

Fair and Flexible Budget-Based Clustering

Fehmi Ben Abdesslem*, Artur Ziviani†, Marcelo Dias de Amorim*, and Petia Todorova‡

*UPMC Univ Paris 06, Paris, France. Emails: {fehmi,amorim}@rp.lip6.fr

†National Laboratory for Scientific Computing (LNCC), Petrópolis, Brazil. Email: ziviani@lncc.br

‡Fraunhofer FOKUS, Berlin, Germany. Email: Petia.Todorova@fokus.fraunhofer.de

Abstract—An efficient way to bound the size of clusters in large-scale self-organizing wireless networks is to rely on a budget-based strategy. The side effect of conventional budget-based clustering approaches is that they generate a potentially large number of small, even single-node, clusters. The consequence is that while clusters are bounded, their average size may be far from the expected value (the budget), which negatively impacts the performance of the communication systems running on top of it. In contrast, we propose Fair and Flexible Budget-Based Clustering (FFBC) to form size-controlled clusters in large-scale self-organizing networks. For a given target cluster size, our approach outperforms previous budget-based algorithms by creating clusters of average size closer to the requested value and avoiding isolated nodes.

I. INTRODUCTION

Deploying large-scale wireless networks without any static infrastructure provides multiple benefits, such as flexibility and robustness to failures [1]. Typical examples are sensor networks deployed randomly, or spontaneous ad hoc networks deployed by users in a temporary event. With the expansion of the Internet, large-scale communication networks have become a challenging topic for research works, mainly in terms of scalability. Indeed, many communication algorithms do not provide satisfying performance when executed on large networks. Clustering is a convenient solution to this problem, and consists in partitioning the network topology into small groups of nodes (clusters), which then allows local execution of communication algorithms within each cluster. For wired large-scale networks such as the Internet, partitioning relies on the static and hierarchical properties of the topology. Indeed, since the topology is already known, cluster formation can be done off-line by a central entity. Clusters can also be formed according to the hierarchical design of the network (e.g., an Autonomous System) [2], [3], or any other considerations such as the distance between nodes [4].

In self-organizing wireless networks, however, we cannot make the same assumptions, as the global topology is unknown and there is no central authority. Indeed, we consider a wireless network composed of randomly deployed nodes that cannot be aware of the global topology. In such a case, clustering mechanisms must be distributed [5]. Since the cluster formation algorithm is fully distributed, and nodes lack a global view of the topology, controlling the size of the clusters (in terms of number of nodes) is challenging [3], [6]. On the one hand, if an algorithm or protocol needs to partition the network for scalability reasons, the number of nodes in each cluster must be bounded, otherwise the clusters would still be

too large for the clustering algorithm to be executed efficiently. On the other hand, too small clusters also lead to inefficiency. Therefore, clustering algorithms for self-organizing networks must be able to control the size of the formed clusters.

Most clustering algorithms do not control the number of nodes in the clusters. Instead, they consider the convergence time, diameter of clusters, clusters stability, or network lifetime. Preliminary works proposed algorithms to create clusters where all nodes are one-hop away from the cluster-head (CH) [7]–[12]. In such approaches, the average cluster size is indeed limited, and depends on the network density and the transmission range, which cannot be changed by the algorithm. Other clustering algorithms require knowledge of the whole topology (by building a spanning tree for instance) [13]–[16], which is not scalable in large flat networks. In the clustering algorithm by Lin and Chu [17], and in the one by Mitton et al. [18], although nodes are only aware of the local topology and form multi-hop clusters, the cluster size cannot be controlled by the algorithm.

In this paper, we address the issue of splitting a multi-hop topology into smaller and size-controlled clusters. Only a few algorithms attempt to create bounded size multi-hop clusters in a distributed way. Ramamoorthy et al. [19] propose the Expanding Ring algorithm to bound the size of clusters based on an expanding ring search. A more efficient way to control the size of clusters in a distributed way is to rely on a budget-based strategy. The idea is to distribute tokens to neighbors, and each neighbor receiving a token joins the cluster under formation. Three instantiations of such an approach in the literature are the Rapid [6], Persistent [6], and Potential-Based Clustering [20] algorithms, which aim at providing cluster sizes as close as possible to the budget.

The problems with the existing budget-based approaches are twofold. First, some neighbors available to join a cluster in formation might not receive any token, whereas other neighbors receive more than one. Second, the budget value is always considered as a bound. In most contexts, the goal would be rather to create clusters of size close to a target value, may they be slightly smaller or larger than this value.

To address the problems raised above, we propose Fair and Flexible Budget-based Clustering (FFBC), a budget-based algorithm that improves budget distribution by allocating one token to every neighbor, and distributes the remaining budget according to the connectivity degree of neighbors. Our algorithm allows the formation of multi-hop clusters, with nodes more than one-hop away from the CH. FFBC does

not rely on the knowledge of the whole topology, and does not adopt any flooding or spanning tree strategy to build its clusters. All the nodes need only to be aware of their one-hop neighbors. FFBC also avoids single-node clusters by joining isolated nodes to a neighboring cluster. In our algorithm, the targeted cluster size is no more a bound for the size of clusters in formation. The budget is flexible as clusters accept neighboring isolated nodes, even if they did not receive any token. We show through simulation that FFBC outperforms existing budget-based algorithms by creating clusters of size closer to the requested value and avoiding isolated nodes. Our simulation results indicate that the proposed FFBC algorithm provides an improvement in performance up to 30% better than with Persistent and up to 18% better than with PBC.

The remainder of this paper is organized as follows. In Section II, we briefly review budget-based clustering algorithms. Our new FFBC algorithm is presented in Section III and evaluated in Section IV. We finally conclude in Section V.

II. BUDGET-BASED CLUSTERING

Budget-based algorithms rely on the distribution of tokens to form clusters without prior knowledge of the global topology. Existing algorithms usually require each node to set a random timer, and to wait for a token from one of its neighbor in order to join a cluster. When the timer is over, the node stops waiting for a token and becomes a CH that initiates a cluster formation. Tokens are then propagated in a tree rooted at the CH. The CH is assigned a budget of β tokens, out of which it accounts for itself and distributes $\beta - 1$ to its neighbors (except to its parent). These neighbors do the same until the budget is exhausted. If a node receives a budget that it cannot propagate (no neighbors in its subtree), this node either discards the extra tokens or send the extra tokens back to the initiator.

In Rapid and Persistent budget-based algorithms [6], tokens are equally distributed to neighbors. This strategy does not always allow a fair distribution. For instance, let A be a node that handles 5 tokens. It consumes one of them and still has to distribute the remaining 4 tokens. Let B and C form the set of its neighbors; B has 2 other neighbors while C does not have any other neighbor than A . Equally distributing the tokens would allocate 2 tokens to B and 2 tokens to C . In this case, B would consume one token and send the remaining one to one of its neighbors. As for C , it would consume one token and discard the other one (no more neighbors available). In this simple scenario, B should have been allocated more tokens than C , because it has more possibilities for token distribution.

In Potential-Based Clustering (PBC) [20], the potential of a node is defined as the number of neighbors it has and the distribution is based on the connectivity degree (potential) of each neighbor. This leads to a better distribution of clusters when discarding unallocated tokens, and to the formation of more compact clusters when unallocated tokens are sent back to be distributed elsewhere. Distributing tokens according to the connectivity degree of neighbors provides better performance in uniform random topologies than blindly distributing tokens to all neighbors. Although PBC increases the average

cluster size, it is still prone to generating a large number of small, even single-node, clusters.

III. FAIR AND FLEXIBLE BUDGET-BASED CLUSTERING

In FFBC, CHs are also randomly chosen with a timer as in previous budget-based algorithms. Then every node maintains a database containing the potentials of its neighbors and uses this database to determine the amount of budget assigned to each one of them. For any node X in the network, let $\mathbf{N}(X) = \{x_1, x_2, \dots\}$ denote the set of X 's neighbors and $|\mathbf{N}(X)|$ be the size of this set. The potential of node X , denoted $\pi(X)$, is then given by $\pi(X) = |\mathbf{N}(X)|$. The efficiency of the algorithm depends on the way the budget is distributed among the neighbors.

FFBC relies on two principles that are described in details below: (i) one token is first distributed to each neighbor. If some tokens remain after a first round, they are distributed according to the connectivity degree and (ii) when a node N does not belong to any cluster yet, and has no more neighbors available because all of them already belong to a cluster, then node N automatically joins the cluster of the last of its neighbors to be clustered, even if N did not receive any token from this neighbor.

Fair round first. Let A be a node handling some budget β_A to be distributed among its neighbors. Node A first accounts for itself and then distributes the remaining tokens to its neighbors. One token is first distributed to each neighbor (for the sake of fairness).

The rationale behind this first distribution is that tokens should be allocated directly to available neighbors, instead of being distributed to neighbors that will recruit nodes elsewhere. When distributing tokens according to the connectivity degree, some low potential neighbors do not receive any token, whereas high potential neighbors receive more than one token. FFBC already attempts to avoid low potential neighbors to stay aside of the cluster formations by distributing remaining tokens to lower potential neighbors. Nevertheless, when there are too many low potential neighbors, these remaining tokens are not enough to guarantee that every neighbor will receive at least one token. If a neighbor does not receive any token, it will not join the cluster and wait for a token from another cluster. However, since it has a low potential, it has only a few neighbors and is likely to create its own small cluster. Hence, before allocating more than one token to some neighbors, a node should first ensure that all of its neighbors belong to its cluster. To form more compact clusters, recruiting one-hop neighbors is always better than recruiting multiple-hop ones. For this reason, FFBC allocates one token per neighbor in a first round, and then distributes the remaining tokens according to the potentials of nearby nodes. This approach leads to more compact clusters, and avoids available neighbors to form their own small cluster because all their neighbors have already joined a previous formed cluster.

Distribution of remaining tokens. If the budget $(\beta_A - 1)$ to be distributed by node A is smaller or equal to its own potential

$\pi(A)$, the budget distribution disregards the potentials of the neighbors: one token is distributed to each neighbor, in a random order, until the budget is exhausted. If the budget $(\beta_A - 1)$ is greater than $\pi(A)$, at least one token will remain after the first distribution of one token per neighbor. Let β_A^R be this remaining budget to distribute by node A after the first round, then $\beta_A^R = (\beta_A - 1) - \pi(A)$. The amount of budget assigned to neighbor $a_i \in \mathbf{N}(A)$ is defined as:
$$\beta_{a_i} = \left\lfloor \frac{\beta_A^R \pi(a_i)}{\sum_{j=1}^{|\mathbf{N}(A)|} \pi(a_j)} \right\rfloor.$$
 From this equation, we see that some tokens may be unassigned if β_A^R is not a multiple of $\sum_{j=1}^{|\mathbf{N}(A)|} \pi(a_j)$. In this case, the remaining tokens can be evenly distributed among the β_r neighbors with the lowest potentials. We define β_r as $\beta_r = \beta_A^R - \sum_{j=1}^{|\mathbf{N}(A)|} \beta_{a_j}$. For the β_r lowest-potential neighbors, node A performs $\beta_{a_i} \leftarrow \beta_{a_i} + 1$ before distributing the budget. This guarantees that no tokens are wasted. We observed with some preliminary analysis that attributing the remaining tokens to lowest-potential neighbors provides better performance. Without this design choice, a low-potential node frequently remains without any tokens until it initiates its own cluster. However, since it only has a few neighbors, they often belong to another cluster and are not available to accept tokens. As a consequence, the tokens would propagate through the topology avoiding low-potential nodes by clustering only high-potential ones, resulting in small clusters in low-density areas and excessive lost tokens in high density areas. Distributing the remaining tokens to the lowest-potential nodes attenuates this counterproductive phenomenon by giving more chances to low-potential nodes to join growing clusters. The same distribution scheme is applied by the nodes in the subtrees until the budget is exhausted or no further growth is possible.

Recovering unused tokens. In FFBC, to avoid discarding unallocated tokens, these tokens are sent back to be distributed elsewhere. When a node B receives tokens, it sends an acknowledgement to the sender A . Nonetheless, this acknowledgement is not sent immediately. The node B keeps on propagating the tokens to its own neighbors first, and waits for an acknowledgment from all of them. When either the budget is met (no more tokens to distribute) or when no more neighbors are available, an acknowledgment is sent back to A . The acknowledgement includes the number of exceeding tokens (which may be zero). When acknowledgements are received from all the neighbors, node B computes the sum of the exceeding tokens and distributes them to the neighbors that have sent an empty acknowledgement (with no exceeding tokens). If no neighbor is available to redistribute the exceeding tokens, then the extra tokens are sent to the parent A through the acknowledgement. In this way, the algorithm attempts to redistribute the tokens by exploring the whole tree formed by the CH. Exceeding tokens are never discarded, except by the CH, which is the tree root, and only when the entire tree has been explored. For each token distribution, the connectivity degree is taken into account. Of course, after receiving back exceeding tokens, the neighbors that sent

back at least one extra token are excluded from the new distributions. FFBC is more efficient than Persistent because it takes into account the connectivity degree of neighbors when distributing tokens, leading to the formation of more compact clusters. When clusters are more compact, there are less sparse nodes between already formed clusters. Avoiding the presence of these isolated nodes is essential since they will initiate their own small clusters. The size of clusters initiated by isolated nodes will be rather limited, lowering the average size of the formed clusters, because all nodes around already belong to other clusters. To avoid such isolated nodes, FFBC implements two measures to further improve the efficiency of budget-based clustering, namely fair round first, already mentioned earlier, and flexible budget. Both mechanisms are further explained in the following.

Relaxing the bound constraint. By distributing first a token to every neighbor, formed clusters are more compact and there are less small clusters. Nevertheless, in random topologies, we observe that there are still very small clusters, and even single-node clusters. We suggest merging those single-node clusters to neighboring clusters. When the last neighbor of a node is being recruited by a growing cluster, the node will have to initiate its own single-node cluster. Instead of initiating a cluster, it joins the cluster of the last neighbor being recruited. This will result in clusters with a size slightly larger than the initial budget, but avoiding small or single-node cluster, thus increasing the average size of clusters. The budget is then no more considered as a strict upper-bound. We show in simulation that this measure increases the average size of clusters, although cluster sizes larger than the initial budget are still close to the specified size determined by the initial budget.

IV. PERFORMANCE EVALUATION

Size-controlled clustering algorithms are mainly evaluated through the average size of formed clusters. The closer the resulting average cluster size is to the target cluster size defined by the initial budget, the better. In our performance analysis, we evaluate the following budget-based clustering algorithms:

- Rapid – A simple algorithm that equally distributes tokens to neighbors and discards unallocated tokens [6];
- PBC-Simple – An algorithm that distributes tokens to neighbors according to their connectivity degree and discards unallocated tokens [20];
- FFBC-Simple – A simplified version of our contribution, implementing fair round first and flexible budget mechanisms, whereas discarding unallocated tokens;
- Persistent – An algorithm that equally distributes tokens to neighbors and sends back unallocated tokens to distribute them elsewhere [6];
- PBC – An algorithm that distributes tokens to neighbors according to their connectivity degree and sends back unallocated tokens to distribute them elsewhere [20];
- FFBC – A complete version of our contribution, implementing fair round first and flexible budget mechanisms,

and sending back unallocated tokens to distribute them elsewhere.

A. Simulation set-up

To evaluate and compare the considered budget-based clustering algorithms, we use the ns-2 network simulator [21]. Our random topologies were generated by Topogen [22], with a uniform distribution over a circular area of 5,000 m of diameter. The radio range of the nodes was fixed to 100 meters. We varied the density from 2 to 8 nodes per coverage range, by varying the number of nodes from 1,250 to 5,000. For each simulation, we used the same topology for all the algorithms, and generated a new topology for each round of simulations.

B. Results

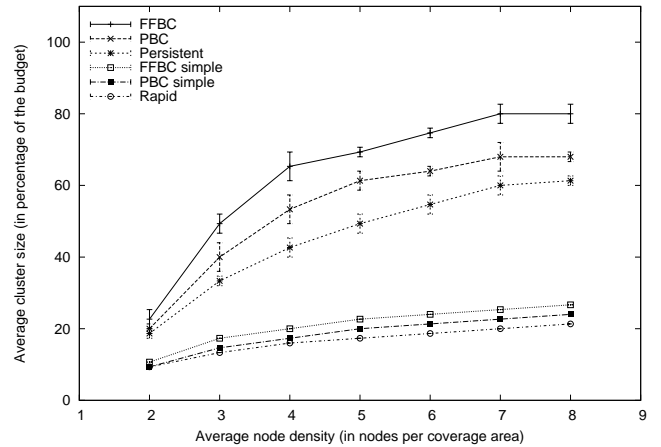
Fig. 1 shows for different budget sizes the resulting average sizes of clusters formed by the different algorithms as a percentage of the target cluster size, i.e. the initial budget. Values plotted in the figures are the average values over 10 different topologies, and error bars indicate the standard deviation. We observe that whatever the budget used, ranging from 25 to 75, our proposed FFBC approach outperforms previous ones.

For instance, in Fig. 1(a), with a budget of 75 tokens and for an average density of 8 nodes per coverage area, the Rapid algorithm creates clusters with an average size of 16.9 nodes (in average over 10 simulations), which is 22.7% of the budget. Using a potential-based distribution (the simple version of PBC), the average size raises to 25.3% of the budget (18.8 nodes). Adding the improvements of FFBC increases the average size to 27.3% of the budget. Here, our approach increases the cluster sizes by more than 20% compared to the Rapid algorithm, and almost 9% compared to PBC-Simple.

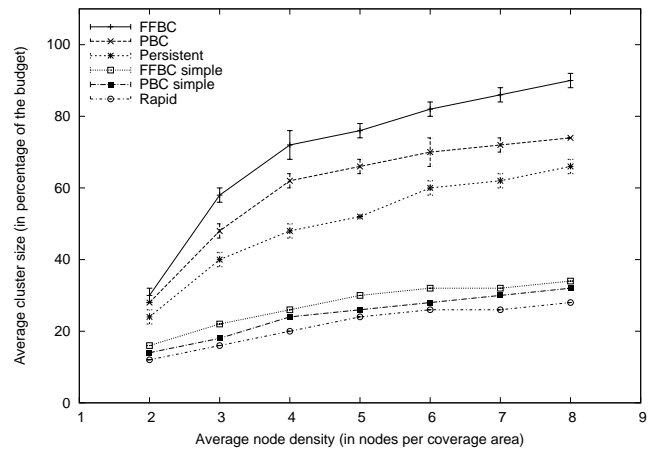
As for algorithms that send back undistributed tokens, we observe in the same Fig. 1(a) and for a density of 8 nodes per coverage area, that the Persistent algorithm creates clusters of an average size of 46.8 nodes (62.4% of the budget). Using a potential-based distribution (PBC), the average cluster size is increased to 51.6 nodes (68.8% of the budget). Adding the improvements of FFBC, the average size of clusters is of 60.9 nodes (81.2% of the budget). In other words, the proposed FFBC algorithm increases the average cluster size by more than 30% compared to the Persistent algorithm and more than 18% compared to PBC, driving the average cluster size closer to the target cluster size, i.e. the initial budget.

For smaller budgets, as shown in Fig. 1(b) and 1(c), the average cluster size of FFBC reaches 90% of the budget (104%, respectively), instead of 75% of the budget (80%) for the PBC algorithm. Note that whereas PBC considers the budget as a bound size for the clusters (like previous algorithms), FFBC allows this budget to be exceeded. Hence, the average cluster size can be greater than the budget as in Fig. 1(c).

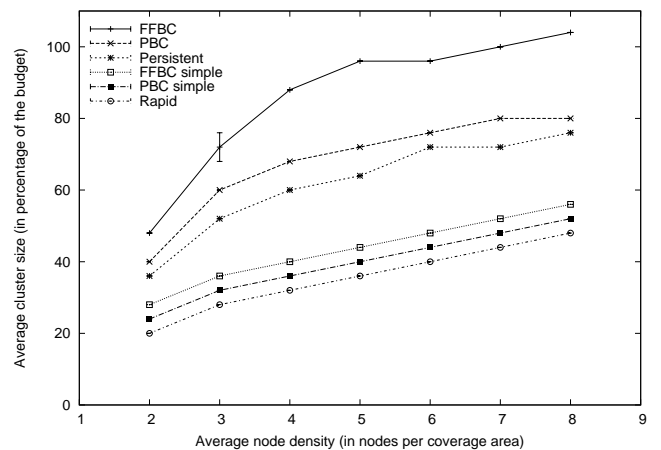
Fig. 2 shows the distribution of the cluster sizes, for a random topology of 5,000 nodes, and with a budget of 75 tokens. Values have been averaged over 10 simulations with



(a) Budget of 75 tokens.



(b) Budget of 50 tokens.



(c) Budget of 25 tokens.

Fig. 1. Average size of formed clusters with different algorithms, and different budgets, varying the node density.

a different topology for every simulation. With the Persistent algorithm, more than 30% of the clusters include less than 5 nodes. Using a potential-based distribution is enough to

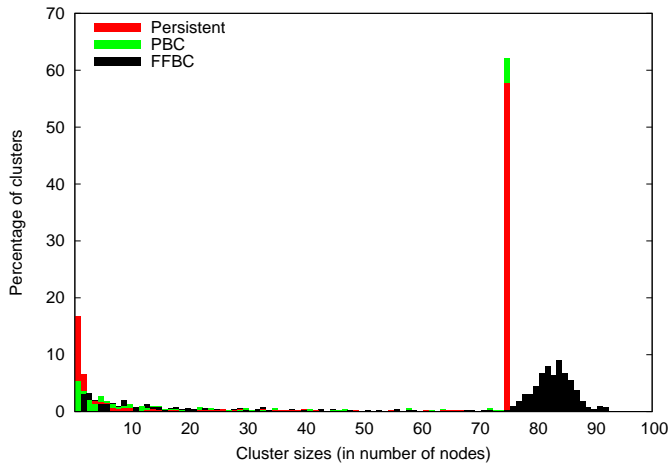


Fig. 2. Distribution of cluster sizes (Budget of 75 tokens, 5,000 nodes).

decrease this value to less than 15%. With the improvement mechanisms introduced by FFBC, less than 10% of the clusters are smaller than 5 nodes. Furthermore, with the Persistent algorithm, clusters that have a size close to the budget ($\pm 20\%$ of the budget) represent around 60 clusters out of 107 (56% of the clusters). For the PBC algorithm, 67% of the nodes have a size close to the budget. As for FFBC, more than 79% of the clusters have a size close to the budget (65 clusters out of 82).

V. CONCLUSION AND ONGOING WORK

In this paper, we proposed FFBC, a distributed cluster formation algorithm to create clusters of a specified size. Contrary to most of previous cluster formation algorithms, we do not rely on a global knowledge of the topology and we assume nodes have only a limited view of the local topology. These assumptions are more realistic for large-scale self-organizing wireless networks, such as sensor networks. FFBC improves the token distribution by distributing first one token to every neighbor before distributing the remaining according to the connectivity degree. Moreover, in contrast to previous work, FFBC does not consider the targeted size of cluster as a bound. The budget is then more flexible and avoids the formation of single-node clusters. Through extensive simulations, we show that our algorithm achieves a significant improvement in the performance as compared to previous budget-based algorithms, thus creating cluster with sizes closer to the target size. In particular, with FFBC, more than 79% of the clusters have a size close to the target value, against only 67% for PBC and 56% for Persistent.

We addressed large-scale wireless networks where nodes are randomly deployed according to a uniform distribution. We are now working on considering other scenarios, with density variations in the topology. To improve even more our algorithm, we are also testing heuristics to choose the set of initiators (CHs). Moreover choosing CHs with timers can be a

practical limitation of budget-based algorithms, since neighbor nodes with close timer values initiate cluster formations at the same time and hence compete in recruiting neighboring nodes, leading to limited token propagation in the area.

REFERENCES

- [1] C. Prehofer and C. Bettstetter, "Self-organization in communication networks: principles and design paradigms," *IEEE Communications Magazine*, vol. 43, no. 7, pp. 78–85, 2005.
- [2] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. F. Gryniewicz, and Y. Jin, "An architecture for a global internet host distance estimation service," in *IEEE INFOCOM*, New York, USA, 1999, pp. 210–217.
- [3] B. Krishnamurthy and J. Wang, "On network-aware clustering of web clients," in *ACM SIGCOMM*, Stockholm, Sweden, 2000, pp. 97–110.
- [4] A. Agrawal and H. Casanova, "Clustering hosts in p2p and global computing platforms," in *IEEE CCGrid*, Tokyo, Japan, May 2003, pp. 367–373.
- [5] O. Younis, M. Krunz, and S. Ramasubramanian, "Node clustering in wireless sensor networks: recent developments and deployment challenges," *IEEE Network Magazine*, vol. 20, no. 3, pp. 20–25, 2006.
- [6] R. Krishnan and D. Starobinski, "Efficient clustering algorithms for self-organizing wireless sensor networks," *Ad Hoc Networks*, vol. 4, no. 1, pp. 36–59, Jan. 2006.
- [7] D. J. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Transactions on Communications*, vol. 29, no. 11, pp. 1694–1701, Nov. 1981.
- [8] A. Ephremides, J. Wieselthier, and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," in *Proceeding of IEEE*, vol. 75, no. 1, Jan. 1987, pp. 56–73.
- [9] A. K. Parekh, "Selecting routers in ad-hoc wireless networks," in *Proceedings of IEEE ITS*, Rio de Janeiro, Brazil, Aug. 1994.
- [10] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal of Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, 1997.
- [11] S. Basagni, "Distributed clustering for ad hoc networks," in *Proceeding of International Symposium on Parallel Architectures, Algorithms and Networks*, Perth, Australia, Jun. 1999, pp. 310–315.
- [12] M. Chatterjee, S. Das, and D. Turgut, "Wca: A weighted clustering algorithm for mobile ad hoc networks," *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, vol. 5, no. 2, pp. 193–204, Apr. 2002.
- [13] R. Ramanathan and M. Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *ACM Mobile Networks and Applications*, vol. 3, no. 1, pp. 101–119, Jun. 1998.
- [14] Y. Fernandess and D. Malkhi, "K-clustering in wireless ad hoc networks," in *Proceedings of ACM POMC*, Toulouse, France, Oct. 2002, pp. 31–37.
- [15] G. Chen, F. Nocetti, J. Gonzalez, and I. Stojmenovic, "Connectivity-based k-hop clustering in wireless networks," in *Proceedings of HICSS*, vol. 7, Big Island, HI, USA, Jan. 2002, p. 188.3.
- [16] G. Venkataraman, S. Emmanuel, and S. Thambipillai, "Size-restricted cluster formation and cluster maintenance technique for mobile ad hoc networks," *International Journal of Network Management*, vol. 17, no. 2, pp. 171–194, Mar. 2007.
- [17] H.-C. Lin and Y.-H. Chu, "A clustering technique for large multihop mobile wireless networks," in *Proceedings of IEEE VTC*, vol. 2, Tokyo, Japan, May 2000, pp. 1545–1549.
- [18] N. Mitton, A. Busson, and E. Fleury, "Self-organization in large scale ad hoc networks," in *Proceedings of MedHocNet*, Bodrum, Turkey, Jun. 2004.
- [19] C. V. Ramamoorthy, A. Bhide, and J. Srivastava, "Reliable clustering techniques for large, mobile packet radio networks," in *IEEE INFOCOM*, San Francisco, USA, 1987, pp. 218–226.
- [20] F. Ben Abdesslem, A. Ziviani, M. Dias de Amorim, and P. Todorova, "Looking around first: Localized potential-based clustering in spontaneous networks," *IEEE Communications Letters*, vol. 11, no. 8, pp. 653–655, Aug. 2007.
- [21] "The Network Simulator NS-2," <http://www.isi.edu/nsnam/ns/>.
- [22] E. B. Hamida, G. Chelius, and J.-M. Gorce, "Scalability versus accuracy in physical layer modeling for wireless network simulations," in *22nd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS)*, Rome, Italy, Jun. 2008.