Distributed Node Location in clustered multi-hop wireless networks.

Nathalie Mitton
CITI/ARES - INRIA
nathalie.mitton@insa-lyon.fr

Eric Fleury
CITI/ARES - INRIA
eric.fleury@inria.fr

keywords: localization, DHT, ad hoc, sensors, wireless, self-organization, clusters, scalability, interval routing.

In a wireless multi-hop network, we need to allow the communication between any pair of nodes even if they are not in transmission range of each other. For it, a routing protocol is needed to provide routes between them. But, flat routing protocols (reactive or proactive) do not scale well. Indeed, such routing protocols become ineffective for large scale wireless multi-hop networks, because of bandwidth (flooding of control messages), latency and processing overhead (routing table computation). A well-known solution to overcome this major drawback is to introduce a hierarchy in the network by grouping geographically close nodes into *clusters* in a distributed and local way. We can thus apply different routing schemes in and between clusters. We thus previously introduced a clustering protocol in [6]. The resulting clusters have been analyzed by theoretical and simulations approaches and have reveal to present better features than other existing ones ([6, 7, 8]. It self-stabilizes in a low and constant time and specifically offers a better stability over node mobility. Because of place limitation, we will not detail our clustering scheme in this paper. We just sum up, in Table 1, the significant properties of the clusters built by these heuristics which have motivated our localization approach. As we will see in the following, one of the most important feature is that the clustering algorithm builds trees even before building clusters (All nodes of a same tree then belong to the same cluster and the tree root becomes the cluster-head). We can also note in Table 1 that the mean tree depth is pretty low and close to the optimal we could expect which is the mean cluster-head eccentricity (The eccentricity of a node is the greater distance in number of hops between it and any other node in its cluster). That means that the routes in the trees from the cluster-head to any other node within its cluster are, of course not optimal, but not so far from it. Moreover, note that, in average, an internal node does not have a lot of children.

In most of the literature, a hierarchical routing means using a proactive routing protocol inside the clusters and a reactive one between the clusters [5, 10, 2, 9]. Nevertheless, our clustering algorithm provides a constant number of clusters when the node intensity increases. Thus, when applying such a routing politic, we would still have $O(n)$ nodes per cluster and the clusters would not help a lot for scalability. Therefore,

	500 nodes	700 nodes	900 nodes	1000nodes
# clusters/trees	11.76	11.45	11.02	10.80
Cluster diameter	4.99	5.5	6.34	6.1
Cluster-head eccentricity	3.01	3.37	3.19	3.23
Node eccentricity	3.70	3.84	3.84	3.84
Tree depth	3.27	3.33	3.43	3.51
Degree in the tree of non-leaf nodes	3.82	4.19	4.51	4.62

Table 1: Some clusters and clustering trees characteristics.

we propose to use the reverse approach *i.e.*, applying a reactive routing protocol inside the clusters and a proactive routing protocol between the clusters. Indeed, as the number of clusters is low and constant when the intensity of nodes increases, each cluster has $O(1)$ routes to maintain toward other ones. Moreover, as we can see in Table 1, the mean node eccentricity is also low and constant (between 3 and 4 hops). Thus, finding a route on-demand inside a cluster should not introduce too much latency. As far as we know, only the SAFARI project [11] has proposed such an approach.

To apply such a routing politic, a node u first needs to locate its correspondent v *i.e.* to know the cluster to which the node v belongs: $\mathcal{C}(v)$. Once it gets this information, u is able either to route toward $\mathcal{C}(v)$ if $\mathcal{C}(u)! = \mathcal{C}(v)$ either to request a route toward node v inside its cluster if $\mathcal{C}(u) = \mathcal{C}(v)$. So, we need a localization function which returns for a node u, the cluster $\mathcal{C}(u)$ to which it belongs.

Basis ideas:

A routing operation is referred as indirect when it is performed in two steps: *(i)* first **locate** the target and then *(ii)* **communicate** with the target by using the information obtained at the first step (See Figure 1). This allows the network to dissociate the location of a node from the location itself. With this approach, the routing information can be totally distributed, which is important for achieving scalability in large scale networks.

We propose to use such a routing scheme. We briefly detail here how we propose to operate for each step of the indirect routing: distributing the virtual space and routing in the virtual space.

We first need to address few questions: *(i)* what kind of location information do we need to disseminate (In Figure 1, what information does u expect?), *(ii)* how to disseminate (how does u know to ask w and how does v know it has to register on node w?), *(iii)* how to recuperate it (how to reach w?).

In our proposal, every node already detains an information about its relative location: its cluster identifier. As we suppose that the nodes do not have any geographical information neither absolute nor relative, this is the only information they have. Thus, we propose that the nodes register their cluster Id as location information. To perform the association between a node and the nodes which store its position, we use a distrib-

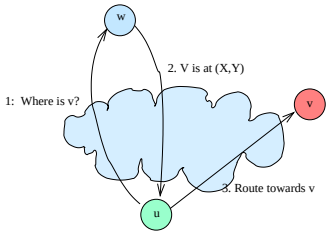

Figure 1: Indirect Routing: Node u needs to communicate with node v but does not have any clue about its position. It thus first needs to ask to node w where v is. w knows where v is since v regularly registers its position on w.

uted hash table (DHT). DHT provide a general mapping between any information and a location. They use a virtual addressing space $\mathcal{V}$. Partitions of this virtual space are assigned to nodes in the network. The idea is to use a $hash$ function to first distribute the node location information among rendezvous points. This same $hash$ function is also used by a source to identify the rendezvous point which stores the needed information. This $hash$ function is known by every node in the network. Each information is *hashed* into a key ($hash(v) = key_v \in \mathcal{V}$) of this virtual addressing space $\mathcal{V}$ and is stored on the node responsible for the partition of the virtual space this key belongs to. A node v is responsible for storing an information which key is comprised in the partition of $\mathcal{V}$ v has been assigned. Yet, in Figure 1, we have $hash(v) = Virtual\_Id(v)$. This is how *(i)* v knew where to register its location and *(ii)* how u knew where to ask the location of v. This operation in DHT-based systems which returns the node responsible for a wanted key *key* is called the lookup operation. For more details about the lookup operation, please refer to [1].

As already mentioned, we get a tree structure from our clustering scheme. We propose to partition a virtual space $\mathcal{V}$ in each tree and that each node registers d times on each cluster in order to add redundancy. In this way, when a node v looks for a node u, it just has to search the information in its cluster. As the node eccentricity is low (Table 1), the latency is reduced. Then, we use an interval routing routing scheme based on a tree interval labeling scheme to then route in the virtual space. Interval routing is very attractive by its simplicity. It was introduced in wired networks by Santoro and Khatib in [12] to reduce the size of the routing tables. It is a way of implementing routing schemes on arbitrary networks. It is based on representing the routing table stored at each node in a compact manner, by grouping the set of destination addresses that use the same output port into intervals of consecutive addresses. The main advantage of this scheme is the low memory requirements to store the routing: $O(\delta(u))$ on each node u. Many aspects of the efficiency of Interval Routing are studied in [13]. The routing is computed in a distributed way with the following algorithm: at each intermediate node x, the routing process ends if the destination y corresponds to x, otherwise, it is forwarded with the message through an edge labeled by a set I such that $y \in I$.

The interval labeling scheme (ILS) is the manner to assign interval to the edges of each node *i.e.*, how to distribute the virtual identifiers on the network, in order to

perform an efficient routing with routes as short as possible. Yet, the authors of [14] showed that undirected trees can support an interval routing scheme with shortest paths (in the tree) and with only one interval per output port. For it, the optimal labeling scheme is obtained by performing a depth-first-search ILS. In such routing schemes in wired networks, nodes have to store every interval for which each of its neighbors is responsible. Otherwise, if it does not know to which neighbor it has to forward the message, it has to query all of them consecutively, which would be costly. Therefore the size of the routing table is in $O(\delta(u))$ where $\delta(u)$ is the degree (# of neighbors) of node u. As $\delta(u)$ increases with the intensity of nodes, this size complexity may be problematic for achieving scalability. But in wireless environments, a transmission by each node can reach all nodes within radius distance from it. Thus the problematic is a bit different as querying only one neighbor (unicast transmission), is as costly as querying all neighbors (broadcast transmission). This is a property that we will use in order to limit the storage size required on each node. In our proposal, each node only stores its own interval (and not its neighbors'one). When a query is sent, as all neighbors receive it in any way, only the one concerned by it answers. For more details on Interval Routing features, please refer to [3, 4].

Summary:

To sum up, we propose to apply an indirect routing scheme over our clustered network by using DHT which associate each node Identifier to a virtual address of a space $\mathcal{V}$. The set of virtual addresses $\mathcal{V}$ is partitioned d times over the nodes of each cluster. Using a DHT, each node thus eventually find d virtual addresses and can register its cluster identity at the d nodes responsible for the partition owning its virtual address, and so in each cluster. Let's say we have c clusters. That means, as the clusters are homogeneous, that each node stores in average $d \times c$ location informations for other nodes. As c is constant for important node intensities and d is a low constant, each node finally stores $O(1)$ location informations.

When a node u wants to communicate with a node v, it uses the DHT to find out the virtual address of v: $hash(v) = key_v \in \mathcal{V}$. Then, by performing an interval routing over the virtual space $\mathcal{V}$ of its own tree, it reaches at least one node in its cluster responsible for storing the location of v, *i.e.* $\mathcal{C}(v)$ (where $\mathcal{C}(v)$ is the cluster of node v). As the intervals of the neighbors are not stored on the node, this one only stores its own interval. At last, it can join v, using a classical on-demand routing protocol if $\mathcal{C}(v) = \mathcal{C}(u)$, otherwise, it pro-actively joins $\mathcal{C}(v)$. Indeed, as the number of clusters is low and constant when the intensity of nodes increases, each cluster has $O(1)$ routes to maintain toward other ones.

Yet, this indirect routing algorithm does not require a great amount of storage resources as each node stores $O(1)$ informations in all.

References

[1] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking up data in P2P systems. *Communications of the ACM*, 46(2):43–48, February 2003.

[2] Y. P. Chen, A. L. Liestman, and J. Liu. Clustering algorithms for ad hoc wireless networks. *Ad Hoc and Sensor Networks*, 2004.

[3] P. Fraigniaud and C. Gavoille. Interval routing schemes. *Algorithmica*, pages 155–182, 1998.

[4] C. Gavoille. A survey on interval routing. *Theoretical Computer Science*, 245:217–253, 2000.

[5] 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.

[6] N. Mitton, A. Busson, and E. Fleury. Self-organization in large scale ad hoc networks. In *The Third Annual Mediterranean Ad Hoc Networking Workshop, MED-HOC-NET 04*, Bodrum, Turkey, June 2004.

[7] N. Mitton, A. Busson, and E. Fleury. Broadcasting in self-organizing wireless multi-hop network. Research report RR-5487, INRIA, February 2005.

[8] N. Mitton, E. Fleury, I. Gurin-Lassous, and S. Tixeuil. Self-stabilization in self-organized multihop wireless networks. In *2nd International Workshop on Wireless Ad Hoc Networking (WWAN'05)*, Columbus, Ohio, USA, June 2005.

[9] N. Nikaein, H. Labiod, and C. Bonnet. DDR-distributed dynamic routing algorithm for mobile ad hoc networks. In *1st ACM international symposium on mobile ad hoc routing and computing*, Boston, MA, USA, November, 20th 2000. ACM.

[10] C. Perkins. *Ad hoc networking.* Addison-Wesley, 2001.

[11] 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.

[12] N. Santoro and R. Khatib. Labeling and implicit routing in networks. *The computer Journal*, 28:5–8, 1985.

[13] S. S. Tse and F. C. Lau. More on the efficiency of interval routing. *The computer Journal*, 41:238–242, 1998.

[14] J. Van Leeuven and R. Tan. Interval routing. *The computer Journal*, 30:298–307, 1987.