On the Message and Time Complexity of a Distributed Mobility–Adaptive Clustering Algorithm in Wireless Ad Hoc Networks

Christian Bettstetter and Stefan König
Technische Universität München (TUM), Institute of Communication Networks
D–80290 Munich, Germany, Christian.Bettstetter@ei.tum.de

Abstract—*This paper investigates the message and time complexity of a distributed mobility–adaptive clustering algorithm for wireless ad hoc networks. Such algorithms are used to dynamically organize the mobile stations of the network into groups and hierarchies. We discuss the reaction of the distributed algorithm on three types of changes in the network topology: appearance of new stations, failures of links, and formation of new links. In our analysis we discover that a single event may cause a restructuring "chain reaction" through the network. It occurs along a path that exhibits certain characteristics. The properties of this path are derived in this paper. Finally, we set up three design criteria for the development of good clustering algorithms.*

Keywords— *Wireless mobile ad hoc networks, network self–organization, distributed algorithms, adaptability, adaptive clustering.*

I. Introduction and Motivation

Clustering algorithms are applied in communication networks to organize all entities into groups and to obtain a hierarchical network organization. Figure 1 gives an abstract example for clustering, in which 12 network entities are organized into three clusters and two hierarchy levels.

Such methods are employed e.g. in Internet–based networks, cellular networks, and ATM networks using the Private Network–to–Network Interface (PNNI). They are also an essential part in the protocol suite of wireless multi–hop networks, so–called ad hoc networks. In general, ad hoc networks do not rely on any fixed network infrastructure (e.g., base stations). They allow direct, wireless peer–to–peer communication between the mobile stations, and each station can act as a router to forward traffic toward its destination (wireless multi–hop routing). The set of stations organizes itself in a distributed fashion. The radio technology Bluetooth, for example, implements a clustering principle in order to form wireless "piconets" of devices. Clustering in such highly dynamic and self–organizing wireless networks is the topic of this paper.

Let us briefly explain the general benefits and drawbacks of clustering. One of the major goals of a hierarchical network organization is to achieve scalability (see e.g. [1]). If we used a flat (non–clustered) structure in large networks, routing tables and location registers would grow to an immense size. Each router would have to maintain an immense topology knowledge with relevant characteristics (link capacities, addresses, etc.). Its memory and processing power would be exceeded, and table lookup times would increase.

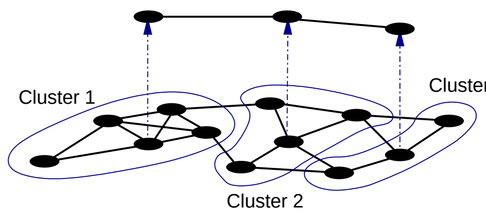

Fig. 1. Clustered network organization and virtual hierarchical topology

A clustered architecture, however, limits the view of each network entity to a fraction of the total network (see Fig. 2). Detailed topology information is only exchanged among local cluster members, and aggregated (reduced) information is distributed between neighboring groups in the higher hierarchy level. This principle reduces the signaling traffic exchanged between network entities and the information stored inside them. A clustered organization is also useful for radio resource management, hierarchical address assignment, and medium access control.

The disadvantage of a clustered network organization is that additional signaling traffic is needed in order to maintain the cluster structure. Moreover, some nodes have additional work load (processing, aggregation of topology information, packet forwarding) and might become a bottleneck in the system.

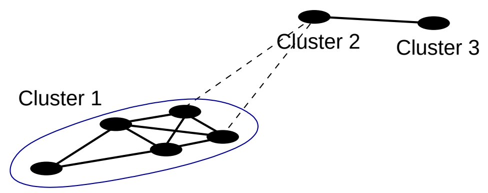

Fig. 2. Restricted topology view of ordinary stations in Cluster 1

Several algorithms for clustering have been proposed in the literature for fixed and mobile communication networks. The choice of the algorithm for a particular network type strongly depends on the dynamic nature of the network. Whereas in fixed networks a centrally organized and rather static network organization is sufficient, in highly dynamic ad hoc networks a completely distributed and adaptive clustering protocol is essential. Paper [2] contains a survey of algorithms proposed for ad hoc networks.

This work is funded by the German research foundation DFG within the program AKOM (Adaptability in heterogeous communication networks with wireless access).

In this paper, we investigate the *Distributed Mobility–Adaptive Clustering* (DMAC) algorithm for wireless ad hoc networks, which has been proposed by Basagni in [3]. We analyze the message and time complexity of this algorithm. Both criteria have significant importance for the performance of clustering, since they determine the signaling traffic as well as the capability of the algorithm to react to topology changes quickly.

The rest of this paper is as follows: Section II introduces our example network and describes the DMAC algorithm. In Section III we define our criteria for investigation. Section IV presents our results, i.e., the reaction of the clustering algorithm on changes in the network topology and the resulting message and time complexity. We consider three typical events (addition of new network stations, link failures, and the appearance of new links) and derive design rules for the development of adaptive clustering algorithms. Finally, Section V concludes this paper.

II. Example Topology and DMAC Clustering

A. Example Topology and Modeling Assumptions

Figure 3 shows our example network topology that is used throughout the paper to explain the clustering process. Mobile stations are represented by nodes. Each node can be identified by a unique identifier i and has a weight w_i (displayed in brackets). For example, Node 5 has weight $w_5 = 8$. These weights are needed for the clustering process and may be assigned randomly or according to certain characteristics of the station (e.g., addresses, battery power, maximum speed).

The wireless links between the stations are represented as edges between the nodes. Two nodes can communicate with each other directly, if and only if there is a link between them. In this paper we only consider bidirectional links, i.e., we simply ignore unidirectional links.

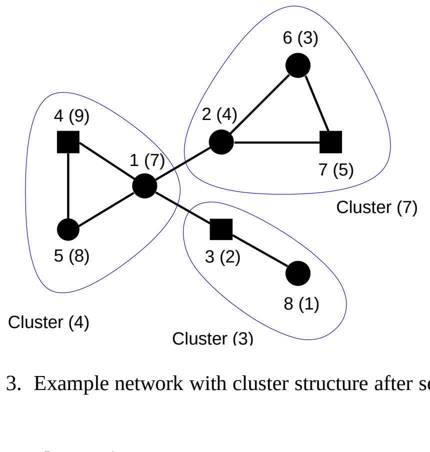

Fig. 3. Example network with cluster structure after setup

B. DMAC Clustering

The Distributed Mobility–Adaptive Clustering (DMAC) algorithm [3], which is subject to evaluation in this paper, is based on the concept of a *clusterhead*. This exposed node acts as a local coordinator within the cluster and can perform aggregation of topology information and exchange to neighboring clusters.

When a node is added to the network, it executes an initialization process to determine its *role*, i.e., whether it should become a clusterhead or an ordinary node. This decision is based on the node's weight w_i and the weights of its neighbors: the larger the weight of a node, the better it is suited as a clusterhead. It is assumed that each node has a different weight. A node decides to join a cluster if there is already a neighboring clusterhead with a higher weight; otherwise it decides to become a clusterhead itself. After making this decision, it immediately informs all its neighbors of its role (by sending out either a JOIN or CLUSTERHEAD message). The algorithm is executed at each node and is message driven. In order to react properly and consistently, each node has to know the following values: its own identifier, weight, and role (if it is already determined), as well as the identifier, weight, and role (if already available) of each of its current neighbors.

To meet the requirements of the network, a valid cluster structure has to be reached after setup and each topology change. Such a stable condition is defined by the following three properties, which must be fulfilled [3]: First, every ordinary node has at least one clusterhead as neighbor. Second, every ordinary node affiliates with the neighboring clusterhead with the largest weight. And, third, clusterheads cannot be neighbors.

Fig. 3 shows the resulting DMAC cluster structure of the example network. Three clusters have formed, and the clusterheads are Nodes 4, 7, and 3 (displayed as squares).

Mobility–adaptive clustering in an ad hoc network is a continuously running online process. As the mobile stations move around, they must decide to which cluster they currently belong and which role they have. In order to be adaptive to mobility, each node reacts to changes in the surrounding topology (e.g., failure of links, appearance of new links), and it changes its status and cluster membership accordingly. This decision is based only on the local view of their neighborhood. Whenever a link failure happens, each node checks if its own role is clusterhead and if the other node belongs to its cluster. In this case, the clusterhead removes the node from its cluster. When the link of an ordinary node to its clusterhead fails, the ordinary node must determine its new role in the same way as it does during initialization. A new link between two nodes is handled in a similar way. A JOIN message is sent out when the new neighbor is a clusterhead with higher weight. Otherwise, the information is ignored since the cluster properties are still fulfilled. The information that is needed to trigger the execution of the corresponding procedure, e.g. that a link failed, is provided by a lower protocol layer. The processing of the JOIN and CLUSTERHEAD messages is performed by the corresponding ONRECEIVING( ) procedure.

III. Criteria For Investigation and Definition of Message and Time Complexity

Finding good criteria for performance evaluation of adaptive clustering algorithms is not a trivial task, and we propose the criteria listed in Table I. Let us analyze how the

Table I: Evaluation Criteria for Clustering Algorithms

Message complexity	How many signaling messages are needed to form groups and to decide which node becomes clusterhead?
Time complexity	How long does it take the algorithm after a change in the topology to re–achieve a valid cluster structure?
Load on nodes	How does the traffic and processing load on nodes change using a hierarchical organization (e.g., link state advertisements, route computations, topology aggregation)?
Routing table size and routing optimality	A hierarchical network representation reduces the number of routing entries (compared to a flat routing). However, the number of alternative paths available is smaller, and the information about remote network areas is vague.
Cluster stability	How stable does the clustering algorithms maintain the cluster structure while the network topology changes?
Level of adaptability	How adaptive is the algorithm to changes in the network? Which parameters are adaptive?
Decision speed	How much knowledge of its neighbors does a node need to make a decision about its role?
Asynchronous operation	Is the algorithm capable of running in an asynchronous way, or does it need synchronization?

DMAC algorithm is rated with these parameters. Concerning adaptability, DMAC reacts to changes in the network topology due to mobility of nodes and node outages (failure and recovery). However, it does not change the cluster size as a reaction to mobility, as the algorithm described in [4] does. Concerning decision speed, a node must know the identifier, weight, and role of all its *one*–hop neighbors in order to decide its own role in the cluster structure. Concerning asynchronous operation, we can say that the DMAC algorithm does not need any time synchronization.

Another important criterion is the cluster stability with respect to the dynamic behavior of the network. A good clustering algorithm should be designed to maintain its cluster structure as stable as possible while the topology changes [5]. In other words, the algorithm should minimize the number of node transitions from one cluster to another. Two types of events must be considered for cluster stability: (a) *election events*, i.e., all events in which a node becomes a clusterhead; and (b) *join events*, i.e., events in which an ordinary node, a clusterhead, or a new node affiliates with a (different) clusterhead [6]. In order to increase the cluster lifetime and the dwell time of nodes in a cluster, these events should be minimized. One reason for this design goal is that each event may trigger a restructuring process of the cluster structure that causes signaling overhead.

In two other papers, [2] and [7], we have investigated the stability of DMAC by simulation. The topic of this paper is the number of signaling messages and the time it takes the algorithm after such an event to re–achieve a valid cluster structure. Instead of performing simulations we take an analytical approach, in which we consider isolated topology change events and investigate the reaction of the clustering algorithm in detail. We are interested in finding bounds for the message and time complexity, defined as follows:

Definition 1: We define the *message complexity* M of a distributed clustering algorithm as the number of messages exchanged between nodes after a change in the topology until a valid clustering structure is re–achieved. If a node has several neighbors, a message sent to these neighbors will increase the message complexity only by one.

Definition 2: We assume discrete time steps, and define the *time complexity* T of a distributed clustering algorithm as the number of time steps it takes the algorithm after a change in the topology to re–achieve a valid clustering structure. We define *one time step* as the time between the sending of a message and the complete processing of the message on the receiver side. This includes the transmission time over the radio link and the processing time (e.g., updating of member lists) in the receiver node. This time complexity definition makes only sense for time–synchronized systems. Although DMAC is capable of operating in an asynchronous manner, we operate the algorithm here in a synchronous environment.

IV. Analysis of Message and Time Complexity

Our approach is to consider three fundamental types for a topology change in an ad hoc network: (1) a new node appears (e.g., because a mobile station is switched on), (2) a link between two nodes fails (e.g., because two nodes move away from each other or the radio propagation characteristics worsen), and (3) a new link between two nodes is established (e.g., because two nodes move toward each other).

At the beginning of each section, we consider our example network, which gives us a better understanding of how clustering works. Then, we try to derive general statements for the message and time complexity.

A. Adding a New Node

A.1 Example Scenario

Let us investigate a scenario in which a new node is added to our example network of Figure 3. Node 9 with weight $w_9 = 6$ joins the network with links to Node 1 and 3. It performs an initialization process and checks its neighboring nodes to determine their role and weight. Since there is no neighboring clusterhead with higher weight, Node 9 decides to become clusterhead itself. It informs its one–hop neighbors of the decision by sending a CLUSTERHEAD(9) message. Both neighbors receive and process this message in timestep 1. Node 1 ignores the message, since it has a clusterhead with higher weight (Node 4 with weight $w_4 = 9$).

Node 3 is clusterhead but has a lower weight than Node 9. Thus, it must change its role to become an ordinary node. It joins the cluster of Node 9 and sends out a JOIN(3,9) message to all its neighbors (time step 2). On receiving this message, Node 8 becomes its own clusterhead (see Fig. 4). Node 1 does not react, since it has a clusterhead with higher weight. Node 9 enters Node 3 in its list of member nodes. At this time step, Nodes 9 and 3 belong to Cluster(9); see Fig. 4. In the third time step, Node 3 receives CLUSTERHEAD(8) but ignores it. In summary, $M = 3$ messages have been sent in $T = 3$ time steps.

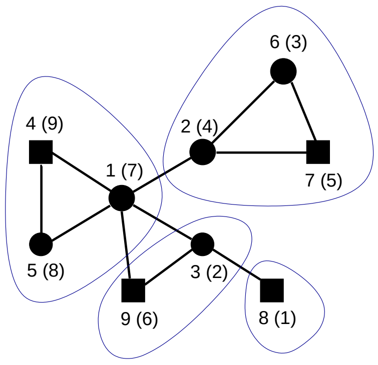

Fig. 4. Clustering structure after adding new node (Node 9)

A.2 General Analysis

With the basic understanding of this example, let us now try to derive a more general description for the message and time complexity after the appearance of a new node. The role decision of the new node and the further process of the algorithm is determined by the roles and weights of the neighboring nodes. We denote the number of neighbors of a node, i.e. its degree, as d, and distinguish four types of neighbors of an ordinary node: d_{hh} clusterhead neighbors with higher weight, d_{hl} clusterhead neighbors with lower weight, d_{oh} ordinary neighbors that belong to a clusterhead with higher weight, and d_{ol} ordinary neighbors that belong to a clusterhead with lower weight, where $d = d_{hh} + d_{hl} + d_{oh} + d_{ol}$ holds.

A trivial case occurs, if a new node i has no neighbors ($d_i = 0$). It becomes its own clusterhead and sends out a CLUSTERHEAD message. This process is finished after $T = 1$ time step.

If a new node has at least one neighboring node that is clusterhead with higher weight (i.e., if $d_{hh} > 0$), the new node will send a JOIN message during its initialization process. On receiving this message the clusterhead neighbor with the highest weight adds the node to its member list. One message ($M = 1$) and one time step ($T = 1$) is required, independent of the total degree d and the roles of the other nodes (see Table II, second line entry).

Let us now consider cases in which there is no clusterhead neighbor with higher weight (i.e., $d_{hh} = 0$). The new node sends out a CLUSTERHEAD message and becomes its own clusterhead.

First, we assume that only ordinary nodes are in the neighborhood of the new node, i.e., $d_{hh} = d_{hl} = 0$. All $(d_{oh} + d_{ol})$ ordinary neighbors receive the CLUSTERHEAD message and check whether the new node has a lower or higher weight than their current clusterhead. The d_{ol} nodes with lower weighted clusterheads join the new node, and the new node receives d_{ol} JOIN messages in the second time step. The former clusterheads also receive these messages and delete the nodes from their member lists. Ordinary nodes do not react upon receiving these JOIN messages. In summary, $M = 1 + d_{ol}$ messages are needed. The time complexity is $T = 2$ time steps. The d_{oh} nodes with higher weighted clusterheads do not react at all to the CLUSTERHEAD message of the new node, i.e., we obtain $M = 1$ and $T = 1$ for the special case $d_{ol} = 0$. In a mixed scenario, the value of d_{oh} has no influence on message and time complexity, i.e., we obtain $M = 1 + d_{ol}$ and $T = 2$. These results are again summarized in Table II, line 3–4.

Table II: Message and Time Complexities for “New Node” Event

Neighbor constellation				Role of	Complexities	
d_{hh}	d_{hl}	d_{oh}	d_{ol}	new node	M	T
0	0	0	0	cl.head	1	1
$\neq 0$	any	any	any	ordinary	1	1
0	0	d	0	cl.head	1	1
0	0	any	$\neq 0$	cl.head	$1 + d_{ol}$	2
0	$\neq 0$	any	$\neq 0$	cl.head	$\geq 1 + d_{hl}$ $+ d_{ol}$	≥ 2

Second, we allow that there is also at least one clusterhead with lower weight, i.e., $d_{hl} \neq 0$ while $d_{hh} = 0$. All d_{hl} clusterhead neighbors join the new cluster and thus d_{hl} JOIN messages are needed (in addition to the d_{ol} JOIN messages of the ordinary nodes). If none of the d_{hl} former clusterheads has any members, a valid cluster structure is obtained after $T = 2$ time steps with a message complexity of $M = 1 + d_{hl} + d_{ol}$. Otherwise (this is the general case), the JOIN messages are received by the members of the former clusterheads. Each of them must re–decide its role, either by joining a different cluster or by becoming a clusterhead. Thus, in any case, those nodes must send out one further message, either a JOIN or CLUSTERHEAD (third timestep). The JOIN messages do not trigger any further process. The CLUSTERHEAD messages, however, may indeed cause further events. Upon receiving a CLUSTERHEAD message, each node must again re–decide its role and check whether it should join the new clusterhead.

To sum up, we can say that the number of messages and the time needed to achieve a valid constellation after the addition of a new node is strongly dependent on the current topology, in particular on the degree and role of the new node's neighbors. Because of this fact, only lower bounds can be derived easily for the general case: at least $1 + d_{hl} + d_{ol}$ messages and one time step is needed. In this context, we made an interesting observation: In certain topologies, the appearance of a single node may even cause a re–clustering chain reaction. This is an undesirable characteristic of the algorithm and is worth deeper investigation.

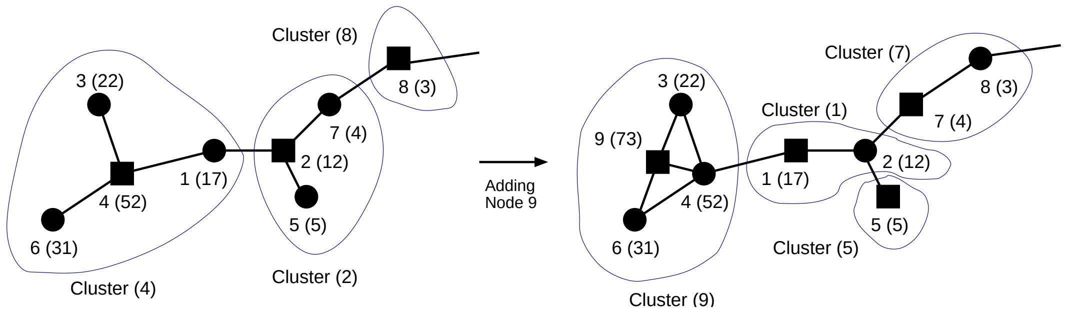

Fig. 5. Re–clustering chain reaction after adding a node (Node 9)

A.3 Re–clustering Chain Reaction

Figure 5 illustrates how such a chain reaction could look like in the worst case. The appearance of Node 9 with weight $w_9 = 73$ causes Node 4 to change its role. As a consequence of this, almost the entire network is affected: Nodes 1, 2, 5, 7, 8 must also change their role, and the time and message complexity is only limited by the number of nodes on the path of this chain reaction. The chain reaction proceeds along the path of Nodes 9, 4, 1, 2, 7, and 8. A more detailed investigation of this phenomenon yields the following statement:

Proposition: After the appearance of a new node in an ad hoc network that uses the DMAC algorithm for clustering, the cluster structure can change along a directed path (chain reaction), as long as the following conditions are fulfilled: (1) clusterheads and ordinary nodes must appear alternately in the path, (2) the successor of a node in the path has lower weight than the node itself, and (3) no ordinary node has a clusterhead with higher weight than its own predecessor in the path.

The chain reaction is triggered, if the new node has a higher weight than its neighboring clusterhead node. The length of the chain reaction is also determined by the three parameters, i.e., the re–structuring path will stop when one of the conditions is not fulfilled anymore.

Condition 1 is required, since two neighboring ordinary nodes will stop the chain reaction. Suppose, for example, the weights of Node 2 and 5 in Fig. 5 are switched, i.e., $w_2 = 5$ and $w_5 = 12$. In this case, Node 5 would be the clusterhead of Node 2, and Node 7 and 8 would form Cluster(7). Conditions 2 and 3 are still fulfilled for the known path. In the restructuring process, Node 2 would remain an ordinary node and join Cluster(1). After this JOIN event, no further messages will be processed. Condition 2 is needed because only in this case will successor nodes in the path change their role. We need Condition 3, because a node that has a clusterhead i with higher weight than its own predecessor clusterhead j in the path will remain member of its current clusterhead i and thus not produce any further events (see Table II, $d_{hh} \neq 0$).

Besides this phenomenon and its criteria, our investigations produce some design goals for the development of adaptive clustering algorithms.

A.4 Design Criteria

In DMAC, the message and time complexities are very low for join events, in which a node affiliates with an existing cluster. Only election events (in which a node becomes clusterhead) may cause high complexity. The resulting total signaling traffic over time depends on the fact how frequently a certain event occurs. As we have seen in our stability analysis in [2], in general, join events occur more frequently than election events for given simulation parameters. These frequently occurring join events produce only $M = 1$ signaling message, and a stable and valid structure is already achieved after $T = 1$ time step. On the other hand, clusterhead elections are costly but occur infrequently.

Design Goal 1: Frequent events (such as join events) should produce only low signaling effort and a low time complexity for fast convergence. High signaling effort and high time complexity should only occur for infrequent events (such as election events).

Design Goal 2: Clusterheads should be especially stable, because a change of their role may trigger many other events with high message and time complexity. It may even cause a re–clustering chain reaction.

The possibility of having a chain reaction in large parts of the network due to a local change is an ugly characteristic of the DMAC algorithm. The strategy for the allocation of weights to nodes plays an important role here. For example, a strategy that sets the weight of a node dynamically to its age (starting with $w = 0$) would avoid a new node having neighbors with lower weight. Since then $d_{hl} = d_{ol} = 0$, the re–clustering for a “new node” event would always be completed within $T = 1$ time step.

Design Goal 3: Reactions to topology changes in a distributed adaptive clustering algorithm should only be based on the local knowledge of a node’s neighborhood (e.g., 1–hop neighborhood). Moreover, they should only affect the neighborhood of the node (e.g., its n–hop neighborhood, where usually $n > 1$). *Keep changes local!*

This design goal is also motivated by the statement of McDonald in [4], that clustering in ad hoc networks makes a large network appear smaller (as in fixed networks), but even more important it can make a highly dynamic network appear less dynamic (on a higher hierarchy level).

B. Link Failure

If a link failure is reported from a lower layer protocol, the current cluster structure has to be checked on its validity. The two involved nodes will react to the topology change accordingly.

B.1 Example Scenario

Suppose there is a link failure between Node 1 and Node 4 in Fig. 3. Node 1 realizes that it no longer has direct connection to its clusterhead. It thus checks whether there is a neighboring clusterhead with higher weight than its own. Since this is not the case, it becomes a clusterhead and sends out a CLUSTERHEAD(1) message, which is received by Node 2, 3, and 5. Node 2 decides to join Cluster(1), since it had a clusterhead with lower weight than that of Node 1 ($w_7 < w_1$). It sends out a JOIN(2,1) message. Node 3 also joins Cluster(1), since its own weight is lower than that of Node 1 ($w_3 < w_1$). It sends out a JOIN(3,1) message. Node 5 ignores the CLUSTERHEAD(1) message. Node 1 adds Node 2 and 3 to its cluster, and Node 7 deletes Node 2 from its member list. Node 8 receives JOIN(3,1) and becomes clusterhead. Finally, Node 3 ignores the CLUSTERHEAD(8) message of Node 8. The resulting cluster structure is shown in Fig. 6. In summary, $T = 3$ time steps and $M = 4$ messages are needed.

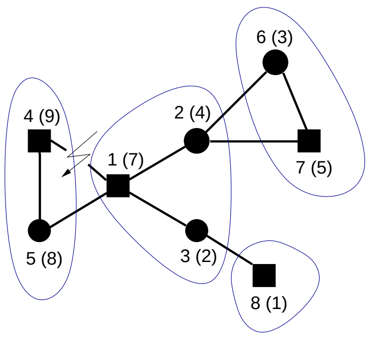

Fig. 6. Clustering structure after link failure

B.2 General Analysis and Recall of Design Criteria

A link failure between different clusters or between any two ordinary nodes does not cause any messages in the DMAC algorithm. Since these events may happen very frequently, this design is advantageous, according to Design Goal 1. DMAC does not allow two clusterheads to be neighbors, so we can also exclude this case from our analysis. Only link failures between an ordinary node and its clusterhead trigger a re–structuring process. In this case, only the ordinary node must actively react to the topology change; the clusterhead just deletes the node from its member list. Two cases can be distinguished (see Table III):

If the ordinary node has a neighboring clusterhead with higher weight (that was located in a different cluster before the link failed), it will join this cluster. One message and one time step is needed.

Otherwise, if the node has no neighboring clusterhead with higher weight, it will become a clusterhead itself. It sends out a CLUSTERHEAD message, which is processed by all d neighbors. The further procedure depends on the role of these neighbors. The restructuring process is the same as if a new node becomes a clusterhead (Section IV-A, complexities in Table II). If we have no information about the neighboring nodes, only lower bounds for the complexities can be stated.

Table III: Message and Time Complexities for "Link Failure" Event

Link fails within cluster between		Complexities	
Node 1	Node 2	M	T
ordinary node	ord. node	0	0
node with neighboring cl.head of higher weight	cluster-head	1	1
ordinary node without neighboring clusterhead of higher weight	cluster-head	$\geq$1+d_{hl} +d_{ol}	$\geq$1

C. New Link

The last scenario to be investigated is a new link between two nodes. Such an event may occur when two nodes move toward each other and reach their transmission range.

C.1 Example Scenario

Let us give an example: A new link between Node 3 ($w_3 = 2$) and Node 7 ($w_7 = 5$) in Fig. 3 is established. Since both nodes are made aware of the presence of a new neighbor, they both run the same procedure to process the new information. Node 7 decides to keep its role as clusterhead, since its own weight is higher than the weight of Node 3 ($w_7 > w_3$). Node 3 recognizes Node 7 as clusterhead with higher weight and, therefore, sends out a JOIN(3,7) message. Node 7 receives and processes this message by assimilating Node 3 to its current Cluster(7), which now consists of the Nodes 7, 2, 3, and 6. On receiving the JOIN(3,7) message, Node 1 does not react at all, but Node 8 decides to become a clusterhead itself and sends out a CLUSTERHEAD(8) message in the second time step. Node 3 will process this CLUSTERHEAD(8) message, but will not react because of having a current clusterhead with a higher weight than that of Node 8. Thus, to summarize, a valid cluster structure is reached (see Fig. 7) after the sending of $M = 2$ messages in $T = 2$ time steps.

C.2 General Analysis and Recall of Design Criteria

Let us now make general considerations about new links. Four different combinations of the roles of the two involved nodes are possible (see Table IV).

In case a new link is established between two ordinary nodes, both of them have their clusterhead and thus ignore the new link, since the cluster structure is still valid.

If an ordinary node becomes neighbor of a clusterhead with a weight lower than the weight of its current clusterhead, the ordinary node decides to stay in its current cluster. Again, no messages are sent out.

Table IV: Message and Time Complexities for "New Link" Event

New link between		Description	Complexities	
Node 1 (w_1)	Node 2 (w_2)		M	T
ordinary node	ordinary node	Nothing happens	0	0
ord. node with clusterhead (w_{1h})	clusterhead ($w_2 < w_{1h}$)	Node 1 stays in cluster	0	0
ord. node with clusterhead (w_{1h})	clusterhead ($w_2 > w_{1h}$)	Node 1 joins Cluster(2)	1	1
clusterhead	clusterhead ($w_2 > w_1$)	Node 1 joins Cluster(2)	$\geq$1+d_{om}	$\geq$1

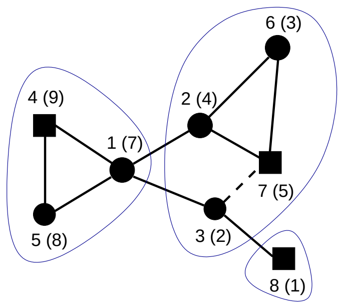

Fig. 7. Clustering structure after new link

Another case occurs when the newly neighboring clusterhead of an ordinary node has a higher weight than its current clusterhead. The ordinary node sends out a JOIN message to become a member of the new clusterhead. The latter assimilates its new member, its former clusterhead deletes the node from its cluster, and we obtain $M = 1$ and $T = 1$.

The last and most complicated case is represented by two clusterheads that become neighbors. This constellation is forbidden and must be resolved. The node with the lower weight, referred to as Node 1 in Table IV (line 4), joins the cluster of the other node. The JOIN message is also received by all, say d_{om}, members of the former clusterhead. They notice that they do not anymore belong to a valid cluster at this time. This enforces at least one further message in each member node, namely JOIN or CLUSTERHEAD. The $d_{oh} = d_1 - d_{om}$ direct neighbors of Node 1 that are assigned to other clusters do not react. The total number of messages and time steps needed to reach a valid clustering structure depend on the number and roles of the former clusterhead's members. A similar situation occurred in Section IV-A by adding a new node which has clusterhead neighbors with lower weight. Also in this case, only lower bounds can be derived and chain reactions are possible.

Note that Design Goal 1 is also fulfilled for a new link event. Very frequent events, such as a new link between two ordinary nodes, produce very low traffic and are terminated within at most one time step.

V. Conclusions and Further Work

The theoretical investigations of this paper gave an insight into how clustering algorithms react to topology changes. We analyzed the message complexity M and time complexity T of the DMAC algorithm in a wireless ad hoc network. The complexities for a "new node" event depend on the role and number of neighbors of the new node. For some topologies, we were able to derive exact values for T and M; for the general case, we could state lower bounds. We made the interesting observation that a single event may cause a restructuring "chain reaction" along a path that fulfills certain characteristics. These characteristics have been investigated. The analysis of a "link failure" and "new link" event yields similar results. Another original contribution of this paper are three design goals, which can be used for the development of new algorithms as well as for the evaluation of existing clustering schemes.

In further research, it will be interesting to carry out investigations on how to avoid the chain reaction in the algorithm. Moreover, a simulation–based analysis of different strategies for weight allocation to improve cluster stability (and thus also message and time complexity) should be performed. First results on the latter topic have been published, e.g., in [2].

References

[1] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks," *Computer Networks*, vol. 1, pp. 155–174, Jan. 1977.
[2] C. Bettstetter and R. Krausser, "Scenario-based stability analysis of the distributed mobility-adaptive clustering (DMAC) algorithm," in *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, (Long Beach, CA, USA), Oct. 2001.
[3] S. Basagni, "Distributed clustering for ad hoc networks," in *Proc. International Symp. on Parallel Architectures, Algorithms, and Networks (ISPAN)*, (Perth/Fremantle, Australia), June 1999.
[4] A. B. McDonald and T. F. Znati, "A mobility–based framework for adaptive clustering in wireless ad hoc networks," *IEEE J. on Sel. Areas in Communications*, vol. 17, Aug. 1999.
[5] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE J. on Sel. Areas in Communications*, vol. 15, Sept. 1997.
[6] S. Basagni, "Distributed and mobility–adaptive clustering for multimedia support in multi–hop wireless networks," in *Proc. IEEE Vehicular Techn. Conf. (VTC)*, (Amsterdam, Netherlands), Sept. 1999.
[7] C. Bettstetter and J. Xi, "Mobility modeling and analysis of adaptive clustering algorithms in ad hoc networks," in *Proc. European Personal Mobile Communications Conf. (EPMCC)*, (Vienna, Austria), Feb. 2001.