Distributed algorithms for Constructing and Maintaining a Spanning Tree in a Mobile Ad hoc Network

[1]Hichem Megharbi and [2]Hamamache Kheddouci

[1] Université de Bourgogne
Laboratoire LE2I - Aile des Sciences de l'Ingenieur
BP 47870 - 21078 Dijon Cedex, France
hichem.megharbi@u-bourgogne.fr

[2] Université Claude Bernard Lyon1
Laboratoire PRISMa - Bâtiment Nautibus
843, Bd. du 11 novembre 1918,
69622 Villeurbanne Cedex France
hkheddou@bat710.univ-lyon1.fr

Abstract. A highly dynamic topology is a distinguishing feature and challenge of a mobile ad hoc network. Links between nodes are created and broken, as the nodes move within the network. This node mobility affects not only the source and/or destination, as in conventional network, but also intermediate nodes, due to the network's multihop nature. In this paper we propose a *virtuel spanning tree* in ad hoc networks as an analogue of the fixed communication infrastructure or distributed data structure in wired networks. We propose simple distributed algorithms where each node makes local decisions collectively guarantee the construction and the maintenance of a such tree in a mobile ad hoc network.

I. Introduction

Wireless ad hoc networks consists of a set of mobile devices (nodes) that communicate with each other via wireless links. The growing importance of ad hoc wireless networks can hardly be exaggerated, as portable wireless networks are now ubiquitous to grow in popularity and in capabilities. In such networks, all of the nodes are mobile and so the infrastructure for message communication must be self-organizing and adaptive. Building an infrastructure for ad hoc network that guarantees reliable communication is an important problem.

Several researchers have proposed construction of infrastructure in dynamic topology of a mobile ad hoc network. These infrastructures are often more adapted for the types of communication (routing, service discovery,…) and increase network longevity (optimize the use of the battery power and maintain network connectivity). L. Yena and K. Chib [12] propose a ring structure for ad hoc network. This structure is widely used for leader election [1,2,3,7], multicast [5,6] or mutual exclusion [9]. Rodoplu and Meng [8] propose an ingenious distributed topology control algorithm that guarantees connectivity of the entire network. Hu [4] describes a distributed Delaunay triangulation-based algorithm for choosing logical links and as a consequence carrying out topology control. In choosing these links he follows a few heuristic guidelines such as not exceeding an upper bound on the degree of each node and choosing links that create a regular and uniform graph structure. The CEDAR [10] algorithm establishes and maintains a routing infrastructure called core in ad hoc networks. An extension structure of core called k-core is used by S. Srivastava and R.K. Ghosh [11] which can be quite useful as a communication infrastructure in ad hoc networks.

In this paper we propose a *virtuel spanning tree* in ad hoc networks as an analogue of the fixed communication infrastructure or distributed data structure in wired networks. We propose simple distributed algorithms where each node makes local decisions collectively guarantee the construction and the maintenance of a such tree in a mobile ad hoc network.

The rest of the paper is organized as follows. In Section 2 we give a distributed algorithm for constructing a spanning tree in an ad hoc network. In Section 3, we present a distributed algorithm for maintaining the spanning tree in time. Section 4 concludes the paper.

II. Distributed algorithm for constructing a spanning tree

In this section we describe the procedures which are executed in each node in order to construct a spanning tree which covers all nodes of the ad hoc network at a given time. The algorithm is described in the following steps:

Initialisation step: each node is in *passive state*, which means that a node is not yet in a constructed subtree Fig 1.b.

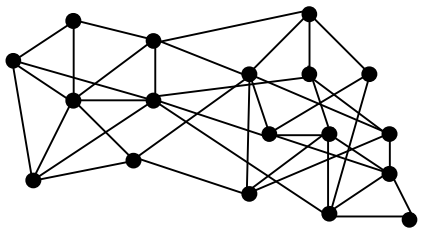
Construction step: Any passive node v becomes *active* either after given t times (and takes color C_v) Fig 1.c. or it is selected by one of its active neighbor v' Fig 1.d. (is this case v takes color of v', i.e. $C_v = C_{v'}$). Any active belongs to a subtree which maybe reduced to it (one node). All nodes of a subtree have the same color. Two distinct subtrees have different colors. Each active node sweeps its neighborhood and try to add new nodes to its subtree. So in its neighborhood, it can meet either a passive node or an other active node:

- invitation of passive neighbors to become active: Any active node detects these passive neighbors and invite them to become active and take its color Fig 1.d-f.. The corresponding links between the active node and the chosen passive neighbors are also actived. One can see that it is possible to limit the degree of the nodes in the constructed spanning tree. In this case, any active node must invite at most limited degree passive nodes to become neighbors in a spanning tree.

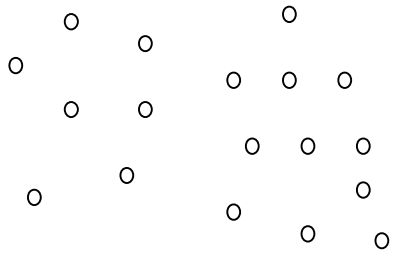
- subtrees fusion: Any active node v of a subtree T_v detects an other active node with the same colors that means the two nodes belong to the same subtree T_v. So v doesn't make anything. If v detects an active neighbor v' of another subtree $T_{v'}$ save *detection time* t and informs by flooding all nodes of T_v that it has detected v' at time t (v' will do the same processing than v in $T_{v'}$). After a given time, the active node of T_v which has the smallest detection time actives the link with the corresponding subtree. So we obtain by composition only one subtree. The color of the new subtree is given by the composition of the colors of the two subtrees. In order to avoid the construction of cycles during this operation, we active only one link each time Fig.1.g-i.

We illustrate this algorithm by the following example.

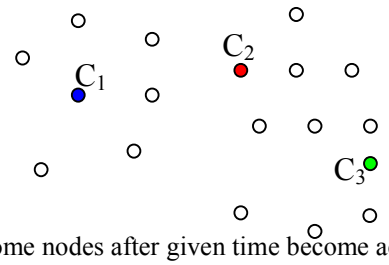
II.1. Example



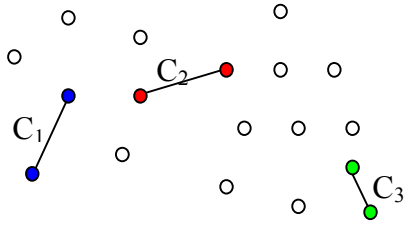
a) Network topology at a given time.



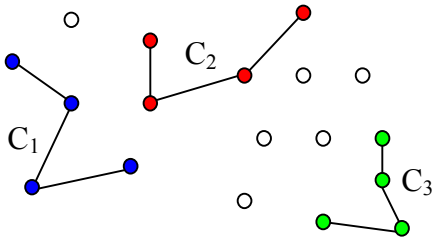
b) Initialisation step.



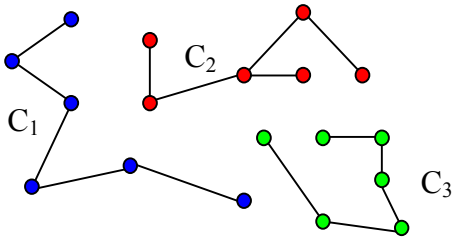
c) Some nodes after given time become active, with colors C_1, C_2 et C_3



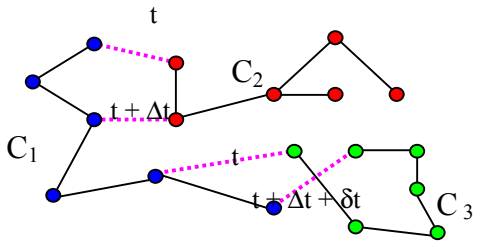
d) Invitation of passive neighbors to become active.



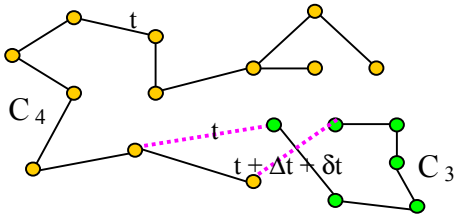
e) Invitation of passive neighbors to become active.



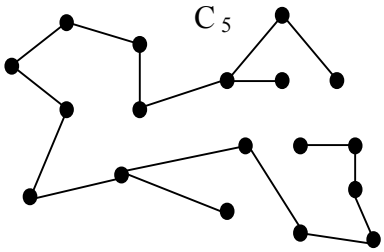
f) Invitation of passive neighbors to become active.



g) Subtrees fusion.



h) Subtrees fusion.



i) Spanning tree with only one color C_5.

Fig.1: Distributed algorithm for constructing a spanning tree in ad hoc network

II.2. Convergence and correctness

Lemma 1. Our algorithm ends and gives a spanning tree in ad hoc network.

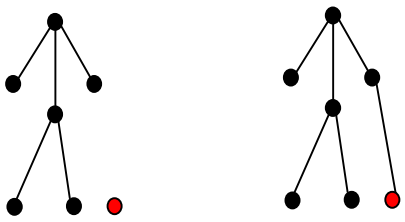
Proof. In our construction the subtrees spread and grow by adding passive nodes or by fusion operation. So the number of subtrees decreases by construction. So the processing stops when all nodes are active and have the same color. Observe that active nodes belonging to the same subtree don't active the link between them. Moreover active nodes belonging to distinct subtrees active only one link each time. So the construction gives a connected and acyclic subgraph in polynomial time.

III. Distributed algorithm for maintaining a spanning tree

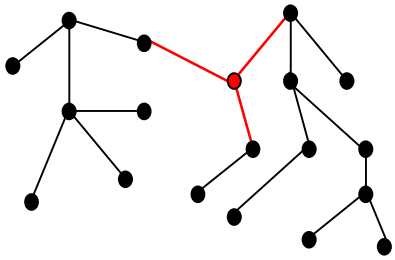
As ad hoc network has dynamic topology, it is necessary that our spanning tree adjusts to these changes. In other words, we must maintain a spanning tree in ad hoc network in spite of the movement, the arrival and the departure of the mobiles. We propose a distributed algorithm for maintaining a spanning tree when some mobiles move, arrive or leave the network.
Our algorithm takes in account adding/deleting node/link:
- add a node: when a new mobile enters in the network, it is considered as a passive node. So the first active node detects this passive node will integrate it in the spanning tree by activating it and the link between them Fig.2.
- delete a node: when a mobile leaves a network (and a spanning tree), it will create k subtrees, with k is a degree of the removed node in a spanning tree. Each neighbor of removed node informs all nodes of its subtree by the disconnection, to become active and to take a new proper color. So the active nodes execute the subtree fusion step of our first algorithm Fig.3.
- delete link: when a mobile moves far from another then the link between them will be broken. So we obtain two subtrees. As is the previous case, each neighbor of removed node informs all nodes of its subtree by the disconnection, to become active and to take a new proper color. So the active nodes execute the subtree fusion step of our first algorithm Fig.3.

Observe that when two mobiles come closer and are together active in a spanning tree then there link is not active since they belong to the same spanning tree (as is described in spanning tree construction algorithm).

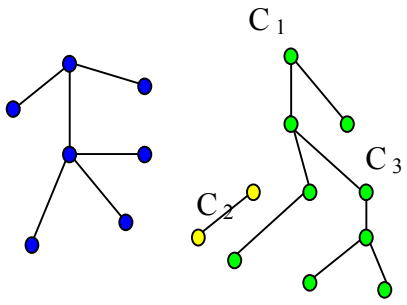
III.1. Example



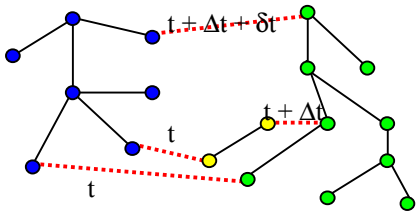
a) New node to add in a spanning tree b) New spanning tree
Fig.2: Added node in a spanning tree.



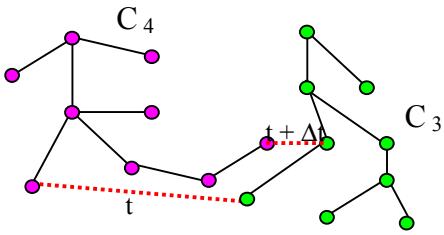
a) (red) node is removed.



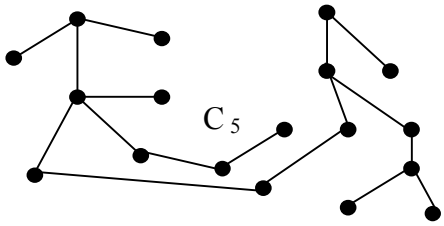
b) created subtrees with different colors and active nodes.



c) Subtrees fusion.



d) Subtrees fusion.



e) New spanning tree with color C_5.

Fig.3: Removed node.

IV. Conclusion

We have presented in this paper two distributed algorithms to construct and maintain in the time a spanning tree in mobile ad hoc network. This spanning tree maybe considered as infrastructure for the communication in ad hoc network. The communication will be optimise since we avoid flooding method often used in such network. Actually we are developing routing and service discovery protocols on a spanning tree in mobile ad hoc network.

References

[1] E. Chang and R. Roberts, An improved algorithm for decentralized extrema-finding in circular configurations of processes, Comm. Assoc. Comput. Mach. 22 (5) pp. 281–283, May 1979.

[2] L. Higham and T. Przytycka, A simple, efficient algorithm for maximum finding on rings, in: A. Schiper, (Ed.) Distributed Algorithms (Seventh International Workshop, WDAG '93) Lausanne, Switzerland, September 1993, Springer, Berlin, pp. 249-263; L. Higham, T. Przytycka, Lecture Notes in Computer Science, vol. 725, Springer, Berlin.

[3] D.S. Hirschberg and J.B. Sinclair, Decentralized extrema-finding in circular configurations of processes, Comm. Assoc. Comput. Mach. 23 (11) pp. 627–628, November 1980.

[4] L. Hu, Topology control for multihop packet radio networks, IEEE Trans. On Communication, vol. 41, no. 10, October 1993.

[5] W. Jia, J. Cao, T.Y. Cheung and X. Jia, A multicast protocol based on a single logical ring using a virtual token and logical clocks, Comput. J. 42 (3) pp. 203–220, 1999.

[6] I. Nikolaidis, J.J. Harms, A logical ring multicast protocol for mobile nodes, in: Proceedings of Seventh International Conference Network Protocols, Toronto, Canada, 1999.

[7] G.L. Perterson, An O(n log n) unidirectional distributed algorithm for the circular extrema problem, ACM Trans. Programming Languages Systems 4 (4), pp. 758–762, October 1982.

[8] V. Rodolplu and T. H. Meng, Minimum energy mobile wireless networks, IEEE J. Selected Areas in Communications, vol. 34, no 1, pp. 38-44, January 1986.

[9] A. Silberschatz, P. Galvin and G. Gagne, Applied Operating System Concepts, first ed. 2000, Wiley, New york, pp. 526–527.

[10] R. Sivakumar, P. Sinha, V. Bharghavan, CEDAR: a core-extraction distributed ad hoc routing algorithm, in: Proc. INFOCOM 1999, pp.202–209

[11] S.Srivastava and R.K.Ghosh, Distributed algorithms for finding and maintaining a k-tree core in a dynamic network ELSEVER, Information Processing Letters 88 pp. 187–194, 2003.

[12] L. Yena and K. Chib. Maintaining a ring structure for mobile ad hoc computing, J. Parallel Distrib. Comput. 64 pp. 1371–1379, 2004.