Distributed and Mobility-Adaptive Clustering for Multimedia Support in Multi-Hop Wireless Networks

Stefano Basagni

Center for Advanced Telecommunications Systems and Services (CATSS)
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
e-mail: basagni@utdallas.edu

Abstract – A distributed algorithm is presented that partitions the nodes of a fully mobile network (*multi-hop* network) into *clusters*, thus giving the network a hierarchical organization. The algorithm is proven to be adaptive to changes in the network topology due to nodes' mobility and to nodes addition/removal. A new weight-based mechanism is introduced for the efficient cluster formation/maintenance that allows the cluster organization to be configured for specific applications and adaptive to changes in the network status, not available in previous solutions. Specifically, new and flexible criteria are defined that allow the choice of the nodes that coordinate the clustering process based on mobility parameters and/or their current status. Simulation results are provided that demonstrate up to an 85% reduction on the communication overhead associated with the cluster maintenance with respect to techniques used in clustering algorithms previously proposed.

I. INTRODUCTION

In this paper we introduce a distributed and mobility-adaptive protocol that partitions the nodes of a *multi-hop wireless network*, i.e., a wireless network in which possibly all nodes can be mobile, into groups (*clusters*), thus giving the network a hierarchical organization.

The notion of cluster organization, often called *clustering*, has been used for multi-hop networks since their appearance (when these networks were often called "multi-hop packet radio networks," see, e.g., [1, 2]). With the advent of multimedia communications, the use of the cluster architecture for multi-hop networks has been revisited by Gerla et. al. [3, 4]. Clustering has also been proven effective in supporting quality of service in multi-hop networks, as well as location procedures and virtual circuit management (see [5], where extensive references can also be found). It is thus important to define efficient clustering algorithms that meet the requirements of multimedia applications in multi-hop networks while imposing the minimum clustering management overhead due to network mobility.

Previous solutions for clustering a multi-hop network usually perform the clustering in two phases: clustering *set up* and clustering *maintenance*. The first phase is accomplished by choosing some nodes that act as coordinators of the clustering process (*clusterheads*). Then a *cluster* is formed by associating a clusterhead with some of its *neighbors* (i.e., nodes within the clusterhead's transmission range) that become the *ordinary nodes* of the cluster. A common assumption for the set up phase is that the nodes *do not move* while the cluster formation is in progress. This is a major drawback, since in real situations, no assumptions can be made on the mobility of the nodes.

Once the nodes are partitioned into clusters, the non mobility assumption is released, and techniques are described on how to maintain the cluster organization in the presence of mobility (*clustering maintenance*). Among these techniques, some are based on a periodical reorganization of the clusters [1]. Of course, during the re-clustering process the network cannot rely on the cluster organization. Thus, this is a feasible solution only when the network does not need too many reorganizations (or when the network is not mobile). Other techniques (such as, e.g., the one used in [3]) need each node to know the identity of its neighbors up to *two* hops, which increases the overhead due to clustering maintenance. Moreover, the obtained clustering has different properties with respect to the initial one.

In [6] a distributed clustering algorithm is presented that overcomes the above mentioned limitations and that is suitable for both the clustering set up and maintenance. The Distributed and Mobility-Adaptive Clustering (DMAC) algorithm generalizes existing solutions by allowing the selection of the clusterheads based on nodes' mobility-related parameters expressed by generic *weights* associated to each node (instead of using a node's identifier or the number of its current neighbors). The idea is that with the weights we can express how suitable a node is for the role of clusterhead given its own current status: The bigger a node's weight, the more suitable it is for the role of clusterhead. (Of course, weights can change in time, reflecting the changing conditions in a node's status.) Similar to existing solutions, however, the clusterheads are bound to never be neighbors. This implies that, when due to the mobility of the nodes two or more clusterheads become neighbors, those with the smaller weights have to *resign* and affiliate with the now bigger neighboring clusterhead. Furthermore, when a clusterhead

v becomes the neighbor of an ordinary node u whose current clusterhead has weight smaller than v's weight, u has to affiliate with (i.e., *switch* to the cluster of) v. These "resignation" and "switching" processes due to nodes' mobility are a consistent part of the clustering management overhead that should be minimized.

In this paper we introduce a generalization of the DMAC protocol that aims to overcome the DMAC limitations while retaining its desirable properties. In particular, as for the DMAC algorithm, in our *Generalized DMAC* (G-DMAC):

- Nodes can move, even during the clustering set up.
- A node decides its own role (clusterhead or ordinary node) solely knowing its current *one* hop neighbors.

Furthermore, for G-DMAC we also obtain the following properties:

- The number of clusterheads that are allowed to be neighbors is a parameter of the algorithm (*degree of independence*), and:
- A new weight-based criterion is defined that allows the nodes to decide whether to change (switch) its role or not depending on the current condition of the network.

The last two properties introduce the possibility for G-DMAC to be configured for the specific multi-hop network in which it operates. Indeed, two or more clusterheads should be allowed to be neighbors depending on specific network conditions and applications. The same reasoning applies when an ordinary node becomes a neighbor of another clusterhead: the criterion with which it chooses whether to affiliate with the new neighboring clusterhead or not should depend again on the current conditions of the network, and on the specific applications that use the cluster organization.

We demonstrate the advantage of our new weight-based setting by presenting simulation results that compare G-DMAC with a "lowest ID first" algorithm based on the one presented in [3]. We show up to 85% reduction on the communication overhead associated with cluster management overhead (measured, as mentioned, in terms of the number of *reaffiliations* and *elections* of new clusterheads due to nodes' mobility).

In the rest of this section we give some basic definitions that are used throughout the paper. We model a multi-hop network by an undirected graph $G = (V, E)$ in which V, $|V| = n$, is the set of (wireless) nodes and there is an edge $\{u, v\} \in E$ if and only if u and v can mutually receive each others' transmission. In this case we say that u and v are neighbors. The set of the neighbors of a node $v \in V$ will be denoted by $\Gamma(v)$. Due to mobility, the graph can change in time.

Every node v in the network is assigned a unique identifier (ID). For simplicity, here we identify each node with its ID and we denote both with v. Finally, we consider weighted networks, i.e., a weight w_v (a real number ≥ 0) is assigned to each node $v \in V$ of the network. For the sake of simplicity, in this paper we stipulate that each node has a different weight.

Clustering a multi-hop network means partitioning its nodes into *clusters*, each one with a *clusterhead* and (possibly) some *ordinary nodes*. The choice of the clusterheads here is based on the *weight* associated to each node: the bigger the weight of a node, the better that node for the role of clusterhead. The process of cluster formation/maintenance is continuously executed at each node, and each node decides its own role so that the following three requirements (that we call "multi-hop clustering properties") are satisfied:

1. Every ordinary node always affiliates with (only) one clusterhead.
2. For every ordinary node v there is no clusterhead $u \in \Gamma(v)$ such that $w_u > w_{Clusterhead} + h$, where *Clusterhead* indicates the current clusterhead of v.
3. A clusterhead cannot have more than k neighboring clusterheads (k being an integer, $0 \leq k < n$).

Requirement number 1. ensures that each ordinary node has direct access to at least one clusterhead (the one of the cluster to which it belongs), thus allowing fast intra- and inter-cluster communications. The second requirement guarantees that each ordinary node always stays with a clusterhead that can give it a "guaranteed good" service. By varying the threshold parameter h (a real number ≥ 0) it is possible to reduce the communication overhead associated to the passage of an ordinary node from its current clusterhead to a new neighboring one when it is not necessary. Finally, requirement number 3. allows us to have the number of clusterheads that can be neighbors as a parameter of the algorithm. This, as seen for requirement number 2., and as will be demonstrated by the use of simulations, allows us to consistently reduce the communication overhead due to the change of role of nodes.

The rest of the paper is organized as follows. In the next section we describe the G-DMAC algorithm in details and we prove that the multi-hop clustering properties are always satisfied. The paper concludes with simulation results.

II. GENERALIZED DISTRIBUTED MOBILITY-ADAPTIVE CLUSTERING (G-DMAC)

In this section we describe a distributed algorithm for the set up and the maintenance of a cluster organization in the presence of nodes' mobility that satisfies the three properties listed in the previous section.

We start by making the following two common assumptions:

1. A message sent by a node is received correctly within a finite time (a *step*) by all its neighbors.
2. Each node knows its own ID, its weight, its role (if it has already decided its role: either a clusterhead or an ordinary node) and the ID, the weight and the role of all its neighbors (if they have already decided their role). When a node has not yet decided what its role is going to be, it is considered as an ordinary node.

The algorithm is executed at each node in such a way that at a certain time a node v decides (to change) its role. *This decision is entirely based on the decision (i.e., the role) of the nodes* $u \in \Gamma(v)$ *such that* $w_u > w_v$.

Except for the initial procedure, the algorithm is message driven: a specific procedure will be executed at a node depending on the reception of the corresponding message. We use three types of messages that are exchanged among the nodes: $\text{CH}(v)$, used by a node $v \in V$ to make its neighbors aware that it is going to be a clusterhead, $\text{JOIN}(v, u)$, with which a node v communicates to its neighbors that it will be part of the cluster whose clusterhead is node $u \in \Gamma(v)$, $v, u \in V$, and $\text{RESIGN}(w)$ that notifies a clusterhead whose weight is $\leq w$ that it has to resign its role. In the discussion below we use the following notation:

• v, the generic node executing the algorithm (from now on we will assume that v encodes not only the node's ID but also its weight w_v);

• *Cluster*(v), the set of nodes in v's cluster. It is initialized to $\emptyset$, and it is updated only if v is a clusterhead;

• *Clusterhead*, the variable in which every node records the (ID of the) clusterhead that it joins. It is initialized to **nil**;

Furthermore:

• Every node is made aware of the failure of a link, or of the presence of a new link by a service of a lower level (this will trigger the execution of a corresponding procedure).

• The procedures of G-DMAC (M-procedures, for short) are "atomic," i.e., they are not interruptible.

The following two rules define how the nodes assume/change their roles adapting to changes in the network topology.

1. Each time a node v moves into the neighborhood of a clusterhead u with a bigger weight, we require that v switches to u's cluster only if $w_u > w_{Clusterhead} + h$, where *Clusterhead* is the clusterhead of v (it can be v itself) and h is a real number ≥ 0. This should happen independently of the current role of v. With this rule we want to model the fact that we incur the switching overhead only when it is really convenient. When $h = 0$ we simply obtain that each ordinary nodes affiliates with the neighboring clusterhead with the biggest weight.

2. We allow a clusterhead v to have up to k neighboring clusterheads, $0 \leq k < n$. We call this condition the k-neighborhood condition. Choosing $k = 0$ we obtain that no two clusterheads can be neighbors (maximum degree of independence).

The parameters h and k can be different from node to node, and they can vary in time. This allows G-DMAC to self-configure dynamically in order to meet the specific needs of upper layer applications/protocols that requires an underlying clustering organization. At the same time, different values of h and k allow our algorithm to take into account dynamically changing network conditions, such as the network connectivity (related to the average nodal degree, i.e., to the average number of the neighbors of the nodes), variations in the rate of the mobility of the nodes, etc. Notice that the case with $h = k = 0$ corresponds to the DMAC protocol introduced in [6].

The following is the informal description of the six M-procedures. Details and pseudo codes can be found in [7].

• *Init.* At the clustering set up, or when a node v is added to the network, it executes the procedure *Init* in order to determine its own role. If among its neighbors there is at least a clusterhead with bigger weight, then v will join it. Otherwise it will be a clusterhead. In this case, the new clusterhead v has to check the number of its neighbors that are already clusterheads. If they exceed k, then a RESIGN message is also transmitted, bearing the weight of the first clusterhead (namely, the one with the $(k+1)$th biggest weight) that violates the k-neighborhood condition. On receiving a message $\text{RESIGN}(w)$, every clusterhead u such that $w_u \leq w$ will resign. Notice that a neighbor with a bigger weight that has not decided its role yet (this may happen at the clustering set up, or when two or more nodes are added to the network at the same time), will eventually send a message (every node executes the *Init* procedure). If this message is a CH message, then v could possibly resign (after receiving the corresponding RESIGN message) or affiliate with the new clusterhead.

• *Link_failure.* Whenever made aware of the failure of the link with a node u, node v checks if its own role is clusterhead and if u used to belong to its cluster. If this is the case, v removes u from *Cluster*(v). If v is an ordinary node, and u was its own clusterhead, then it is necessary to determine a new role for v. To this aim, v checks if there exists at least a clusterhead $z \in \Gamma(v)$ such that $w_z > w_v$. If this is the case, then v joins the clusterhead with the bigger weight, otherwise it becomes a clusterhead. As in the case of the *Init* procedure, a test on the number of the neighboring clusterheads is now needed, with the possible resigning of some of them.

• *New_link.* When node v is made aware of the presence of a new neighbor u, it checks if u is a clusterhead. If this is the case, and if w_u is bigger than the weight of v's current clusterhead plus the threshold h, than, independently of its own role, v affiliates with u. Otherwise, if v itself is a clusterhead, and the number of its current neighboring clusterheads is $> k$ then the weight of the clusterhead x that violates the k-neighborhood condition is determined. If $w_v > w_x$ then node x has to resign, otherwise, if no clusterhead x exists with a weight smaller than v's weight, v can no longer be a clusterhead, and it will join the neighboring clusterhead with the biggest weight.

• *On receiving* $\text{CH}(u)$. When a neighbor u becomes a clusterhead, on receiving the corresponding CH message, node v checks if it has to affiliate with u, i.e., it checks whether w_u is bigger than the weight of v's clusterhead plus the threshold h or not. In this case, independently of its current role, v joins u's cluster. Otherwise, if v is a clusterhead with more than k neighbors which are clusterheads, as in the case of a new link, the weight of the clusterhead x that violates the k-neighborhood condition is determined, and correspondingly the clusterhead with the smallest weight will resign.

• *On receiving* JOIN(u,z). On receiving the message JOIN(u,z), the behavior of node v depends on whether it is a clusterhead or not. In the affirmative, v has to check if either u is joining its cluster ($z = v$: in this case, u is added to *Cluster*(v)) or if u belonged to its cluster and is now joining another cluster ($z \neq v$: in this case, u is removed from *Cluster*(v)). If v is not a clusterhead, it has to check if u was its clusterhead. Only if this is the case, v has to decide its role: It will join the biggest clusterhead x in its neighborhood such that $w_x > w_v$ if such a node exists. Otherwise, it will be a clusterhead. In this latter case, if the k-neighborhood condition is violated, a RESIGN message is transmitted in order for the clusterhead with the smallest weight in v's neighborhood to resign.

• *On receiving* RESIGN(w). On receiving the message RESIGN(w), node v checks if its weight is $\leq w$. In this case, it has to resign and it will join the neighboring clusterhead with the biggest weight. Notice that since the M-procedures are supposed to be not interruptible, and since v could have resigned already, it has also to check if it is still a clusterhead.

We conclude this section by showing that by using the M-procedures we obtain and maintain for any multi-hop network a clustering that always satisfies the multi-hop clustering properties listed in the Introduction (a more detailed proof based on the actual procedure pseudo-code can be found in [7]).

Theorem 1 *Using the M-procedures, any multi-hop network is clustered in such a way that the multi-hop clustering properties are always satisfied.*

Proof We start by noticing that as soon as a node has executed the *Init* procedure as described above, it is always assigned a role which is consistent with the clustering properties. We then proceed by showing that by executing the M-procedures in reaction to changes in the network topology, the nodes assume/change their roles so that the multi-hop clustering properties are always satisfied.

1. That each ordinary node v does not affiliate with more that one clusterhead is evident by noticing that anytime it sends a JOIN(v,u) message its variable *Clusterhead* is initialized (only) to u. That v affiliates with at least a clusterhead derives from the fact that when it has to decide its role and there are clusterheads with bigger weights among its neighbors, or when a switch to another cluster is required, the ordinary node v always looks for the clusterhead with the biggest weight and affiliates with it. Thus, there is no case in which an ordinary node v remains without a clusterhead.

2. At the clustering set up, or when a node is added to the (already clustered) network, or when its current clusterhead either resigns or moves away, or, finally, when it is forced to resign, a node always affiliates with the clusterhead with the biggest weight (if there is no such clusterhead, it will become a clusterhead itself), so that the second multi-hop clustering property is always satisfied. The other cases to consider are when an ordinary node v switches from a cluster to another, or when v is a clusterhead that resigns to join the cluster of a new neighboring clusterhead. In these cases, the second property is guaranteed by the procedures *New_Link* and On Receiving CH, where node v switches to u's cluster only if $w_u > w_{Clusterhead} + h$.

3. Each time a node becomes a clusterhead, i.e., it transmits a CH message, it does so because there is no other neighboring clusterhead with which it can affiliate. In this case, it always checks if it has more than k neighboring clusterheads, and if this is the case, it decides (on a weight basis) the clusterheads that have to resign. When these clusterheads receive the corresponding message RESIGN, they have no choice but to resign, joining the clusterhead with the biggest weight around them. The other cases that remain to be checked are when either a clusterhead v has one of its neighbors that becomes a clusterhead, or a clusterhead moves into its neighborhood, and the weight of the new neighbor does not force v to affiliate with it. In both these cases, the third multi-hop clustering property is guaranteed by checking if, upon the arrival of a new clusterhead, the k-neighborhood condition is violated. If this is the case, the clusterhead with the minimum weight is determined, and the corresponding RESIGN message is transmitted. •

III. SIMULATION RESULTS

Here we demonstrate by the use of simulations that G-DMAC achieves substantial improvements with respect to algorithms based on techniques for the selections of the clusterheads introduced in earlier works.

We simulated our algorithm by placing $n = 30$ nodes randomly on a grid of size 100×100. The speed s_v and the power p_v of every node v are given in grid units. Two nodes v and u in the network are neighbors if the Euclidean distance d_g between their coordinates in the grid is less than the minimum between their transmission radii (i.e., $d_g(v,u) < \min\{p_v, p_u\}$).

At every tick of the simulation clock, each node determines its direction randomly, by choosing it uniformly between 0 and 2π. Each node will then move in that direction according to its current speed. When a node reaches the bounds of the grid, it bounces back with an angle determined by the incoming direction.

We define the *stability* of a mobility-adaptive clustering algorithm in terms of the number of *elections*

		$k = 0$	$k = \frac{n}{10}$	$k = \frac{n}{6}$	$k = \frac{n}{3}$
VARIABLE SPEED (constant power = 30)	elections	$0 - 19$	$50 - 73$	$70 - 78$	$71 - 81$
	reaffiliations	$26 - 58$	$53 - 78$	$55 - 83$	$55 - 88$
VARIABLE SPEED (constant power = 40)	elections	$0 - 8$	$63 - 73$	$76 - 80$	$70 - 82$
	reaffiliations	$28 - 64$	$53 - 81$	$55 - 83$	$57 - 88$
VARIABLE POWER (speed $\in [0, 6]$)	elections	*loss* (< 10)	$50 - 73$	$53 - 74$	$56 - 74$
	reaffiliations	$38 - 53$	$72 - 79$	$75 - 80$	$77 - 80$
VARIABLE POWER (constant speed = 6)	elections	*loss* (< 10)	$49 - 75$	$52 - 75$	$55 - 75$
	reaffiliations	$40 - 53$	$70 - 79$	$72 - 82$	$74 - 82$

Table 1 G-DMAC vs. "lowest ID first:" min-max percentage gain in networks with 30 nodes.

and *reaffiliations* per tick. An election occurs all the times that, due to a change in the topology of the network, either an ordinary node or a node that becomes alive decides to be a clusterhead. We have a reaffiliation when either an ordinary node or a clusterhead, or a node that becomes alive, affiliates as an ordinary node with a newly close clusterhead.

G-DMAC is compared here with the clustering algorithm based on the "lowest ID first" approach presented in [3, 4]. In this algorithm the node with the minimum ID in its current neighborhood is chosen as a clusterhead. Since in [4] it is shown that the "lowest ID first" clustering is always more stable than the clustering obtained following a "largest degree first" approach, we do not consider the latter one.

In the first set of simulations the weight associated to a node represents its speed: the slower a node is, the better it is for the role of clusterhead. We define the weight of the node v by $w_v = x+1-s_v$, where the speed s_v is chosen uniformly in $[1, x]$, $x \in [1, 100]$. In this way, for each x, the nodes that are moving at lower speed are assigned bigger weights. The percentage gains obtained for elections and reaffiliations by G-DMAC with respect to the "lowest ID first" algorithm when all the nodes $v \in V$ have constant power $p_v = 30$ and $p_v = 40$ are shown in Table 1 for four different values of k. The choice of these two values for the power is motivated by the fact that with values between 30 and 40 the network is always guaranteed to be connected.

In the other set of simulations, weights represent nodes' transmission power: the bigger the power of a node, the better that node for the role of clusterhead. If $x \in [10, 100]$ indicates the maximum transmission power, we initialize each node with a power p_v chosen with uniform distribution in the range $[5, x]$. The weight of the node v is then $w_v = p_v$ (in this way, for each x, the nodes with the bigger transmission powers are assigned bigger weights). The third and fourth rows of Table 1 show the percentage gains for elections and reaffiliations when the speed s_v of each node $v \in V$ varies in the range $[0, 6]$ and it is constantly $= 6$, respectively.

In all the simulations $h = \frac{y_{max} - y_{min}}{2}$, where $y_{min} = 1$ for the simulations with variable speed and $y_{min} = 5$ for the simulations with variable power, and $y_{max} = x$. The confidence level of our results is 95% and their precision is 5%.

IV. ACKNOWLEDGMENTS

This paper was supported in part by the 1998/1999 TxTEC Consortium Fellowship and by the Army Research Office under contract No. DAAG55-96-1-0382. The author is also pleased to thank Marco Monguzzi for his collaboration in the earlier stages of this research.

REFERENCES

(1) D. J. Baker, A. Ephremides, and J. A. Flynn, "The design and simulation of a mobile radio network with distributed control," *IEEE Journal on Selected Areas in Communications*, vol. SAC-2, pp. 226–237, January 1984.

(2) A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," *Proceedings of the IEEE*, vol. 75, pp. 56–73, January 1987.

(3) C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *Journal on Selected Areas in Communications*, vol. 15, pp. 1265–1275, September 1997.

(4) M. Gerla and J. T.-C. Tsai, "Multicluster, mobile, multimedia radio network," *Wireless Networks*, vol. 1, no. 3, pp. 255–265, 1995.

(5) R. Ramanathan and M. Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *Mobile Networks & Applications*, vol. 3, pp. 101–119, June 1998.

(6) S. Basagni, "Distributed clustering for ad hoc networks," in *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, (Perth/Fremantle, Australia), pp. 310–315, IEEE Computer Society, June 23–25 1999.

(7) S. Basagni, "Distributed and mobility-adaptive clustering for ad hoc networks," Tech. Rep. UTD/EE-02-98, Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas, July 1998.