Self-Stabilizing Tree and Cluster Management for Dynamic Networks

Olivier FLAUZAC, Bachar Salim HAGGAR and Florent NOLOT

CReSTIC/SYSCOM
University of Reims Champagne-Ardenne
UFR Sciences Exactes et Naturelles
Department of Mathematics, Mecanics and Computer Sciences
E-mail: {olivier.flauzac, bachar-salim.haggar, florent.nolot}@univ-reims.fr

Abstract: The lack of infrastructure and dynamic nature of mobile ad hoc networks demand new networking strategies to be implemented in order to provide efficient end-to-end communication. Some researches proposed to organize the network into groups called clusters and use different routing protocols for inter and intra cluster to propagate an information. But with these solutions, the network needs first to be organized into clusters and next, we need to construct each routing table. Other researchers proposed to build a spanning tree on the network to forward informations on a tree but many solutions need to know the global network topology. In this paper, we propose a self-stabilizing algorithm both to construct cluster and simultaneously build a spanning tree on the network. Without any global knowledge, we use only one type of periodically exchanged messages of size $Log(5n + 3)$ bits, and we construct clusters and the spanning tree on the network with a convergence time of at most $D+6$ rounds.

1 Introduction

Today, wireless networks are increasingly popular because of ease of deployment. These networks provide information access to users regardless of their location. However, mobile networks are divided into two main categories: cellular networks and ad hoc networks [BKP02]. While cellular networks are characterized by centralized devices, ad hoc networks are characterized by the absence of infrastructure. Thus, an ad hoc network is a collection of mobile entities inter-connected by a technology without wire, forming a temporary network without the assistance of any management and any fixed architecture. The concept of ad hoc mobile networks tries to extend the notions of mobility to all the components of the environment, contrary to the networks based on the cellular communication. Due to mobility of nodes, the network topology may change quickly and unpredictably over time. The network is decentralized, meaning network organization and message delivery must be executed by the nodes themselves, i.e., routing functionality will be incorporated into mobile nodes.
Mobile ad hoc network can be widely and quickly deployed, without any support from an existing infrastructure or any other kind of fixed stations. The main characteristics of

ad hoc systems: they are self-organizing, fully decentralized and highly dynamic. These characteristics prohibit usage of many applications of algorithms which work in a wired network. On the other hand they provide opportunities for a range of new and interesting applications: conferences, meetings, wireless communication between vehicles in road traffic, disaster relief, rescue missions, military applications, etc. Such scenarios typically lack a central administration or wired infrastructure and, hence, ad hoc systems are very useful for them. Under the limited resources such as network bandwidth, memory capacity, and battery power, the efficiency of routing schemes in ad hoc wireless networks becomes more important and challenging.
In this paper, we proposed a new self-stabilizing algorithm to create clusters on ad hoc network which simultaneously constructs a spanning tree of the network. In each cluster, a node can be clusterhead, gateway or ordinary node. A clusterhead manages data forwarding in its cluster. A gateway is charged to relay messages between clusters. An ordinary node has no particular function, it is neither a clusterhead nor a gateway. With this solution, we have a new solution, with few messages, to forward information over the network.

2 Related Works

We present in this part some existing works on clustering and spanning tree problem. Many solutions for clustering ad hoc networks are intended to identify a subset of nodes geographically closed in a network.
In the Lowest-ID Cluster Algorithm [EWB88], each node in the network must hold an unique identity. The node with the lowest identity over all its neighbors is elected cluster head and the cluster is formed by the cluster head and all its neighbors. In High-Connectivity Clustering [GPL99] and [YC03], cluster head election is based on degree of each node instead of node identity. A node is elected as a cluster head if it has the highest connected node.
The three previous cited algorithms are not self-stabilizing solutions. So they need another algorithm to maintain clusters. Least Clusterhead Change Algorithm (LCC) [Chi97] is designed to minimize cluster head changing. Cluster heads only change when they come neighbors, or when a node becomes disconnected from all cluster heads. This is an improvement (in stability) over existing algorithms which select the clusterhead every time the cluster membership changes.
A different approach of clustering is taken by Basagni in [Bas99]. He presents two clustering algorithms, Distributed Clustering Algorithm (DCA), for "quasi-static" network and Distributed and Mobility-Adaptive Clustering algorithm (DMAC) for mobile network. Each node reacts locally to any topological change in its neighborhood. Both DCA and DMAC assign to nodes different weights and assume that each node is aware of its respective weight. A node is chosen to be a clusterhead if its node-weight is higher than any of its neighbors node-weight. In the DMAC protocol, if two clusters leaders become neighbors, the one with the smaller weight must revoke its leader $Status$. In [JN06a], [JN06b] and [JN09] the authors propose a self-stabilizing version of DCA and DMAC. Moreover,

their solution is robust.
In [CR09] all the previous cited algorithm, Mobility Metric Based Algorithm (MOBIC) [BKL01], Weighted Clustering Algorithm (WCA) [CDT02], [CDT00a], [CDT00b], and Weight Based Clustering Algorithm (WBCA) [YZ07] are studied and compared.
The cluster construction algorithms are not a solution to propagate any information over the network. We need either routing algorithm or spanning tree. We concentrate now our study on existing spanning solution on cluster network. Some authors propose LMST algorithm (Local Minimum Spanning Tree) as in [LHS03]. Each node builds a graph of its neighborhood and broadcasts periodically a $hello$ message which contains its identity and its position. Each node needs to use a system to gather its position, applying Prim's algorithm [Pri57] independently to obtain its local minimum spanning tree. In [CSS04] the authors propose Directed LMST Broadcast Oriented Protocol, an algorithm based on LMST and using directional antennas. The nodes require the knowledge of neighbors position. In [MJ06], the author presents a self-stabilizing distributed algorithms to build a spanning tree. Although this algorithm is self-stabilizing, the number of exchanged messages during operations is important. In [EOD08], distributed algorithms to construct a spanning tree over a network with cluster. In first time, the authors use HEED (Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach) to build the clusters of the networks. In HEED the cluster formation is based on the residual energy of a node and its degree. After clustering, the authors modify the distributed spanning tree formation algorithm for general networks. After formation of spanning trees, each node will have a unique subroot cluster head node. This algorithm uses different kind of messages and are not self-stabilizing.
From this study of existing algorithms, to the best of our knowledge, we can notice there exists no self-stabilizing solution which organize a network in clusters and simultaneously, without a full knowledge of the topology and without a positioning system, builds a spanning tree on the network and on the clusters, with only one type of message.

3 Contributions

From all existing algorithms which build clusters of diameter two, the built clusters can be overlapping, i.e., a node can be in two clusters simultaneously. The deterministic algorithm $MaxCwST$ proposed in this paper builds both clusters of diameter at most equal to two and simultaneously, a spanning tree on the network and on the built clusters. Moreover, it does need neither initialization phase, nor network discovery nor cluster maintain phase. To obtain this result, each node periodically exchanges only one type of message of size $Log(5n + 3)$ bits, when n denotes the number of nodes in the network. The convergence time of our algorithm is at most equal to $D + 6$ rounds, with D the diameter of the graph.

4 Preliminaries

We consider the network as an undirected graph $G = (V, E)$ in which V is the set of nodes and E the set of edges. The size of the network is denoted by $|V| = n$ and we say there exists a link between two nodes u and v if there is an edge $\{u, v\} \in E$. In this case we say that u and v are neighbors and the set of neighbors of a node $u \in V$ will be denoted in this paper by $Neigh_u$. The link to Node v is denoted by $link_v$. We also assume that every node u in the network has an unique identifier which will be u. We define $d(u, v)$ the distance between two nodes u and v in G as the number of edges along a minimal path between the two nodes in G and D is the diameter of the graph.
Clustering means partitioning network nodes into groups called clusters. A *cluster* (illustrated in Figure 1) is a subgraph of G and we assume that the diameter of a cluster must be lower or equal to two and each node belongs to only one cluster and the intersection between any cluster is empty. A node uses Variable $Cl\text{-}id$ to store the identity of its cluster and we denote a cluster by Variable Cl.
Each node exchanges only one type of messages : $hello$ message. This message contains some variables and we use m to denote a message. $m.x$ denotes the variable x contained in Message m. For other variable x, used by Node u, to avoid conflict reading we use notation x_u.
The algorithms presented in this paper are self-stabilizing. The *self-stabilizing* concept was introduced for the first time by E. Dijkstra in [Dij74] as a system, regardless its initial state, which is guaranteed to converge to a legitimate state in a finite number of steps. For the clustering problem, to define the legitimate state, we use the following definition.

Definition 4.1 (Cluster well formed) *A cluster is said well formed when it verifies the four properties :*

1. *it contains only one cluster head*
2. *the cluster head is the node with the largest identity among all nodes in the cluster*
3. *the diameter of the cluster is at most equal to two*
4. *for every pair x, y of clusterhead, x is not a neighbor of y*

From this definition, we can define the legal state as a network in which all clusters are well formed and all nodes are in one cluster.

5 $MaxC$ Self-Stabilizing Clustering Algorithm

The choice of the cluster heads is based on the identity of each node. The cluster head is the node which has the highest identity among all its neighbors, in its cluster. But without lost of generality, we could also choose the node which has the lowest identity. Moreover, from our algorithm, each node eventually satisfies the three following properties : (i) every node

in the network must belong to only one cluster,(ii) all nodes which are not cluster head, are at a distance at most one of a cluster head, and (iii) each cluster has only one cluster head.

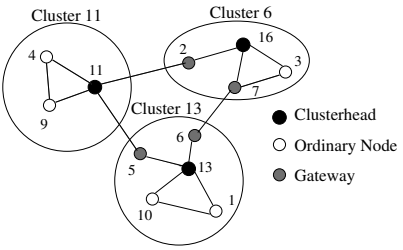


Figure 1: Cluster example

Each node executes the first enabled rule of $MaxC$ algorithm (Algorithm 1). It uses only two variables: $cl\text{-}id$ and $Status$. $cl\text{-}id$ stores the identify of the cluster in which it belongs. $Status$ stores the type of the node. Each node can be in one of the following types : cluster head (CH), gateway (GN) and ordinary node (ON). These three types can be described like this: a cluster head is a node which has the highest identity in its cluster like Nodes 11, 13, and 16 in Figure 1. Gateway node is a node which is adjacent to at least one node belonging to another cluster than him like Node 5 and 6 in Cluster 13 in Figure 1. Node 11 is in cluster 11 and it has two neighbors nodes, Nodes 2 and 5 which are, respectively, in Cluster 6 and 13. Finally, a node which has only neighbor in the same cluster is an ordinary node, like Nodes 4 and 3 for instance, in Figure 1.
We now present the modifications made on $MaxC$ algorithm in order to obtain $MaxCwST$ algorithm, the first algorithm which builds both clusters and simultaneously the spanning tree.

6 Spanning Tree Construction

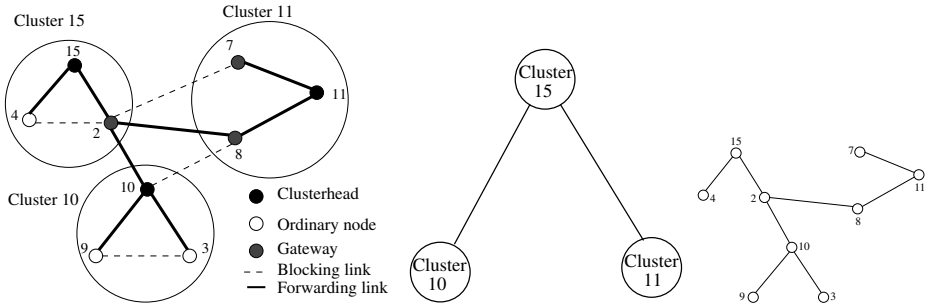
At the same time to clusters creation, we build a spanning tree of the graph and on the cluster, with very simple modification of $MaxC$ algorithm. We have just add two new informations in the $hello$ message: the identity of neighbor cluster and the identity of the gateway node which must be used to join the neighbor cluster.

6.1 MaxCwST algorithm principle

Our spanning tree algorithm is called $MaxCwST$. It is a modification of $MaxC$ algorithm in order to both construct clusters and simultaneously a spanning tree. But to avoid to write all $MaxC$ algorithm rules, we have just, in this paper, write the rules which constructs the spanning tree. It works according to following principle: each node of the graph

Algorithm 1 $MaxC$ Clustering Algorithm on a Node u

$cl\text{-}id$: Identity of the cluster of Node u.
$m.X$: The variable X in the message m
$Status \in \{CH, ON, GN\}$
On receiving $Hello(j, Status, cl\text{-}id)$
R1.a)
if $(Status = CH) \wedge (cl\text{-}id \neq id)$ **then**
  $cl\text{-}id \leftarrow id$;
  $Send\ Hello(id, Status, cl\text{-}id)$;
end if
R1.b)
if $(Status \neq CH) \wedge (\ cl\text{-}id = id \vee (\forall\ m \in Hello, cl\text{-}id \neq m.j) \vee (\exists\ m \in Hello, cl\text{-}id = m.j \wedge m.status \neq CH))$ **then**
  $Status \leftarrow CH$;
  $cl\text{-}id \leftarrow id$;
  $Send\ Hello(id, Status, cl\text{-}id)$;
end if
R2:
R2.a)
if $(Status \neq CH) \wedge (\forall\ m \in Hello, m.j < id)$ **then**
  $Status \leftarrow CH$;
  $cl\text{-}id \leftarrow id$;
  $Send\ Hello(id, Status, cl\text{-}id)$;
end if
R2.b)
if $(Status \neq CH) \wedge (\exists\ m \in Hello, m.Status = CH \wedge m.cl\text{-}id > cl\text{-}id)$ **then**
  $Status \leftarrow GN$;
  $cl\text{-}id \leftarrow m.cl\text{-}id$;
  $Send\ Hello(id, Status, cl\text{-}id)$;
end if
R2.c)
if $(Status \neq CH) \wedge (\exists\ m \in Hello, m.Status = CH \wedge m.cl\text{-}id < cl\text{-}id)$ **then**
  $Status \leftarrow GN$;
  $Send\ Hello(id, Status, cl\text{-}id)$;
end if
R2.d)
if $(Status \neq CH) \wedge (\exists\ m \in Hello, (m.Status = GN \vee m.Status = ON) \wedge m.cl\text{-}id \neq cl\text{-}id)$ **then**
  $Status \leftarrow GN$;
  $Send\ Hello(id, Status, cl\text{-}id)$;
end if
R2.e)
if $(Status \neq CH) \wedge (\exists\ m \in Hello, (m.Status = CH \wedge m.cl\text{-}id = cl\text{-}id)$ **then**
  $Status \leftarrow ON$;
  $Send\ Hello(id, Status, cl\text{-}id)$;
end if
R3)
if $(Status = CH) \wedge (\exists\ m \in Hello, m.Status = CH \wedge m.j > id)$ **then**
  $Status \leftarrow ON$;
  $cl\text{-}id \leftarrow m.cl\text{-}id$;
  $Send\ Hello(id, Status, cl\text{-}id)$;
end if
R4)
$Send\ Hello(id, Status, cl\text{-}id)$;



(a) Cluster and Spanning tree obtained from $MaxCwST$ (b) Spanning tree view on the clusters (c) Spanning tree view on the network

Figure 2: Example of constructing spanning tree of network by $MaxCwST$

chooses only one neighbor node as father node in the spanning tree. Within each cluster, the cluster head will be the father of each nodes of its cluster. for instance, in Figure 2(a), Node 4 and 2 of Cluster 15 have chosen Node 15, their cluster head, as their father in this cluster. Nodes 7 and 8 have chosen Node 11 and Nodes 9 and 3 have chosen Node 10. All links between gateway or ordinary nodes in the same cluster will be in a blocking state. In this state, link will forward only $hello$ messages. Now, between each cluster, we need to make the spanning tree. So, only one gateway between two clusters need to "activate" its link. So, the cluster head will choose which gateway can be use to join a neighbor cluster. For instance, in Figure 2(a), Node 7 and 8 can be used to join Cluster 15. So, to avoid to make a loop, and to construct a spanning tree on the network and on the cluster, we have to activate only one link between Cluster 11 and 15. In the $hello$ message, we add the identity of the neighbor cluster. So, Node 8 will receive an $hello$ message from Node 2 which contains the cluster identity 15. This information will be forward to the cluster head, in the $hello$ message send by Node 8. The same exchange is made between Node 2 and 7 and next, between Node 7 and 11. When the cluster head 11 has received the two $hello$ messages from Node 7 and 8. It can choose only one gateway. This choice is based on the identity of this gateway. The gateway node with the highest identity will be chosen. In our example, cluster head of cluster 11 will choose gateway 8. We need also avoid another case. When a gateway has a link to more than one cluster, like Node 8 in Figure 2(a). In this case, the gateway always chooses the hello message from the highest identity cluster. So, Node 8 chooses Cluster 15 as father. From our algorithm, we obtain a spanning tree on all clusters (Figure 2(b)) and on the network (Figure 2(c)).

6.2 The proof

To prove the $MaxCwST$ algorithm, we use the following property:

Property 6.1 (Characteristic of a tree) *Let S be a subgraph of a graph G and n the size of G. S has exactly n-1 edges iff S is a tree of n nodes.*

Property 6.2 *Let i be a cluster. Each node which is not a cluster head, chooses only one link to another node in Cluster i. This link will be the link to the cluster head of cluster i.*

Proof. Only Rules $R3.a$ and $R3.b$ can be executed in this case. From Rule $R3.a$, the link to the cluster head will be in forwarding mode and from Rule $R3.b$, all other link to nodes in the same cluster ($m.cl\text{-}id$= $cl\text{-}id_i$) will be in blocking state. □

Algorithm 2 $MaxCwST$ on Node i

$MaxCwST$: Clustering and Spanning Tree Algorithm
id_i: Identity of node i
$m.cl\text{-}id$: Identity of the cluster in message m
$cl\text{-}id_i$: Identity of the cluster of Node i
$cl\text{-}id_{adj}$: An array that contains identity of gateway $cl\text{-}id_{adj}.id$ which can be used to access to neighbor cluster $cl\text{-}id_{adj}.cl\text{-}id$
$Forwarding$: In this state, the link will transmit data packets
$Blocking$: In this state, the link will transmit only $hello$ message
$Port \in \{Forwarding, Blocking\}$
$Port(cl\text{-}id_{adj})$: Identity of the link which can be used to join neighbor cluster $cl\text{-}id_{adj}$
$Port(Max(cl\text{-}id_{adj}))$: Identity of the link which can be used to join a neighbor cluster which have the highest identity
$Port(j)$: The link which connects a node i to a node j
NPC: contains $cl\text{-}id$ and identity of gateway chosen
$NPC.id$: identity of gateway contained in the variable NPC
R1)
if $(Status_i = CH) \wedge (cl\text{-}id_{adj}.id > 1)$ **then**
  $NPC \leftarrow Max(cl\text{-}id_{adj}.id)$
end if
On receiving $Hello(j, Status, cl\text{-}id, cl\text{-}id_{adj}, NPC)$
R2.a)
if $(Status_i = CH) \wedge (\forall\ m \in Hello, cl\text{-}id_i$ ¿ $cl\text{-}id_{adj}.cl\text{-}id)$ **then**
  $Port(cl_{adj}) \leftarrow Forwarding$;
end if
R2.b)
if $(Status_i = CH) \wedge (\exists\ m \in Hello, cl\text{-}id_{adj}.cl\text{-}id$ ¿ $cl\text{-}id_i)$ **then**
  $Port(Max(cl\text{-}id_{adj})) \leftarrow Forwarding$;
  $Port(\neg Max(cl\text{-}id_{adj})) \leftarrow Blocking$;
end if
R3.a)
if $(Status_i = ON \vee Status_i = NP) \wedge ((\exists\ m \in Hello, m.Status = CH) \wedge (m.cl\text{-}id$= $cl\text{-}id_i))$ **then**
  $Port_j \leftarrow Forwarding$;
end if
R3.b)
if $(Status_i = NP) \wedge ((\exists\ m \in Hello, m.Status \neq CH) \wedge (m.cl\text{-}id$= $cl\text{-}id_i)$ **then**
  $Port_j \leftarrow Blocking$;
end if
R4.a)
if $(Status_i = NP) \wedge ((\exists\ m \in Hello, m.Status = CH) \wedge (m.cl\text{-}id$= $cl\text{-}id_i) \wedge (id_i \neq NPC.id))$ **then**
  $Port(cl\text{-}id_{adj}) \leftarrow Blocking$;
end if
R4.b)
if $(Status_i = NP) \wedge ((\exists\ m \in Hello, m.Status = CH) \wedge (m.cl\text{-}id$= $cl\text{-}id_i) \wedge (id_i = NPC.id))$ **then**
  $Port(cl\text{-}id_{adj}) \leftarrow Forwarding$;
end if

From this property, for each node in a cluster, only one link to another node in the same cluster will be in forwarding state. So, in each cluster, we have a spanning tree.

Property 6.3 *Each cluster chooses only on father cluster*

Proof. We need to examine two cases. Either the node which can communicate with another cluster is a cluster head, or it is a gateway. For the first case, like a cluster head

is also a gateway, from Rule $R2.b$, only one link to another cluster will be chosen. In a second case, from Rule $R1$, the cluster head of each cluster chooses the highest identity of neighbor cluster and from $R4.a$ and $R4.b$ only one link will be chosen. So a gateway node chooses also only one father. □

From the previous properties, each cluster head chooses only one gateway node which has the permission to activate its link to a father cluster and in each cluster, each node chooses only one link towards a father node. Only the node with the highest identity activates all its links. So, on a graph of n nodes, only $n - 1$ nodes have only one link in forwarding state. From Property 6.1, we have a tree.

7 Conclusion

In this paper, we have proposed the first deterministic and self-stabilizing algorithm for partitioning a network into multiple clusters which simultaneously constructs a spanning tree on the network and on the cluster. After at most $D + 6$ rounds, the spanning tree is created and the clusters are formed. Each node just needs to discover its neighborhood and their identity. No global knowledge is required to make the spanning tree. Moreover, we do not need maintain phase to maintain the cluster. Our solution is self-stabilizing and self-organized on an ad hoc network. The presented algorithm may be easily and efficiently applied in a broadcasting protocol for a distributed network. Unfortunately, our solution overload the cluster heads because they are responsible of the choice of right gateway to propagate informations. We need to find solution to this problem to achieve an improved version of this algorithm.

References

[Bas99] Stefano Basagni. Distributed Clustering for Ad Hoc Networks. In *ISPAN*, pages 310–315, 1999.

[BKL01] Prithwish Basu, Naved Khan, and Thomas D.C. Little. A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks. In *In International Workshop on Wireless Networks and Mobile Computing (WNMC2001*, pages 413–418, 2001.

[BKP02] Claudio Basile, Marc-Oliver Killijian, and David Powell. A survey of dependability issues in mobile wireless networks. Technical Report 02637, LAAS, Toulouse, 2002.

[CDT00a] Mainak Chatterjee, Sajal K. Das, and Damla Turgut. An On-Demand Weighted Clustering Algorithm (WCA) for Ad hoc Networks. In *In Proceedings of IEEE GLOBECOM 2000*, pages 1697–1701. ACM Press, 2000.

[CDT00b] Mainak Chatterjee, Sajal K. Das, and Damla Turgut. A Weight Based Distributed Clustering Algorithm for Mobile ad hoc Networks. In *HiPC '00: Proceedings of the 7th International Conference on High Performance Computing*, pages 511–521, London, UK, 2000. Springer-Verlag.

[CDT02] Mainak Chatterjee, Sajal K. Das, and Damla Turgut. WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks. *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks*, 5:193–204, 2002.

[Chi97] Ching-Chuan Chiang. Routing In Clustered Multihop, Mobile Wireless Networks With Fading Channel, 1997.

[CR09] Suchismita Chinara and Santanu Kumar Rath. A Survey on One-Hop Clustering Algorithms in Mobile Ad Hoc Networks. *J. Netw. Syst. Manage.*, 17(1-2):183–207, 2009.

[CSS04] J. Cartigny, D. Simplot, and I. Stojmenovic. An Adaptive Localized Scheme for Energy-efficient Broadcasting in Ad hoc Networks with Directional Antennas. In I. Niemegeers and S. Heemstra de Groot, editors, *Proc. 9th IFIP Int. Conf. on Personal Wireless Communications (PWC 2004)*, volume 3260 of *Lecture Notes in Computer Science*, pages 399–413, Delft, The Netherlands, 2004. Springer-Verlag, Berlin. Best paper award.

[Dij74] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, 1974.

[EOD08] Kayhan Erciyes, Deniz Ozsoyeller, and Orhan Dagdeviren. Distributed Algorithms to Form Cluster Based Spanning Trees in Wireless Sensor Networks. In *ICCS '08: Proceedings of the 8th international conference on Computational Science, Part I*, pages 519–528, Berlin, Heidelberg, 2008. Springer-Verlag.

[EWB88] A. Ephremides, J-E. Wieselthier, and D-J. Baker. A design concept for reliable mobile radio networks with frequency-hopping signaling. *NASA STI/Recon Technical Report N*, 89:17772–+, September 1988.

[GPL99] M. Gerla, G. P, and S-J. Lee. Wireless, mobile ad-hoc network routing. In *ACM FOCUS*, 1999.

[JN06a] Colete Johnen and Le Huy Nguyen. Self-stabilizing Weight-Based Clustering Algorithm for Ad Hoc Sensor Networks. In Sotiris E. Nikoletseas and José D. P. Rolim, editors, *Algorithmic Aspects of Wireless Sensor Networks, Second International Workshop, ALGOSENSORS 2006, Venice, Italy, July 15, 2006, Revised Selected Papers*, volume 4240 of *Lecture Notes in Computer Science*, pages 83–94, 2006.

[JN06b] Colette Johnen and Le Huy Nguyen. Robust Self-stabilizing Clustering Algorithm. In *OPODIS*, pages 410–424, 2006.

[JN09] Colette Johnen and Le Huy Nguyen. Robust self-stabilizing weight-based clustering algorithm. *Theor. Comput. Sci.*, 410(6-7):581–594, 2009.

[LHS03] Ning Li, Jennifer C. Hou, and Lui Sha. Design and Analysis of an MST-Based Topology Control Algorithm. In *INFOCOM*, 2003.

[MJ06] Ricardo Marcelin-Jimnez. Locally-Constructed Trees for Adhoc Routing. In *PWC*, pages 194–204, 2006.

[Pri57] R. C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*, pages 1389–1401, nov 1957.

[YC03] J.Y Yu and P.H.J Chong. 3hBAC (3-hop between adjacent clusterheads): a novel non-overlapping clustering algorithm for mobile ad hoc networks. 1:318–321, 2003.

[YZ07] Wei-Dong Yang and Guang-Zhao Zhang. A Weight-Based Clustering Algorithm for Mobile Ad Hoc Network. *Wireless and Mobile Communications, International Conference on*, 0:3, 2007.