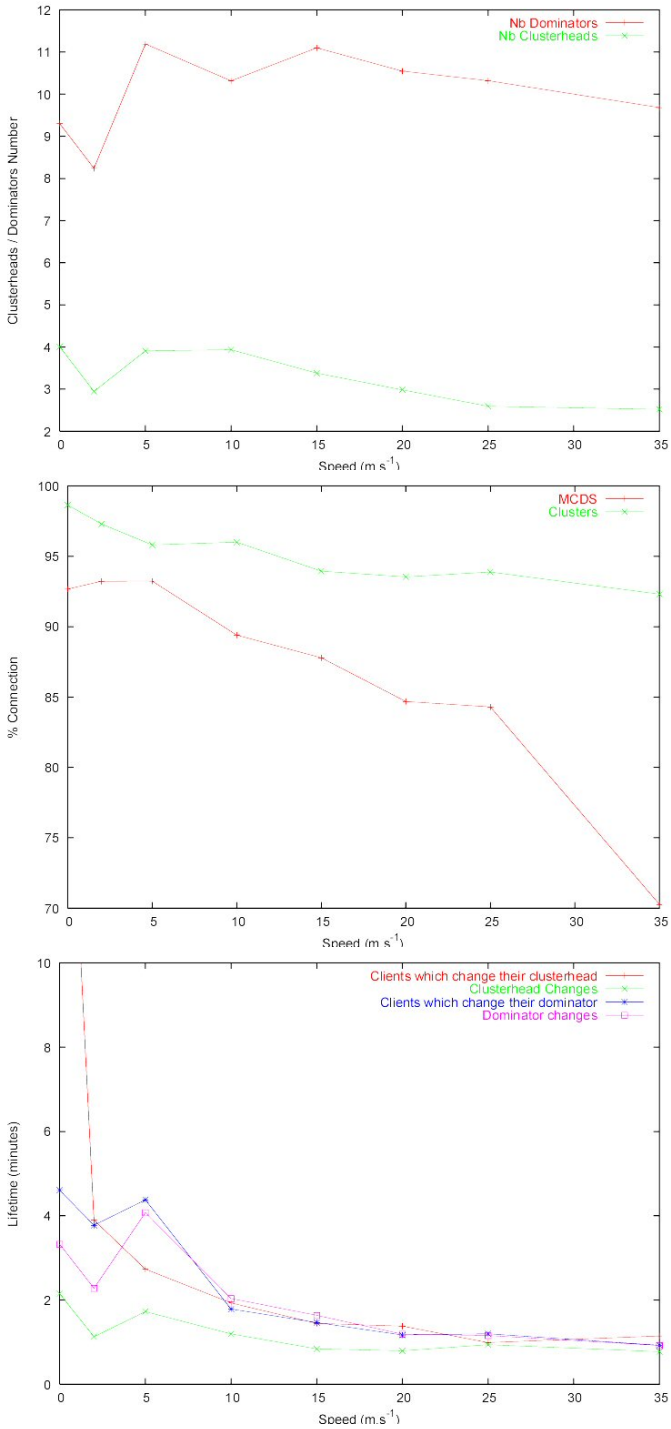

Fig. 4. Performances according to mobility

exist a maximum of 3.5 clusterheads (1/8 of nodes) and 11 dominators (1/3 of nodes). The connectivity to the clusterhead, over 92%, remains almost constant, even with high mobilities. The connectivity to backbone decreases, but remains over 90% for usual speeds of less than $10m.s^{-1}$. Finally, a clusterhead keeps its role of clusterhead during about 1 to 2 minutes and a node keeps its clusterhead during 1 to 50 minutes.

V. Conclusion and Future Work

We propose an integrated virtual and dynamic infrastructure to organize an hybrid network. This structure combines both backbone and clusters, with the introduction of a distributed algorithm for both construction and maintenance. We show the structure is stable and clusterheads keep their role during a long time. Moreover, our algorithms are robust since they present a very good behavior according to high mobilities. Finally, the cardinality of our structure is completely parameterizable according to the environment, the backbone and cluster diameters being parameters. A key contribution of this article is to present a solution organizing an hybrid network. It constitutes a weel-suited framework to implement network functionalities such as the mobility management, disregarding the physical topology. Such a structure will integrate multiple functionalities to mutualize the cost of construction and maintenance. Next step of this study will consist in improving mobility management with a fine or coarse localization solution. In parallel, the clustering potentially allows to implement a solution of sleeping mode so that normal nodes *sleep* and spare their energy. Besides, we propose a solution with a unique AP which serves as gateway. It represents a single point of failure, multiple AP are also required to have a certain redundancy. Finally it is important to further theoretically investigate our structures to find for example an upper bound of backbone cardinality and analytically prove the efficiency of our proposition.

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] Prithwish Basu, Naved Khan, and Thomas Little. A mobility based metric for clustering in mobile ad hoc networks. In *Proceedings of Distributed Computing Systems Workshop 2001*, Phoenix, Arizona, USA, April 2001. IEEE.

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

[5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, August 2002.

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

[7] Thomas Clausen, Philippe Jacquet, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized link state routing protocol. draft-ietf-manet-olsr-07.txt, November 2002.

[8] 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.

[9] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[10] 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.

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

[12] 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.

[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.