Time and Message Complexities of the Generalized Distributed Mobility-Adaptive Clustering (GDMAC) Algorithm in Wireless Multihop Networks

Christian Bettstetter and Bastian Friedrich
Technische Universität München (TUM), Institute of Communication Networks, Munich, Germany
Email: bettstetter@ei.tum.de, Web: http://www.lkn.ei.tum.de

***Abstract*— Distributed mobility–adaptive clustering algorithms are used in multihop ad hoc networks to separate the nodes into logical groups and build up a hierarchical network architecture. This paper studies the convergence time and message complexity of Basagni's generalized DMAC clustering algorithm. Our results show how many time steps and signaling messages are typically needed after a single topology change to re–achieve a stable and valid cluster structure. Furthermore, we discuss chain reactions that can occur along a path if certain conditions are fulfilled. Finally, we regard a mobile scenario in order to analyze (a) the number of signaling messages per node and time step and (b) the percentage of time steps in which the cluster structure is invalid. Our results give a qualitative insight on the behavior of clustering in ad hoc networks. In particular, they show that tuning the density of clusterheads and employing a hysteresis parameter for cluster changes can significantly improve the performance.**

***Index Terms*— Ad hoc networking, clustering, leader election, distributed algorithms, sensor networks.**

I. Introduction

AN ESSENTIAL requirement to achieve scalability in large networks is the logical separation of network nodes into groups, so–called "clusters" or "subnets." Such clustering enables us to set up hierarchies which can be used for address assignment, routing, and resource control. Also in large multihop ad hoc networks, e.g., wireless sensor networks, it is a desirable feature to obtain a clustered network. It is therefore not surprising that several distributed clustering algorithms have been proposed in this area during the last few years [1, 2].

One promising and yet simple algorithm, called *Distributed Mobility–Adaptive Clustering (DMAC)*, was presented by Basagni in [3] and analyzed by Bettstetter *et al.* in [2, 4]. An extended version of this algorithm, called *Generalized DMAC (GDMAC)*, was proposed in [5, 6]. The paper at hand analyzes this enhanced algorithm with respect to its convergence time and message complexity. These two values are fundamental criteria in the design and performance evaluation of distributed algorithms. Moreover, this paper makes a contribution to the fundamental understanding of how distributed clustering algorithms behave in a wireless multihop scenario.

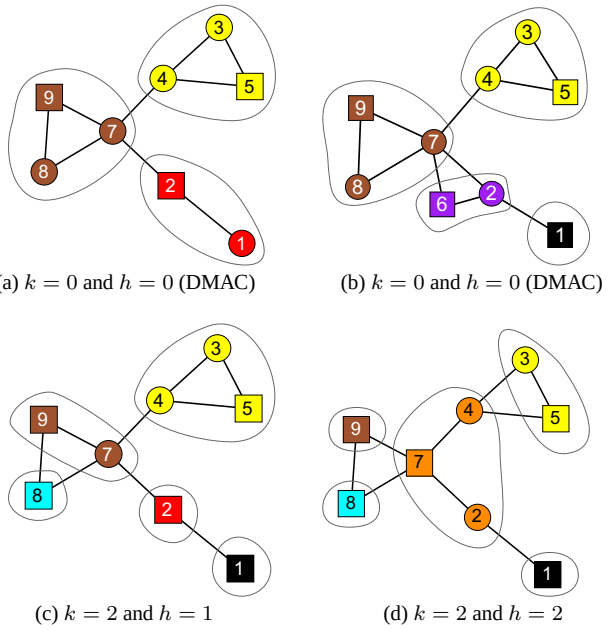
In Section II, we review the basic operation and message types of the GDMAC algorithm. Sections III and IV present our simulation results on convergence time and message complexity for a single topology change, namely a new node event. We analyze (a) how long it takes on average and (b) how many messages must typically be sent after such an event until a validly clustered network structure is re–achieved. Section V addresses the possibility of chain reactions — an undesirable characteristic of the GDMAC algorithm. Section VI considers a scenario with mobile nodes. Here, we investigate (a) the message complexity and (b) the percentage of time with invalid cluster structures. Finally, Section VII concludes this paper.

II. Network Model and Clustering

Given is a wireless multihop network with n uniformly randomly distributed nodes, each with radio transmission range r_0, on a square system area of size $a \times a$. Two nodes establish a wireless link if they are within range of each other. Each node runs the GDMAC clustering algorithm and exchanges signaling messages to achieve a clustered network structure. Based on a node weight w and the weights of neighboring nodes, a node either becomes a *clusterhead* or *ordinary node*. The larger the weight of a node, the better it is suited to be a clusterhead.

The GDMAC clustering rules are designed in such a manner that the network converges within a finite time to a valid cluster structure. Such a valid structure is defined by the following three conditions [5]: (a) every ordinary node joins exactly one clusterhead; (b) for every node affiliated with a clusterhead CH (and for every clusterhead CH itself), there is no other neighboring clusterhead CH' with weight $w(CH') > w(CH) + h$, where $h \in \mathbb{N}_0$; (c) a clusterhead can have up to k neighboring clusterheads. Figure 1 shows four examples of valid GDMAC cluster structures. The numbers within the nodes indicate their weights w. Clusterheads are shown as squares and ordinary nodes as circles.

In the following, we explain how the nodes must act such that the above three conditions are fulfilled. When a node appears in the network, it executes an initialization process to determine its *role*, i.e., whether it should become an ordinary node or clusterhead. This decision is solely based on the local view of the node. It decides to join a cluster — thus becomes an ordinary node — if there is already a neighboring clusterhead with a higher weight; otherwise it decides to become clusterhead itself. After making this decision, the node informs all its neighbors of its role. It sends out a JOIN message if it joins a cluster or a CLUSTERHEAD message if it becomes clusterhead. The algorithm is message driven and executed at each node. In order to react properly and consistently, each node has to know its own weight and role as well as the weight and role of each of its current neighbors. If the situation occurs that



(a) $k = 0$ and $h = 0$ (DMAC) (b) $k = 0$ and $h = 0$ (DMAC)

(c) $k = 2$ and $h = 1$ (d) $k = 2$ and $h = 2$

Fig. 1. Examples for GDMAC clustering.

a clusterhead has more than k neighboring clusterheads, it informs its neighbors about this invalid cluster structure by sending out a RESIGN(w) message. The value of the parameter w in this message indicates that all neighboring clusterheads CH' with weight $w(CH') \leq w$ must resign in order to re–achieve a valid structure.

As the nodes move around, they must decide which cluster they currently belong to 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 adapts its status and cluster membership accordingly. Whenever a link failure happens between two nodes, both nodes check if their own role is clusterhead and if the other node belongs to its cluster. If this is the case, the clusterhead removes the other node from its cluster. In case 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. An ordinary node affiliated with a clusterhead CH remains member of this clusterhead as long as there is no neighboring clusterhead CH' with $w(CH') > w(CH) + h$. The same rule holds for clusterheads: They keep their role as long as the above condition is fulfilled, where $w(CH)$ now denotes the weight of the node itself. Using the described procedures, any multihop network can be clustered such that the above three conditions for cluster validity are fulfilled [3].

Let us briefly discuss the clustering parameters h and k. The parameter h gives the dynamic clustering a sort of a hysteresis effect. Upon initialization, a node joins the clusterhead CH with the highest weight in its neighborhood. While the topology changes, the node switches to other clusterheads CH' appearing in its radio range. If $h = 0$ it must always switch to the clusterhead that has a higher weight than its current clusterhead. If $h > 0$, however, it can keep its old clusterhead even if there appears a neighboring clusterhead CH' with higher weight, as long as $w(CH') \leq w(CH) + h$. Thus, a higher h typically results in fewer cluster changes.

The parameter k controls the spatial density of clusterheads. If $k = 0$ any two clusterheads must be at least two hops away from each other. The RESIGN messages can be skipped in this case. In the extreme case $k \to n$ the network may consist of clusterheads only. The GDMAC algorithm with $h = 0$ (no hysteresis) and/or $k = 0$ (two clusterheads must not be neighbors) corresponds to the DMAC algorithm [3].

The reader is referred to [3, 5, 6] for further details about the GDMAC algorithm. In the paper at hand, the node weights w are taken from a uniform random distribution between 1 and 80 000. We operate the clustering algorithm in a synchronous manner. In each time step a node can process all signaling messages received in the previous time step.

III. CONVERGENCE TIME

Let us first analyze the convergence time of the GDMAC algorithm as a function of the network and clustering parameters. To do so, we regard a validly clustered network and investigate the consequence of a new node that is randomly placed on the system area. This single new node event may trigger a re–clustering of the network. If the new node has a high weight, it may become clusterhead and its neighboring nodes may join it. There also exist situations in which these neighbors give up their clusterhead role. This resignation triggers in turn reactions of nodes affiliated with the former clusterhead. An example is given in Figures 1a and b. Fig. 1a represents a valid cluster structure with $n = 8$ nodes. A new node with weight $w = 6$ is added to the network. This node decides to become clusterhead since it has no neighboring clusterhead with higher weight. It sends out a CLUSTERHEAD message which is received by both neighbors. The node with $w = 7$ ignores the message, since it has a clusterhead with higher weight. The node with $w = 2$, however, gives up its clusterhead role and joins the new node. Therefore the node with $w = 1$ has to become clusterhead itself. Fig. 1b shows the re–clustered structure.

We now study the convergence time T_{valid}^{new} that is needed to re–achieve a valid cluster structure after such a new node event. We define this convergence time as the number of time steps from the new node event until all clustering rules are fulfilled again; no node changes its role or joins a different cluster afterwards. Our goal is to find out how the network parameters n and r_0 and the clustering parameter k influence T_{valid}^{new}. To do so, we generate a random network topology with n nodes, let the clustering algorithm run until a valid structure is achieved, and then add a new node at a random position and measure the number of time steps until a valid structure is re–achieved. If the new node joins an existing cluster, only one time step is needed, i.e., the minimum value of T_{valid}^{new} is always 1. The same experiment is repeated for 30 000 random topologies, and the resulting values are averaged to give us the expected convergence time $E[T_{valid}^{new}]$.

Figure 2 shows our results on the impact of the number of nodes n on $E[T_{valid}^{new}]$. Let us first look at the curve for transmission range $r_0/a = 0.1$. Using a network with $n = 5$ nodes and adding one node, we need on average $T_{valid}^{new} \approx 1.05$ time

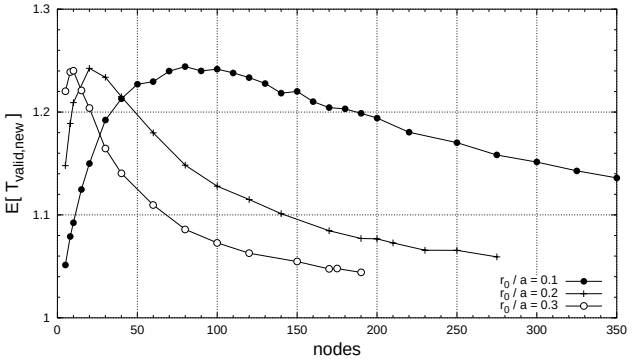


Fig. 2. Average conv. time $E[T_{valid}^{new}]$; new node event; $h = 10\,000$, $k = 2$.

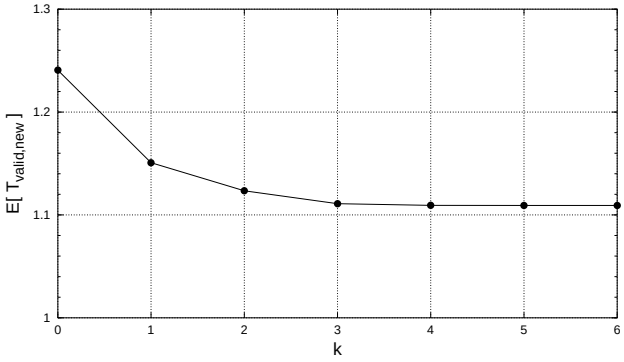


Fig. 3. Average convergence time $E[T_{valid}^{new}]$ over GDMAC parameter k, new node event; $r_0/a = 0.29$, $n = 50$, gives $P(\text{con}) = 95\%$; $h = 10\,000$.

steps to re–achieve a valid structure. If we increase n, the average convergence time increases until a maximum of about $E[T_{valid}^{new}] \approx 1.24$ is achieved for $n = 70 \ldots 100$. Increasing n further, results again in a shorter convergence time. We conjecture that this behavior is a consequence of two opponent effects. On the one hand, the convergence time is long if a new node becomes a clusterhead. On the other hand, it is also long if many nodes are affected by the new clusterhead, i.e., if the new node has many neighbors. For increasing node density n/a^2, the probability of a new node to become clusterhead is decreasing, whereas the number of affected nodes is increasing. An additional result of our simulations is that the worst case convergence time was $T_{valid}^{new} = 7$.

What happens if we increase the radio transmission range? The curves for $r_0/a = 0.2$ and 0.3 in Figure 2 show the following behavior: the higher the radio range r_0/a, the sooner the maximum of the average convergence time is achieved and the shorter the convergence time is for a large number of nodes. This is because the higher the range of the new node the smaller is its probability to become clusterhead.

It seems that the convergence time for new node events is somehow related to the connectivity of the network. Thus, in a second experiment, we choose the network parameters n and r_0 such that the resulting random multihop network is connected with a probability of $P(\text{con}) = 95\,\%$. These $(r_0/a, n)$ pairs have been taken from [7]. For example the pairs $(0.44, 20)$, $(0.32, 40)$, $(0.23, 80)$, $(0.207, 100)$, $(0.185, 129)$ (0.17,151), $(0.120, 313)$, and $(0.1, 455)$ achieve a connected network topology with a probability of $95\,\%$. The GDMAC parameters remain unchanged. The result of these simulations is that $E[T_{valid}^{new}]$ remains at a *constant value* of 1.12.

In a third experiment, we study the impact of the clustering parameter k on T_{valid}^{new} for a network which is connected with probability $P(\text{con}) = 95\,\%$ (see Figure 3). Compared to the original DMAC algorithm ($k = 0$, clusterheads cannot be neighbors), the introduction of the parameter k reduces the convergence time. For $k = 3$ we obtain $E[T_{valid}^{new}] = 1.11$, as opposed to 1.24 for $k = 0$. Values $k > 3$, however, do not significantly reduce the convergence time.

Let us now investigate a second, slightly different definition of convergence time, namely T_{stable}^{new}. We define it as the number of time steps needed after a new node event until a so–called *stable* cluster structure is obtained: each node already determined its role at this time step, but ordinary nodes may join a different clusterhead after this time step in order to achieve a valid structure. Clearly, $T_{stable}^{new} \leq T_{valid}^{new}$ for a given scenario. If the new node joins an existing cluster, we obtain $T_{stable}^{new} = 0$. Figure 4 shows the expected value $E[T_{stable}^{new}]$ over n for fixed r_0/a. In this case, about one time step is needed on average for convergence of $5 + 1$ nodes. As the node density increases, the average number of time steps decreases because the probability of the new node to become clusterhead is also decreasing. In fact, it seems that the convergence time falls off exponentially.

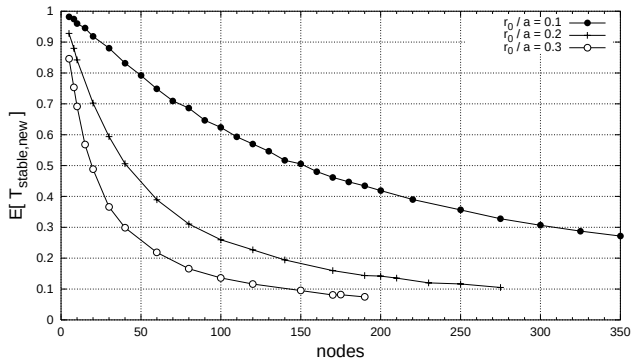
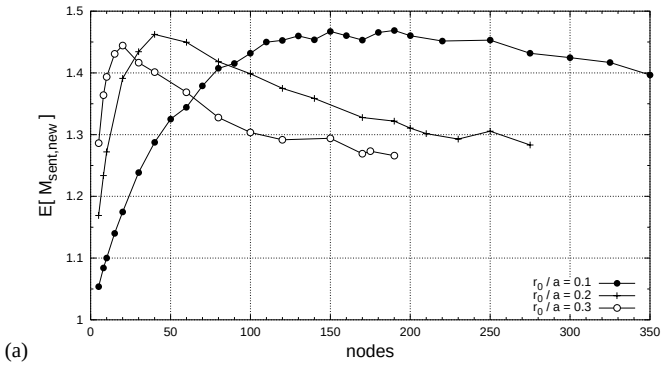


Fig. 4. Average conv. time $E[T_{stable}^{new}]$, new node event; $h = 10\,000$, $k = 2$.

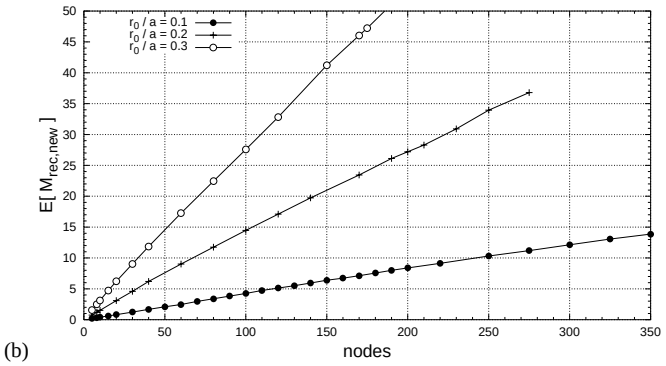
IV. Message Complexity

After our analysis of the number of time steps required to re–establish a stable or valid cluster structure, we now investigate the *number of messages* that is exchanged between nodes after a new node event. We define two message complexities: the number of sent messages M_{sent} and the number of received messages M_{rec}. Again, each point of the simulation results is based on the outcome of $30\,000$ random topologies.

First, we consider the number of sent messages. We note that at least one message is sent out by a new node, i.e., $M_{sent} \geq 1$ for any random scenario. Depending on the message type and the role of neighboring nodes, additional messages from other nodes may or may not be triggered. Figure 5a shows the expected value $E[M_{sent}^{new}]$ over n for fixed range. Starting at low n, the average number of messages increases until a maximum of about $E[M_{sent}^{new}] \approx 1.5$ messages is achieved. Increasing the node density further, results again in a lower message complexity. This behavior is due to the same two opponent effects as



(a)



(b)

Fig. 5. Expected number of (a) sent messages $E[M_{sent}^{new}]$ and (b) received messages $E[M_{rec}^{new}]$ for new node event; $h = 10\,000$, $k = 2$.

for the time complexity $E[T_{valid}^{new}]$, although the maxima are obtained for different n. Studying the impact of the transmission range, we can say that higher transmission ranges yield a higher complexity for low n and lower complexity for high n. Thus, in another experiment, we choose (r_0, n)–pairs such that the connectivity probability is $P(\mathrm{con}) = 95\,\%$. Keeping the clustering parameters unchanged yields a constant message complexity of $E[M_{sent}^{new}] = 1.4$. The maximum number of sent messages in the latter experiment was $M_{sent}^{new} = 38$.

We now consider the expected number of received messages $E[M_{rec}^{new}]$. As mentioned above, a new node sends out at least one message, which is received by a certain number of neighboring nodes. A CLUSTERHEAD message is received by all neighbors and a JOIN message by a single clusterhead. Clearly, a higher range r_0/a and/or a higher node density n/a^2 increases the number of receiving nodes. It now depends on various parameters, how many further messages are triggered and received by other nodes in the following time steps. Our simulation–based investigation of $E[M_{rec}^{new}]$ yields an interesting result: as shown in Figure 5b, $E[M_{rec}^{new}]$ increases approximately linearly with the number of nodes n.

To conclude this section, Figure 6 shows the impact of the GDMAC parameter h on $E[M_{sent}^{new}]$. Recall that $h > 0$ introduces a hysteresis for cluster changes. A node may keep its current clusterhead CH as long as there is no other clusterhead in its range with weight $w(CH') > w(CH) + h$. Using no hysteresis ($h = 0$) yields a message complexity of $E[M_{sent}^{new}] \approx 2.0$. Choosing h in the order of $h_{max}/2 = 40\,000$ makes the cluster structure much more stable. Thus, the message complexity can be significantly reduced: about 1.2 messages are needed in this case to re–achieve a valid clustering. In other words, the signaling overhead has been reduced by $40\,\%$. Increasing h further does not result in an additional gain.

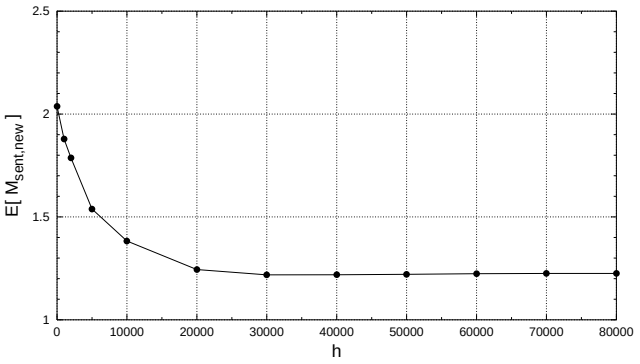


Fig. 6. $E[M_{sent}^{new}]$ over GDMAC parameter h, new node event; $r_0/a = 0.29$, $n = 50$, $k = 2$.

V. CHAIN REACTIONS

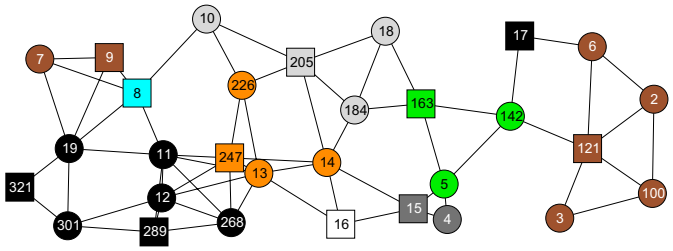
As we have shown in [4], the usual (non–generalized) DMAC algorithm [3] has the undesirable property that chain reactions might occur in certain scenarios. In the worst case, a single change in the topology forces many nodes along a tree throughout the entire network to change their roles. The question arises: does the generalized version of the DMAC algorithm also have this property? Unfortunately, the answer is "yes." As shown in Figure 7, a new node may trigger a re–clustering chain reaction, for instance, if the following conditions are fulfilled: (1) clusterheads and ordinary nodes appear alternately in the path, (2) the successor of a node in the path has lower weight than the node itself and fulfills the GDMAC-h-condition, and (3) no ordinary node has a clusterhead with higher weight than its own predecessor in the path. These conditions, however, are more strict than in the conventional DMAC algorithm. Consequently, a chain reaction occurs with a lower probability. In addition, a GDMAC chain reaction is limited in length to $\min\left(\left\lfloor \frac{w_{max}}{h+1} \right\rfloor, n\right)$ links, whereas it can be up to n using the conventional DMAC.

VI. MOBILE NODES

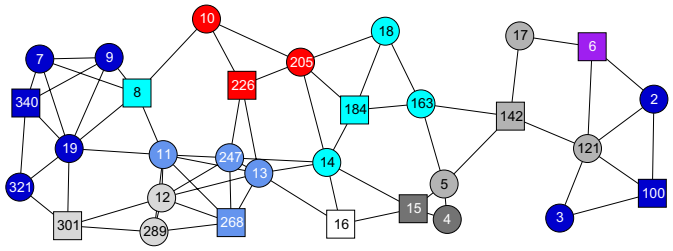
Finally, we investigate GDMAC clustering in a scenario with mobile nodes. We analyze (a) the message complexity and (b) the percentage of time during which a given node has an invalid cluster structure. The used mobility model is explained in [2], and a bounce–back behavior at the borders is employed.

Figure 8 shows the averaged number of sent messages $E[M_{sent}^{mobile}]$ per time step and per node. Both a fixed range $r_0/a = 0.1$ and range values for $P(\mathrm{con}) = 95\,\%$ have been simulated. Starting at a low number of nodes, the message complexity first increases dramatically. As we increase n further, the curve levels off and a certain saturation seems to be achieved. The same qualitative behavior can be observed for the percentage of invalid time steps, averaged over all nodes (see Fig. 9).

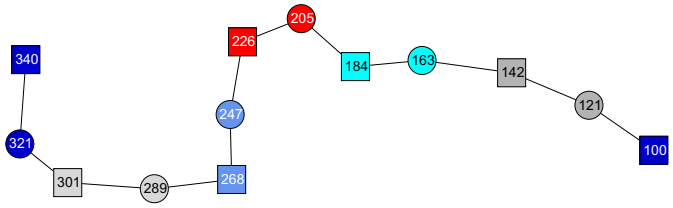
Finally, Figure 10 shows that increasing the cluster parameter k can significantly reduce the number of invalid time steps.



(a) Cluster structure before chain reaction



(b) Cluster structure after chain reaction



(c) Path of chain reaction

Fig. 7. Chain reaction triggered by new node with weight $w = 340$; $h = 10$, $k = 1$; squares are clusterheads.

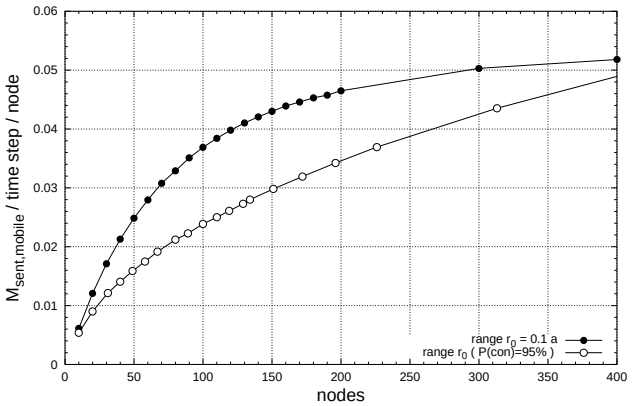


Fig. 8. Average value of sent messages $E[M_{sent}^{mobile}]$ per time step per mobile node. GDMAC parameters $h = 10000$, $k = 2$. Mobility $v_{max} = 0.0075a$ per time step, movement each second step.

Using clustering with $k = 0$ in an almost surely connected network results in an invalid structure for a node during $3\,\%$ of the time. Allowing that each clusterhead can have $k = 2$ neighboring clusterheads decreases this value down to about $1.5\,\%$ — a reduction of $50\,\%$. Choosing a higher k does not significantly improve the performance.

VII. Conclusions

This paper investigated the GDMAC clustering algorithm for wireless multihop networks. Our results can be used to give a qualitative estimate of the convergence time and signaling load of this algorithm. To summarize, the average values of both

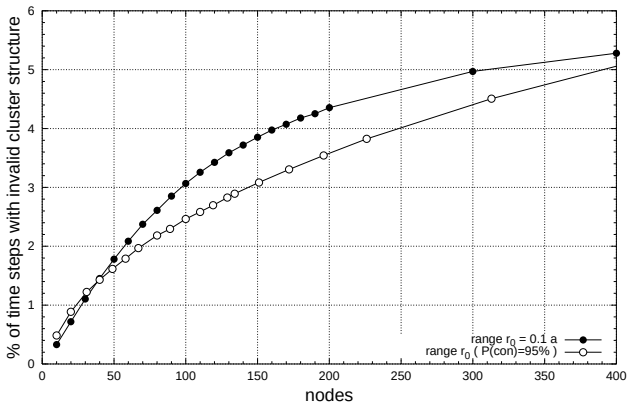


Fig. 9. Percentage of time steps with invalid cluster structure for a node. Same parameters as in Fig. 8.

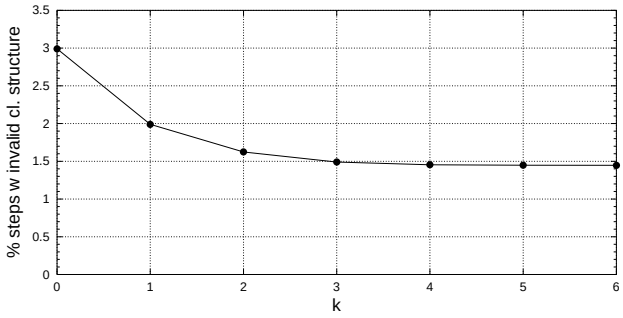


Fig. 10. Percentage of time steps with invalid cluster structure for a node. $r_0/a = 0.29$, $n = 50$, gives $P(\mathrm{con}) = 95\%$; $h = 10\,000$.

parameters seem to scale well for increasing n. Chain reactions are theoretically possible but with a very low probability. The GDMAC parameters k and h can be tuned in such a way that the performance is improved.

Acknowledgement

This research was funded by the Deutsche Forschungsgemeinschaft (DFG) in the program 'Adaptability in heterogeneous communication networks with wireless access.'

References

[1] C. E. Perkins, ed., *Ad Hoc Networking*. Addison Wesley, 2001.

[2] C. Bettstetter and R. Krausser, "Scenario-based stability analysis of the distributed mobility-adaptive clustering (DMAC) algorithm," in *Proc. ACM Symp. on Mobile Ad Hoc Netw. and Comp. (MobiHoc)*, (Long Beach, USA), pp. 232–241, October 2001.

[3] S. Basagni, "Distributed clustering for ad hoc networks," in *Proc. Intern. Symp. on Parallel Architectures, Algorithms and Networks (ISPAN)*, (Perth/Fremantle, Australia), June 1999.

[4] C. Bettstetter and S. König, "On the message and time complexity of a distributed mobility-adaptive clustering in wireless ad hoc networks," in *Proc. European Wireless (EW)*, (Florence, Italy), February 2002.

[5] 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.

[6] S. Basagni, "Distributed and mobility-adaptive clustering for ad hoc networks," Tech. Rep. UTD/EE-02-98, The University of Texas at Dallas, Dallas, USA, July 1998.

[7] C. Bettstetter, "On the connectivity of wireless multihop networks with homogeneous and inhomogeneous range assignment," in *Proc. IEEE Vehicular Techn. Conf. (VTC)*, (Vancouver, BC, Canada), Sept. 2002.