Comparison of Two Self-organization and Hierarchical Routing Protocols for Ad Hoc Networks

Betânia Steffen Abdallah Goncalves, Nathalie Mitton, Isabelle Guérin Lassous

Comparison of two Self-Organization and Hierarchical Routing Protocols for Ad Hoc Networks

Betânia Steffen Abdallah Goncalves[1], Nathalie Mitton[2], and Isabelle Guérin-Lassous[3]

[1]INSA de Lyon, [2]POPS USTL - INRIA - CNRS, Nathalie.Mitton@lifl.fr
[3] LIP - UCBL, Isabelle.Guerin-Lassous@ens-lyon.fr
FRANCE

Abstract. In this article, we compare two self-organization and hierarchical routing protocols for *ad hoc* networks. These two protocols apply the reverse approach from the classical one, since they use a reactive routing protocol inside the clusters and a proactive routing protocol between the clusters. We compare them regarding the cluster organization they provide and the routing that is then performed over it. This study gives an idea of the impact of the use of recursiveness and of the partition of the DHT on self-organization and hierarchical routing in ad hoc networks.
keywords: *ad hoc* networks, hierarchical routing, DHT, comparison, simulation.

1 Introduction

Ad hoc networks are composed of independent terminals that communicate via wireless interfaces. Every mobile node can move everywhere, disappear or appear in the network at any time. In order to allow communications between any pair of nodes that are not within communication range, intermediate nodes need to relay the messages. A routing protocol is thus required to provide routes between any pair of nodes in the network. *Ad hoc* networks have been becoming very popular these last years due to their easiness of use and deployment (no infrastructure needed). Their applications range from the network extension to spontaneous networks in case of natural disaster where the infrastructure has been totally destroyed, to the monitoring and the gathering of data with wireless sensor networks. Due to the dynamics of wireless networks and the terminal specificities (limited memory size and computing capacities), the routing protocols for fixed networks are not adapted. *Ad hoc* routing protocols proposed in the MANET working group at IETF[1] are all flat routing protocols, with no hierarchy. If flat routing protocols (proactive[2] and reactive[3] routing protocols) are quite effective on small and medium size networks, they are not suitable for large scale or very dense networks because of bandwidth and processing overheads they generate [18,8]. A common solution to this scalability problem is to introduce a hierarchical routing.

A hierarchical routing relies on a self-organization of the network in a specific partition, called *clustering*: the terminals are gathered into clusters according to some criteria, each cluster being identified by a special node called *cluster-head*. In this way,

[1] http://www.ietf.org/html.charters/manet-charter.html
[2] Nodes permanently keep a view of the topology. All routes are available as soon as needed.
[3] Routes are searched on-demand. Only active routes are maintained.

nodes store full information concerning nodes in their cluster and only partial information about other nodes. In addition to its scalable feature, such an organization also presents numerous advantages as to synchronize mobile nodes in a cluster or to attribute new service zones. Based on this partition, different routing policies are used in and between clusters:

(i) either proactive routing in the clusters and reactive routing between the clusters, which is the most common approach in the literature [5,7,16],

(ii) or reactive routing in the clusters and proactive routing between the clusters [17,11].

In this paper, we study the second approach of the hierarchical routing, *i.e.* using a reactive routing in the clusters and a proactive routing between the clusters. Such a hierarchical routing implies an indirect routing, *i.e.* the routing is performed in two steps: the look-up step that locates the destination node and the routing step to directly join it. Such an approach for hierarchical routing seems to us more scalable and more promising than the first one. Indeed, most of the clustering algorithms found in the literature provide a constant number of clusters when the intensity of nodes increases [4,9,15]. Thus, when the node density increases, there are still $O(n)$ nodes per cluster and using a proactive routing scheme in each cluster implies that each node still stores $O(n)$ routes, which is not more scalable than flat routing.

As far as we know, nowadays, in the literature, only two works propose this reverse hierarchical routing approach. They mainly differ in the self-organized structure they provide. The first one is called the density-based protocol [11] and uses a simple clustering structure. The second one is SAFARI [17] and uses a recursive hierarchical clustering structure. Both protocols use a DHT to perform the indirect routing.

In this paper, we compare SAFARI and the density-based protocol to analyze the impact of the use of recursiveness in the self-organization. Comparisons are lead regarding to the clustering structure provided and the quality of each indirect routing step (look-up and final routing). We will see that the main differences concern the stabilization time and the way the DHT has to be implemented over the resulting clustering structure. The remaining of this paper is organized as follows. Section 2 summarizes the indirect routing and the DHT principles. Section 3 briefly describes SAFARI and the "density-based" algorithm. Section 4 presents the simulation model. The cluster organizations are analyzed and compared in Section 5 while Section 6 provides a comparison of both routing steps and DHT utilization. Finally, in Section 7, we discuss some improvements and future works.

2 Indirect Routing and DHT

The goal of this paper is to compare two ways of applying a proactive routing protocol between clusters and a reactive routing protocol inside clusters. For such a hierarchical routing, an indirect routing is required. Indeed, a proactive routing protocol between clusters means that, knowing the cluster of the target node, the source node is able to route toward this cluster without any extra information. Then, once the message has reached the destination cluster, it can reach the target node thanks to a reactive routing protocol inside this cluster. Nevertheless, to be able to proceed like this, the source node has to know to which cluster the target belongs to. This routing process thus needs a

preliminary step which allows the source node to learn the location of the target node. Routing is thus performed in two steps: a first step called *look-up* which allows to learn the location of the destination and a second step which sends the message toward this location.

The most common tool used to performed an indirect routing is the Distributed Hash Table (DHT). It allows to share routing information over the nodes of the network. In this way, the required memory size on each node is minimized, which allows network scalability. A DHT uses a virtual addressing space $\mathcal{V}$. Each node u of the network is assigned a "virtual address" $VId_u \in \mathcal{V}$ as well as a partition of this virtual space. Let denote by I_u the partition of $\mathcal{V}$ assigned to node u. A $hash$ function associates the identifier of every node v to a virtual $key_v \in \mathcal{V}$ (one says that the id of v is *hashed* into the $key_v \in \mathcal{V}$). v will then register its location at node y such that $key_v \in I_y$. This allows to identify some rendezvous points: y is a rendezvous point for node v. The $hash$ function is known by every node of the network and may then be used by a source to identify the rendezvous point that stores the location of the target node.

The routing process used by the source to reach the rendezvous point and then the final node is either dependent or independent of the virtual space of the DHT. In DHT-independent routing schemes, the virtual address is not used for the routing operation. The nodes generally know their geographical coordinates, either absolute (by using a GPS for example) or relative, that is the location information they associate to the key. By performing $hash$(target), a node u gets the geographical coordinates of a rendezvous area $\mathcal{A}$. u then applies a geographical routing protocol to join a node v laying in $\mathcal{A}$ and that is aware of the geographical coordinates of the target node. From it, node u is able to reach the destination by performing a geographical routing once more [1,2,14].

In DHT-dependent routing schemes, the virtual space of the DHT is used not only for locating but also for routing toward the destination. The virtual address is dependent of the location. In this way, the consistency of the routing protocols rely on the consistent sharing of the virtual addressing space among all nodes in the network. The routing is performed over the virtual structure. In such scenarii, a node u performing $hash(w)$ gets the virtual address of the rendezvous point v. From it, u routes in the virtual space to v and obtains the virtual address of w. Thus u is able to reach w by routing in the virtual space again. The routing scheme used is generally a greedy routing: "Forward to the neighbor in the virtual space which virtual address matches the best the virtual address of the destination". This is for instance the case of Tribe [20] or L+ [3]. The main challenge here is to disseminate the partitions of the virtual space in such a manner that the paths in the virtual space are not much longer than the physical routes.

In most of the proposals, both indirect routing phases (look-up and final routing) are performed in the same way, either in the physical network (for DHT-independent routings) or in the virtual one (for DHT-dependent routings). SAFARI uses a DHT-dependent routing scheme. In the density-based approach, each routing step is performed in a different manner. The look-up is routed by using the virtual address of the rendezvous point (DHT-dependent) whereas the routing toward the final destination is performed over the physical network (DHT-independent).

There exist many ways to distribute the virtual addressing space over the nodes of the network, as it is strongly linked to the way the routing is attended to be performed.

Each proposal introduces its own way to distribute the virtual space. As we will see in Section 3, SAFARI and the density-based algorithm strongly differ in the way they distribute the partitions of the DHT addressing space. In SAFARI, the self-organization of the network and the distribution of the DHT intervals are performed simultaneously whereas in the density-based algorithm, both steps are distinct.

3 SAFARI and the density-based algorithm

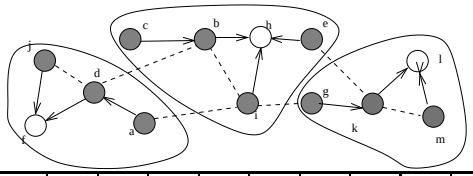
In this section, we describe the two protocols we have considered: the SAFARI protocol [17] and the density-based protocol [11]. We only give the main ideas and basis of the clustering, locating and routing steps of the two protocols. For more details, please refer to their respective references. Both protocols pursue the same goal to offer a network organization allowing a scalable routing. We will see that the two protocols strongly differ in the way they self-organize the network and in the DHT implementation. The density-based algorithm computes an organization with only one level and distributes the DHT addressing space afterward. The SAFARI heuristic proposes a recursive cluster hierarchy: nodes are grouped into clusters (cells or level-1 clusters), clusters into super-clusters (super-cells or level-2 clusters) and so on. This hierarchical structure is built simultaneously with the DHT implementation.

3.1 Density-based heuristic

Cluster formation: The cluster formation is based on a metric called "density", previously introduced in [10]. The density of a node u is the ratio between the number of links[4] in its neighborhood (links between u and its neighbors[5] and links between two neighbors of u) and the number of its neighbors. To compute clusters, each node locally computes its density value and periodically broadcasts it locally to its neighbors. Each node is thus able to compare its own density value to its neighbors' density values and decides by itself whether it joins one of them (the one with the highest density value) or it wins and elects itself as a cluster-head. Figure 1 illustrates the cluster formation. Node i has elected node h as its parent. In case of ties, the node with the lowest identity (denoted Id henceforth) wins. This is the case in Figure 1 for instance for nodes j and f which both have the same density value but as $Id(f) < Id(j)$, node j joins node f. A node's parent can also have joined another node and so on (node c joins node b which joins node h). A cluster extends itself until it reaches another cluster and the cluster radius is thus not defined *a priori*. Clusters are then identified by the Id of the cluster-head. By performing this joining process, we actually build a directed acyclic graph (DAG). The clustering process builds clusters by building a spanning forest of the network. One-level hierarchy is built.

Distributing the DHT addressing space [11]: In the density-based algorithm, the virtual space $\mathcal{V}$ is shared in each cluster among the branches of the trees. $\mathcal{V}$ is first shared by the cluster-heads between themselves and their children, proportionally to the size of

[4] There is a link as soon as two nodes are within transmission range.
[5] Two nodes are neighbors if there exists a link between them.



Node	a	b	c	d	e	f	g	h	i	j	k	l	m
Density	1	1.25	1	1.25	1	1.5	1	1.33	1.25	1.5	1.25	1.5	1.5

Fig. 1. Example of trees and clusters built by the density-based algorithm. Dashed links represent the wireless links which do not belong to the DAG, arrows represent a link in a tree directed from a node to its parent, cluster-heads appears in white.

the subtree of each of them. Then, each internal node recursively shares, between itself and its children, the partition given by its parent, and so one, till reaching the leaves of the trees. This step has a time complexity in $O(tree\_length)$. The tree length has been proved to be bounded by a low constant [13].

Indirect routing [11]: In the density-based protocol, the two steps of the indirect routing are not performed in the same way. The first step (look-up and registration) is routed in the DHT space, whereas the second step is performed in the physical space. The location of nodes is the identifier of their cluster. The $hash$ function returns one virtual address which exists in each cluster. Each node registers its location in each cluster, at nodes identified by the $hash$ function. In this way, the location of every node in the network is contained in each cluster. The look-up can thus be performed locally in a cluster. Routing in the DHT logical space is done by using an interval routing over the clustering tree. Interval routing allows to minimize information routing stored at each node. As the partition of the DHT in each tree of the DAG, the interval routing provides the shortest paths in the tree [19]. The second step of the indirect routing is performed in the physical space: if the source and the target node are in the same cluster, a reactive routing is performed by using an energy-efficient broadcasting operation described in [12]. Otherwise, the target cluster is reached by using the path of clusters returned by the inter-cluster proactive routing.

3.2 The SAFARI project

Cluster formation: SAFARI provides a l-level hierarchy of cells. It is built recursively, based on an automatic self-selection of nodes as cluster-heads (also called *drums*). At the initializing step, nodes have to wait during a random time before deciding to self-declare themselves as level-1 drums or not. A level-i drum may decide to up or down its level according to how far it is from other level-$(i+1)$ and level-(i) drums. If a level-i drum does not hear from any level-$(i+1)$ drum within a distance lower than a threshold depending on i, it self-declares itself as a level-$(i+1)$ drum. If two level-i drums are within a distance lower than another threshold also depending on i, only the greatest Id remains a level-i drum, the other one becoming a level-$(i-1)$ drum back. A level-i drum is also a level-j drum for all $0 \leq j \leq i$. With such a process, level-i cells are gathered into level-$(i+1)$ cells and so on. Plain nodes are seen as level-0 cells

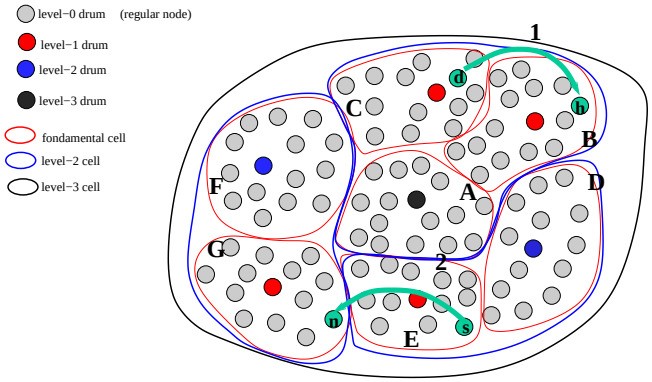


Fig. 2. Example of a SAFARI cluster organization. Fundamental cells have the following coordinates : A=[256 387 966], B=[102 387 966], C=[071 387 966], D=[308 659 966], E=[285 659 966], F=[003 741 966], G=[593 741 966]. Nodes S and D thus have the following DART: S =([256 387 966];[308 659 966];[285 659 966];[003 741 966]);D =([256 387 966];[102 387 966];[071 387 966];[308 659 966];[003 741 966])

and the unique highest level cell (level-l cell) gathers all the nodes of the network. Each level-i drum joins a level-$(i+1)$ drum. All level-i drums joining the same level-$(i+1)$ drum belong to the same level-$(i+1)$ cell. The radius of each cell is bounded, according to its level to D_i hops, i being the cell level. Each level-i drum periodically emits a packet called *beacon* every T_i, T_i depending on the level i. The higher the drum level, the longer the beacon emission period. These beacons are forwarded by all nodes within $h \times D_i$ hops, h being a constant. All nodes store all the beacons they forward in a Drum Ad Hoc Routing Table (DART). This hierarchical algorithm gives a unique ancestry for each node. Figure 2 gives a SAFARI cluster organization that has a recursive structure.

Distributing the DHT addressing space: Contrariwise of the density-based algorithm, SAFARI distributes the DHT virtual addressing space when building the clustering structure. Indeed, each node is assigned a coordinate based on the drum structure. If $COORD(d_i)$ is the coordinate of a level-i drum d_i and $PARENT(d_i)$ is the level-$(i+1)$ drum joined by d_i, we have: $COORD(d_i) = COORD(PARENT(d_i)).Rand$ where $Rand$ is a uniform random number. Level-0 drums (regular nodes) are leaves in the hierarchy and their coordinate is $COORD(d_0) = COORD(PARENT(d_0))$. Thus, all nodes in a fundamental cell have the same coordinate. The coordinates of the nodes form the logical Id space $\mathcal{V}$ of the DHT. $\mathcal{V}$ is shared over every node of the network while in the density-based, $\mathcal{V}$ is shared as many times as there are clusters.

Indirect routing: In SAFARI, both steps of the indirect routing, *i.e.* the look-up and the routing toward the final target, are performed the same way in the DHT space. The underlying idea of the look-up process is that generally, nodes communicate more with other nodes that are close to them. When performing $hash(v)$, the $hash$ function returns k different coordinates for each level i ($2 < i \leq l$), that is $k * (l-1)$ rendezvous points. These coordinates will be used by v to identify the nodes at which it has to register its location and by other nodes looking for node v. Node v will register its

coordinate k times in each level-i cell it belongs to, for i ranging from 2 to l (as every node has the same coordinate in the same level-1 cell, nodes do not register in their level-1 cell). As node coordinates are randomly chosen by the nodes themselves, the coordinates returned by the hash function are not necessary hold by a node. Therefore, nodes look for nodes in their DART which coordinate is the closest to the one returned by the $hash$ function. A node x looking for node v first sends a look-up request in the level-1 cells that belong to the same level-2 cell than itself by selecting the closest coordinate in its DART (let's say node u). If it does not find any right rendezvous point, it looks in level-2 cells, and so on, upping the level after each look-up failure. Otherwise, u does the same to forward the request of node x till reaching the fundamental cell of the rendezvous node r. There, the location is stored either by the drum of this cell, or by a node in the cell, reached thanks to a reactive routing inside the fundamental cell (it is not clear in SAFARI). If r knows the coordinate requested, it returns them to u which will then be able to reach x by the same way it had reached r. If r does not know the coordinate, x reiterates the look-up process at an upper level and so on.

4 Simulation Model

We use a simulator we developed in C language that assumes an ideal MAC layer, *i.e.* it does not consider interferences and packet collisions occurring at the MAC layer. Voluntarily, we did not use a network simulator which simulates a realist MAC layer because, as we wish to compare two network layer protocols, we do not want to be mistaken about their performances if any problem occurs at some lower layer. Nodes are randomly deployed in a 1×1 square using a Poisson Point Process (node positions are independent) with various levels of intensity λ (in such processes, λ represents the mean number of nodes per surface unit). Every node has the same transmission range R. There is an edge between two nodes if and only if their Euclidean distance is at most R (derived from the Unit Disk Graph model [6]). All the given results have a 95% - confidence interval. Both algorithms are compared over the same samples of node distribution.

Simulation parameters: In order to fairly compare both protocols, parameters have been tuned similarly for SAFARI and the density-based algorithms. In SAFARI, the cell radius D_i and the periods of beacon transmission T_i need to be determined for the lowest hierarchical level $i = 1$ and then, upper levels parameters are computed from it. Previous simulations of the density-based algorithm have shown that it provides a cluster radius between 3 and 4 hops [10]. Thus, in our simulations, we have fixed $D_1 = 3$ in SAFARI in order to get similar level-1 clusters in both protocols. Note that $D_1 = 3$ is also the value set by the authors of SAFARI in [17]. Still in order to compare both protocols, we assume that the packets exchanged in the density-based protocol are emitted every T_1 time units (period of level-1 beacons transmission in SAFARI). In SAFARI, at the initializing step, nodes have to wait during a random time drawn in $[0..X]s$ before deciding to self-declare themselves as a drum or not. For all these parameters, we used the same values as the authors of SAFARI, *i.e.* $T_1 = 2s$, $X = 5s$ and a SAFARI node registers its location $k = 3$ times at each level.

5 Cluster formation

In this section, we provide an analysis of the differences and similarities of both protocols regarding the cluster formation. As already mentioned, the main difference between both protocols is that the density-based algorithm provides a 1-level hierarchy whereas SAFARI builds a recursive hierarchy of l levels. Nodes in the density-based algorithm only use two-hop-away information (to compute their density value and then to elect their parent) while level-i drums in SAFARI need information in their DART, collected up to D_{i+1} hops. Thus, to build fundamental (level-1) clusters, nodes need to collect information up to D_1 hops. In order to fit different kinds of topologies and environments, the radius of clusters in the density-based algorithm is not set *a priori* whereas in SAFARI, the maximum radius D_i of level-i clusters has to be previously fixed ($1 \leq i \leq l$).

In the density-based algorithm, each new node entering the network checks its neighborhood, computes its density value and elects its parent. The algorithm stabilizes pretty quickly in a time proportional to the cluster radius [13]. In SAFARI, at the initializing step, nodes have to wait during a random time before deciding to self-declare themselves as drums or not. Moreover, the drum selection decision is based on the DART and thus on the beacons emitted by the drums. The stabilization time is thus linked to the initializing random back-off and also to the beacon frequency T_i (thus T_1 for the fundamental cells).

We have compared by simulation the clusters built by both protocols. The simulation model is described in Section 4. Note that, in both cases, because of the clustering algorithms, only the node degree impacts the cluster/drum characteristics of the clustering structures[6]. The network expansion only impacts the number of hierarchical levels built by SAFARI: between 3 and 4 levels for a 500-node topology with radius node set higher than 0.1 and between 2 and 3 levels otherwise. We will see later that this number of levels strongly impacts the stabilization time and the look-up performances in SAFARI.

Table 1 shows the different cluster characteristics we computed for different values of R that correspond to different values for the mean degree. In order to fairly compare these two protocols, all these data concern only the features of the level-1 clusters for SAFARI. The diameter of a cluster is the maximum number of hops between any pair of nodes of this cluster. Results show that clusters present similar average characteristics whatever the node degree (similar amount of clusters and diameters). However, even if the average values are similar, we can note that the density-based algorithm is much more stable as the clustering stabilization time and its standard deviation σ shows. Note that SAFARI does not stabilize at every time. This is due to the fact that, at the initialization step, the first drum that appears in the network is the one which random waiting time has expired the first. The cells are henceforth built according to the order of the waiting time periods of the nodes. As this waiting period is random, the clusters formed in the network may not be distributed in a good way, *i.e.* it is possible that nodes have to up and down their levels many times before stabilizing, which may take a long time. It is also possible that a node oscillates between two different levels

[6] Therefore, in topologies like grid or chain, clusters features remain the same.

$\overline{\delta}$	15.7		18.8		22.0	
	Density	SAFARI	Density	SAFARI	Density	SAFARI
# clusters	11.70	16.2	10.08	12.6	8.06	11.4
Diameter	4.99	4.67	5.52	4.62	5.50	4.76
Clustering stabilization time	5.27	107.67	5.34	113.41	5.33	91.95
$\sigma(Clustering\ stabilization\ time)$	0.63	132.41	0.74	135.56	0.85	123.69
DHT stabilization time	11.29	107.67	11.52	113.41	12.07	91.95
$\overline{\delta}$	25.1		28.3		31.4	
	Density	SAFARI	Density	SAFARI	Density	SAFARI
# clusters	7.03	9.10	6.15	8.10	5.57	7.40
Diameter	5.65	4.83	6.34	4.77	6.1	4.73
Clustering stabilization time	5.34	90.55	5.43	60.61	5.51	61.97
$\sigma(Clustering\ stabilization\ time)$	0.99	111.18	1.21	115.58	1.44	118.69
DHT stabilization time	12.02	90.55	12.29	60.61	12.51	61.97

Table 1. Some cluster characteristics for both metrics over a 500-node Poisson distribution and for different values of R (which gives different values of $\overline{\delta}$).

trying to respect all conditions of the level selection algorithm. As long as a node oscillates, the network never stabilizes. The clustering stabilization time is given in time units in Table 1. It represents the number of steps required before the cluster formation has stabilized. For SAFARI, results have been considered into computations only when the algorithm converges. We can notice that, for the density-based algorithm, the node intensity in the network does not impact the stabilization time. SAFARI is much longer to stabilize than the density-based heuristic (between 12 and 20 times longer) and that the clustering stabilization time of SAFARI fluctuates a lot as the standard deviation $\sigma(clustering\ stabilization\ time)$ shows.

The DHT stabilization time, also given in Table 1, represents the number of steps required before both the clustering structure has stabilized and the DHT addressing space has been shared between nodes. Note that for SAFARI, this corresponds to the clustering stabilization time as both operations are performed simultaneously. Contrarily, the density-based algorithm needs some more steps to distribute the virtual addressing space over each cluster. Nevertheless, note that this number of additional steps remains low and constant as it is equal to twice the tree depth which is bounded by a constant.

6 Look-up and Routing

In this section, we provide analysis and comparisons of the lookup and routing steps of both algorithms. In the density-based heuristic, the look-up is performed inside a cluster by performing an interval routing over the different branches of the clustering tree [11]. As the cluster diameter is generally low (as seen in Section 5), the look-up step is expected to need only a few hops to reach the rendezvous point, unlike SAFARI in which a rendezvous point may be everywhere in the network.

Moreover, if we assume a static network, the look-up in the density-based algorithm always succeeds, which may not be the case in SAFARI. This is due to the fact that level-i drums send their beacons to nodes in the level-i cells in the same level-$(i+1)$ cell than themselves. Thus, in a hierarchy with 3 or more levels, all nodes do not receive

$\overline{\delta}$	15.7		18.8		22.0	
	Density	SAFARI	Density	SAFARI	Density	SAFARI
# look-up Requests	1	1.71	1	1.82	1	1.78
Look-up Success	100%	95.70%	100%	92.20%	100%	90.90%
Look-up length	2.96	14.94	3.07	12.56	3.15	10.68
Route Length	5.69	7.28	6.67	5.87	6.37	6.17
Global Route Length	11.61	37.16	12.81	30.99	12.67	27.53
$\overline{\delta}$	25.1		28.3		31.4	
	Density	SAFARI	Density	SAFARI	Density	SAFARI
# look-up Requests	1	1.79	1	1.58	1	1.54
Look-up Success	100%	85.80%	100%	90.50%	100%	91.00%
Look-up length	3.16	10.36	3.21	8.63	3.24	5.04
Route length	6.75	5.88	6.61	5.73	6.66	5.09
Global Route Length	13.07	26.60	13.03	22.99	13.14	15.17

Table 2. Comparison of SAFARI and the density-based algorithm for the routing steps.

all beacons from all drums and do not have the same information in their DART. When a node d (see Figure 2) wants to register, it hashes its own identifier and sends its registration request to the node u which coordinate is the closest to the one returned by the DHT. u will do the same to forward the request of node d till reaching the fundamental cell of a rendezvous node h. Thus, node h is the node reached from the DART of node d. But as nodes have different information in their DART, when a node s locating in another level-2 cell wants to find node d, it will reach an eventual rendezvous node n from its own DART which would not have been previously contacted by node d. Thus the look-up fails. The bigger the number of levels of the network, the greater the chances that a source does never find any node responsible for storing the coordinate of the destination node.

Table 2 gives several features of look-up and routing steps for both algorithms. As in SAFARI, if a look-up request fails, a node reiterates the look-up operation toward another potential rendezvous node at the upper level, we give the mean number of tries before a success (*i.e.* the average level a query has to visit before succeeding). This number is given only for the look-up steps that finally find the coordinate of the destination node. The mean number of needed look-ups and the look-up success ratio in SAFARI depend on the number of hierarchical levels SAFARI built, as explained before. If 2 levels are built, every look-up query succeeds, as every node has the same DART. The look-up route length is the mean number of hops a query has to do before reaching the rendezvous node. As the look-up queries are routed inside a cluster in the density-based algorithm and in the whole network in SAFARI, the look-up paths are obviously shorter in the density-based approach. The routes length is the path length of the message from the source to the destination in the second step of the indirect routing. The global route length is the total number of hops that a message from the source have to do before reaching the final destination. It is the sum of twice the look-up path length and the route length. Indeed, the query has to make a round trip before the source be able to route toward the destination. We can also note that the look-up paths in SAFARI are much longer than the paths in the second step of the indirect routing, which is not the case in our approach. Note that the route length also depends on the network expan-

	Density-based	SAFARI
Hierarchy	Simple	Recursive
Metric	density + ID	Random time + ID
Cluster radius	automatic	fixed
Level	fixed ($= 1$ for routing)	l levels (automatic)
Convergence	fast and ensured	variable and not ensured
Registration	Once in each cluster	$k * (l - 1)$ times in the network
Lookup Success Ratio	100%	Depends on the number of levels

Table 3. Comparison of SAFARI and the density-based algorithm.

sion. The more expanded the network, the higher number of hops in average between the source and the destination. For the same reasons, the look-up path in SAFARI also depends on the network diameter, but, as the clusters are based only on local information, they do not depend on the network diameter. Since the density-based algorithm routes look-up queries inside a cluster, the look-up paths in the density-based algorithm are the same whatever the network expansion. Thus, even if the look-up path is about half of the global route, this will not be the case when the network will grow as the look-up path length is constant. Then, in a very expanded network, the look-up path length will become negligible before the routing path length whereas in SAFARI, the look-up path length will remain important before the global route length. Finally, because of SAFARI stabilization problem, we did not run simulations for more expanded networks to verify this feature.

7 Conclusion

In this article, we have compared the two only protocols proposing a hierarchical routing protocol for *ad hoc* networks based on a proactive routing between clusters and a reactive routing in clusters. One of them presents a recursive hierarchical organization whereas the other one presents a single level hierarchy. These two algorithms have different features that greatly impact the performances of the built structure and of the routing. The recursive structure and the partition of the DHT in SAFARI seem to make the routing more complex and lead to low performances compared to a simple cluster organization and to a natural distribution of the hash table. We summarize in Table 3 the main differences between both protocols. As we noticed, both algorithms build equivalent clusters (level-1 clusters for SAFARI) in terms of diameter and size. Nevertheless, the SAFARI algorithm has shown stabilization failures as it can be very slow to converge because of oscillation of the levels of the drums. Moreover, the look-up proposed by SAFARI does not guarantee to find the coordinate of the destination node and, even when it succeeds, the look-up path is much longer than the physical path from the source to the destination, which implies latency, weakness and useless energy spent and may lead to scalability problems. At the opposite, the density-based algorithm presents a good behavior compared to SAFARI since it has been proved to stabilize in a low and constant time. Moreover, the look-up queries always succeed and the look-up path length remains low as queries are routed locally.

For future works, we intend to pursue the comparisons *i)* in term of message complexity *(ii)* by evaluating the performances of this hierarchical routing in an *ad hoc*

environment based on a more realistic MAC layer (*i.e.* that takes collision into account) and *iii)* by studying the behavior of each of the algorithm in the context of mobility. Nevertheless, since even in a static and ideal MAC environment, SAFARI presents several failures, we are not very optimistic. Finally, we also intend to compare the performances of this kind of hierarchical routing with a classical *ad hoc* flat routing protocols (like OLSR or AODV) and with a hierarchical routing using a proactive routing inside clusters and a reactive routing between clusters.

References

1. F. Araujo, L. Rodrigues, J. Kaiser, L. Changling, and C. Mitidieri. CHR: A Distributed Hash Table for Wireless Ad Hoc Networks. In *DEBS'05*, Columbus, Ohio, USA, June 2005.
2. L. Blazevic, S. Giordano, and J.-Y. Le Boudec. Self-organized Terminode routing. *Journal of Cluster Computing*, 5(2), April 2002.
3. B. Chen and R. Morris. L+: Scalable landmark routing and address lookup for multi-hop wireless networks. Mit lcs technical report 837, MIT, March 2002.
4. G. Chen, F. Garcia, J. Solano, and I. Stojmenovic. Connectivity-based k-hop clustering in wireless networks. In *HICSS'02*, Hawaii, USA, January 2002.
5. Y. P. Chen, A. L. Liestman, and J. Liu. Clustering algorithms for *Ad Hoc* wireless networks. *Ad Hoc and Sensor Networks*, 2004.
6. B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3):165–177, 1990.
7. P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster based approach for routing in dynamic networks. In *ACM SIGCOMM*, pages 49–65. ACM, April 1997.
8. B.-J. Kwak, N.-O. Song, and L. Miller. On the scalability of *ad hoc* networks. *Communications Letters, IEEE*, 8:503– 505, 2004.
9. C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7):1265–1275, 1997.
10. N. Mitton, A. Busson, and E. Fleury. Self-organization in large scale ad hoc networks. In *MED-HOC-NET 04*, Bodrum, Turkey, June 2004.
11. N. Mitton and E. Fleury. Distributed node location in clustered multi-hop wireless networks. In *AINTEC'05*, Bangkok, Thailand, December 2005.
12. N. Mitton and E. Fleury. Efficient broadcasting in self-organizing multi-hop wireless network. In *Ad Hoc Now'05*, Cancun, Mexico, October 2005.
13. N. Mitton, E. Fleury, I. Guérin-Lassous, and S. Tixeuil. Self-stabilization in self-organized multihop wireless networks. In *WWAN'05*, Columbus, Ohio, USA, June 2005.
14. D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *Proceedings of GLOBECOM'01*, November 2001.
15. N. Nikaein, H. Labiod, and C. Bonnet. DDR-distributed dynamic routing algorithm for mobile ad hoc networks. In *Mobhihoc'00*, Boston, MA, USA, November, 20th 2000. ACM.
16. C. Perkins. *Ad hoc networking.* Addison-Wesley, 2001.
17. R. Riedi, P. Druschel, Y. C. Hu, D. B. Johnson, and R. Baraniuk. SAFARI: A self-organizing hierarchical architecture for scalable ad hoc networking networking. Research report TR04-433, Rice University, February 2005.
18. C. Santivanez, B. McDonald, I. Stavrakakis, and R. R. Ramanathan. On the scalability of ad hoc routing protocols. In *INFOCOM*, New-York, USA, June 2002.
19. J. Van Leeuven and R. Tan. Interval routing. *The computer Journal*, 30:298–307, 1987.
20. A. C. Viana, M. Dias de Armorim, S. Fdida, and J. Ferreira de Rezende. Self-organization in spontaneous networks: the approach of DHT-based routing protocols. *Ad Hoc Networks Journal*, 2005.