Demand-Driven Clustering in MANETs

Curt Cramer, Oliver Stanze, Kilian Weniger, and Martina Zitterbart
Institute of Telematics
University of Karlsruhe (TH)
Germany
eMail: {cramer|stanze|weniger|zit}@tm.uka.de

***Abstract*— Many clustering protocols for mobile ad hoc networks (MANETs) have been proposed in the literature. With only one exception so far [1], all these protocols are proactive, thus wasting bandwidth when their function is not currently needed. To reduce the signalling traffic load, reactive clustering may be employed. We have developed a clustering protocol named *"On-Demand Group Mobility-Based Clustering"* (ODGMBC) which is reactive. Its goal is to build clusters as a basis for address autoconfiguration and hierarchical routing. The design process especially addresses the notion of group mobility in a MANET. As a result, ODGMBC maps varying physical node groups onto logical clusters.**

In this paper, ODGMBC is described. It was implemented for the ad hoc network simulator GloMoSim [2] and evaluated using several performance indicators. Simulation results are promising and show that ODGMBC leads to stable clusters. This stability is advantageous for autoconfiguration and routing mechansims to be employed in conjunction with the clustering algorithm.

***Index Terms*— clustering, multi-hop, reactive, MANET, group mobility**

I. Introduction

Routing was and continues to be one of the prevailing research topics in the field of mobile ad hoc networks (MANETs). In MANETs as well as in static networks, the introduction of hierarchies may aid in achieving routing efficiency. However, due to the dynamic nature of MANETs, one cannot statically assign nodes to subnetworks. Subsequent node movements are very likely to mix up the assignment and therefore diminish the benefits of hierarchical routing.

A straightforward way to dynamically assign nodes to subnetworks is clustering [3]. Here, nodes are grouped together to form a logical control structure based on their physical proximity and their movement history. Each group of nodes, the so-called clusters, then forms a subnetwork. E.g. a designated node in the cluster disseminates an IP address prefix among its fellow nodes. These designated nodes are called *leaders* or sometimes *cluster heads*.

Subnetworks are referred to as clusters which have been assigned an IP address prefix common among the participating nodes. The subnetworks form the basis for hierarchical routing protocols. The actual process of auto-configuration and hierarchical routing is beyond the scope of this paper. It is focused on the issue of building and maintaining clusters. Therefore, the control structures that are dealt with in the paper are called *clusters*. We specifically question whether the existence of *physical* groups of nodes, i.e. nodes moving together in physical proximity over longer time spans, can be leveraged in order to improve the stability of clusters, which is an important measure of a clustering protocol's quality, as will be explained later. Nodes moving individually will not be part of clusters, as they are only crossing the path of groups during small time intervals.

The diameter of a group is likely to be more than one hop. Imagine for example a group of tourists equipped with some short-range radio equipment. Therefore, the clustering protocol should be multi-hop. However, the majority of clustering protocols proposed in the literature thus far only forms clusters of one-hop radius (see, e.g., [1], [4], [5], [6], [7], and [3]).

In the same way as routing protocols can be classified as being either reactive or proactive, clustering protocols may be as well. A proactive clustering protocol continuously tries to build and maintain clusters in a MANET decoupled from the actual necessity to do so (i.e. traffic load). As a consequence, precious bandwidth is wasted when no traffic flows and hence no routing hierarchy is actually needed. To our best knowledge, only one reactive clustering protocol has been proposed so far, namely Passive Clustering (PC) by Kwon and Gerla [1]. Contrary to ODGMBC, PC generates one-hop clusters. The clusters are used to increase the efficiency of flooding, not for supporting hierarchical routing. Only a subset of all nodes is allowed to forward flooded packets. Hence, MAC layer broadcasting conflicts are reduced [8].

This paper discusses design and performance issues of the protocol named "On-Demand Group Mobility-Based Clustering" (ODGMBC). It has the following features:

- It is reactive,
- is multi-hop and
- addresses the special situation of a MANET with group mobility.

The protocol is to be employed together with a reactive routing protocol. It also works with proactive routing, yet the routing protocol's signalling traffic in this case leads to continuous execution of the clustering process.

The next section gives a formal definition of the clustering problem under consideration and ODGMBC is presented in some detail. For comparison reasons, its proactive precursor GMBC, also developed in our lab, is described briefly.

Section III defines the scenarios used to experimentally evaluate ODGMBC by means of simulation. Several performance indicators are defined and employed to make a head-to-head comparison of GMBC and ODGMBC. The work presented in this paper is being compared with previously published clustering solutions in section IV. The last section concludes the findings of this paper and points out some directions to be investigated in subsequent research.

II. Group Mobility-Based Clustering

A. Problem Definition

As stated in the introduction, the objective is to develop a multi-hop clustering protocol. We further required the protocol to generate non-overlapping clusters, i.e. each node is member of at most one cluster at the same time. Otherwise, the routing and addressing mechanisms would have to deal with the situation of multi-homed nodes. This is not the primary goal, as it leads to increased complexity.

The clusters should be the basis for hierarchical routing. By using an auto-configuration mechanism, address prefixes are assigned to the clusters. The address prefix for a cluster is maintained by the cluster's leader. Each node in a cluster has to know its address prefix. From this requirement, it follows that only the cluster members need to know their current leader, but not vice versa. We assume that each cluster can be identified network-wide by some unique value assigned to it. This unique value could for example be the leader's MAC address.

Consequently, a node can be in three different states: It can either be no member of any cluster (*unclustered*), member of exactly one cluster (*clustered*) or the leader of exactly one cluster (*leader*).

As a result of the nodes' movements, there are several possible events in the life cycle of a cluster (see figure 1). Two nodes may meet and form a cluster, one of them declaring itself to be leader (event *create*). Nodes can be added to an existing cluster (event *join*). Or two clusters may be collapsed to one (event *merge*). If a cluster is split in two with one of the remaining parts having no leader, the cluster has to be "repaired" accordingly. Nodes may depart from their clusters (event *leave*), and a cluster disappears when all members leave it.

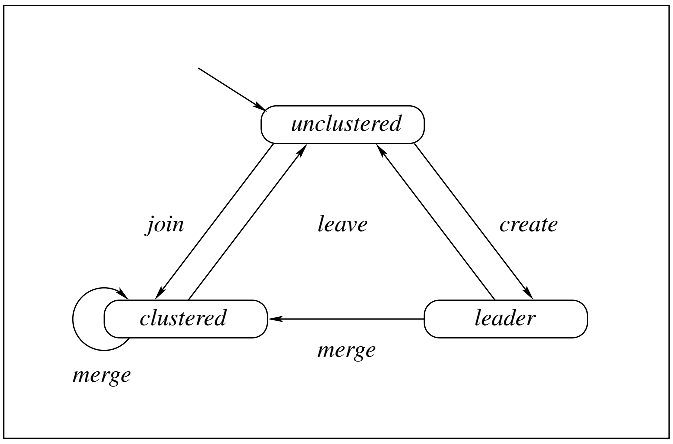


Fig. 1. Finite-state machine for group mobility-based clustering

The last and most significant requirement for ODGMBC is that it has to be reactive. Most of the clustering protocols previously proposed in the literature try to (pro-)actively discover a node's neighbourhood by periodically broadcasting "*hello*"-beacons. By counting the "hello"s received from various nodes over longer timespans, it is possible to draw conclusions about the physical proximity of nodes. ODGMBC aims to demonstrate that passive (reactive) monitoring of the data traffic is sufficient to detect node neighbourhoods.

B. Proactive Clustering – GMBC

First, ODGMBC's precursor is described briefly. This protocol named "Group Mobility-Based Clustering" (GMBC) is quite simple, therefore suitable for an introductory view on clustering with group mobility and lending itself to comparisons of experimental results.

With GMBC, nodes periodically broadcast "hello"-packets. The packets are overheard by nodes within radio range. Every "hello" received from different neighbouring nodes is counted. If the number of "hello"-packets from a specific node exceeds a configurable threshold value (with some allowable packet loss to account for undetected MAC layer collisions; see [8] for a discussion of this problem), that node is regarded as a neighbour. Up to this point, the procedure is quite similar to Toh's Associativity Based Routing (ABR) [9].

A node declares itself to be a cluster leader if it has had neighbours for a certain timespan (another configurable threshold value). The declaration is made by broadcasting a "create"-packet (event *create* in figure 1). All nodes which receive this packet check whether it was sent by one of their neighbours. If this is the case, they assume to be part of a group themselves and hence join the advertised cluster (event *join*). The joining nodes re-broadcast the "create"-packet in order to disseminate the cluster information over multiple hops. Propagation is limited through a hop counter field in the packet. The hop counter (which is unrelated to the TTL field in the IP header) is decremented after each hop, stopping the forwarding when it reaches zero. It is required because GMBC uses application-layer forwarding.

The cluster structure needs to be maintained due to the nodes' mobility. A soft-state approach was adopted. The leaders have to refresh the cluster state periodically. They do so by broadcasting a new "create"-packet. To save bandwidth, this packet replaces the leader's "hello"-packet and is sent with the same frequency.

Clusters merge if two leaders get in reach of each other: The leader with the smaller identifier, i.e. the lexicographically smaller MAC address, continues to be leader, the other one gives up. All nodes in the fading cluster will join the persistent cluster when they first hear its leader's "create"-beacon.

In GMBC, all nodes only know their respective cluster leaders. They also change their clusters whenever they overhear a new (or previously unknown) cluster leader. This has severe implications for the overall cluster stability. In the evaluated scenarios, the average rate of nodes changing their clusters was about five times higher than with ODGMBC. With regard to address auto-configuration and routing, this instability leads to many route failures.

C. Reactive Clustering – ODGMBC

The goal of reactive protocols is to reduce protocol overhead when their function is not currently needed. In our case, this means that the signaling traffic should be kept as minimal as possible when there is no traffic flow in the network. No clusters are needed during these periods because hierarchical routing will not be used when there are no packets to be routed anyway. In this case, there will only be no traffic if a reactive routing protocol, such as AODV, is used. Also note that the routing hierarchy is more beneficial with multiple concurrent end-to-end connections in the network.

The problem of reactive clustering can be split in two parts. First, it has to be detected when there is the need for clustering, i.e. when traffic flows. Second, when it is known which nodes participate in traffic flows, clusters have to be established and maintained. These two parts can be regarded as two completely different protocols with strictly defined interfaces (cf. figure 2). In particular, the protocols described afterwards are

1) the "Neighbourhood Recognition Protocol" (NRP) and
2) the clustering protocol itself.

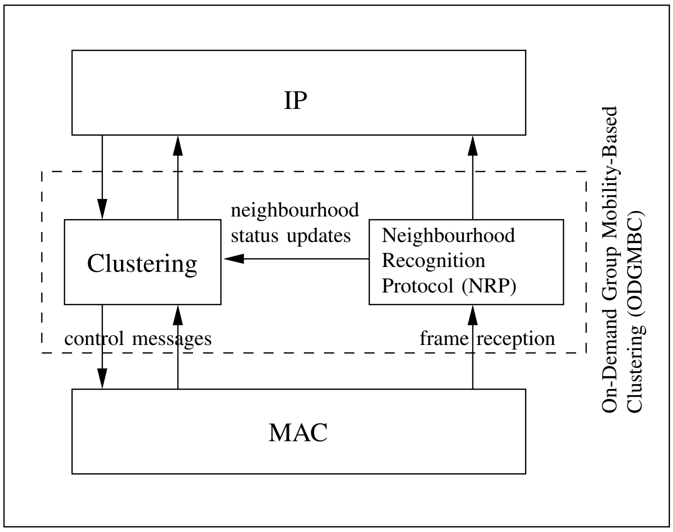

IP
Clustering
neighbourhood status updates
Neighbourhood Recognition Protocol (NRP)
control messages
frame reception
MAC
On-Demand Group Mobility-Based Clustering (ODGMBC)

Fig. 2. Position of NRP and ODGMBC in the protocol stack

1) Neighbourhood Recognition Protocol: We named the protocol which is responsible for the detection of traffic flows "Neighbourhood Recognition Protocol" (NRP). NRP's task is to monitor the traffic flow and inform the actual clustering protocol about the neighbourhood of a node. It maintains a table which maps a MAC address to the number of packets received from the corresponding node during a configurable time period. Each time a frame is received at the MAC layer, NRP looks up the last hop's MAC address in the table. If this address is not found in the table, a new entry is created and a timer is started. Otherwise, the frame counter associated with the MAC address is incremented and the timer keeps running unmodified.

When a timer expires, the frame counter of the corresponding node is compared against a predefined constant threshold. If the counter exceeds the threshold, the node is regarded as a neighbour. Consequently, the clustering protocol is notified of the neighbour's presence. Additionally, the timer is then restarted for this node. The clustering protocol is also informed in case the frame count drops below the threshold during the time period. In this case, the node loses its status as neighbour and the timer is not restarted. Thus, the NRP and the clustering protocol can keep track of an ever changing neighbourhood.

To sum up, the NRP effectively monitors traffic flows by counting the MAC frames received from different neighbouring nodes. Time is divided into equal-sized intervals after the first reception of a MAC frame from a previously unknown node. As a result, the clustering protocol is being informed about the availability of the nodes asynchronously and independently by the NRP.

Note that no packet is sent over the network by NRP. It is purely passive and only communicates with the clustering protocol entity of the monitoring node.

2) Clustering Protocol: By employing the NRP, monitoring nodes know which neighbours are present and suitable for clustering. However, as the neighbourhood recognition only works in one direction, the detected neighbours have to be informed of their detection. Informing the neighbour also helps detecting undirectional physical links. The notification was implemented in two ways: First, if the monitoring node has to send an IP packet to the detected neighbour anyway, a special IP option is placed in the packet header. Second, the detection of neighbours is announced by broadcasting special "*NEIGHBOUR-INFORM*"-packets. As it is very likely that many neighbours are detected in rapid succession, the MAC addresses are held in a queue which is flushed after a (short) predefined time interval. Such a situation is likely to occur when a reactive routing protocol is used. In the initial route discovery phase, a node may detect the presence of multiple neighbours. By using a queue and bundling all the detected nodes' MAC addresses into one broadcast message, bandwidth can be conserved.

Every node keeps record of detected nodes and nodes which detected itself in two separate lists. The neighbourhood graph defined by the NRP and the "*NEIGHBOUR-INFORM*"-messages forms the basic structure for our clustering algorithm. Only the nodes which are contained in this graph participate in the clustering process. Therefore, if all traffic measurements fall below NRP's threshold value, no neighbours are detected and no clusters are built. This exactly was the primary goal.

We adopted an approach similiar to "Random Competition based Clustering" (RCC) [10], which is run by the nodes contained in the neighbourhood graph. Whenever a node detects its first neighbour, it waits for a random time interval (drawn from a uniform distribution). If no other node in its vicinity has declared itself to be leader within the time interval, it does so itself. Note that only nodes which have recognized other nodes as neighbours themselves are allowed to make a leader declaration. This is done by flooding a *LEADER-BEACON*-message. The *LEADER-BEACON* is forwarded only by cluster members. Its propagation is limited by a hop counter in the packet header. Let the maximum propagation distance be denoted by k. Then k-hop clusters are formed. We will refer to k as the maximum cluster radius in the following text.

With RCC, conflicts may arise during the leader election phase. Ties in leader competition are broken as follows: If two nodes within mutual distance of less than or equal to k hops declared themselves to be leaders simultaneously, the node with the lexicographically smaller MAC address has to give up and become cluster member. By the same method, cluster mergers are performed: If two groups cross their ways, they will eventually be merged into one.

Each node records the *LEADER-BEACON*s it has seen, but sticks to its current leader as long as possible. A node in the *unclustered* state (and having neighbours) enters the cluster of the leader it hears first.

State is preserved through a soft-state mechanism: The leaders periodically have to re-broadcast their *LEADER-BEACON*s. In case a node has not heard of its current leader for a prescribed time period, it immediately switches its membership to the cluster whose leader has the minimum distance to the node among all known leaders. In case there are no other leaders, but the node still has neighbours, it declares itself leader. Cluster partitions are repaired this way. Otherwise, the node leaves the cluster by changing back to the *unclustered* state. The destruction of a cluster is determined by all nodes leaving it.

NRP was modified as not to count the control messages sent by the clustering protocol. Otherwise, clusters would not be destroyed when there is actually no more traffic. This is because the clustering protocol keeps sending beacons as long as there are neighbours (as recognized by NRP) which ought to be clustered.

3) Example: Cluster Creation: To conclude the protocol description and to illustrate the cooperation of NRP and the clustering protocol in ODGMBC, the formation of a new cluster is described as an example. Refer to figure 3 for the depiction of the following sequence. Note that both nodes m and n initially are in the *unclustered* state (hollow circles).

1) After receiving and counting a sufficient amount of MAC frames from n, the NRP running on m recognizes n as a neighbour (indicated by the solid arrow).
2) Node m tells n about this event by sending a *NEIGHBOUR-INFORM* message (dashed arrow). At this point of time, m is in the *unclustered* state and does not have any neighbours. Hence, it is a potential leader and therefore starts the timer for the (RCC) leader election process.
3) Node n receives m's *NEIGHBOUR-INFORM* message. Now that n knows about m having recognized it as a neighbour, n also regards m as a neighbour (solid arrow from n to m).
4) Eventually, the timer m set in step 2 expires and causes m to declare itself leader. It does so by broadcasting a *LEADER-BEACON* message.
5) Upon receiving the *LEADER-BEACON*, node n looks up the originator in its data structures. It finds m recorded as a neighbour. As n currently is not member of any cluster, it joins in with m, forming a cluster of two nodes with m being the leader.

III. Evaluation Results

A. Simulation Setup

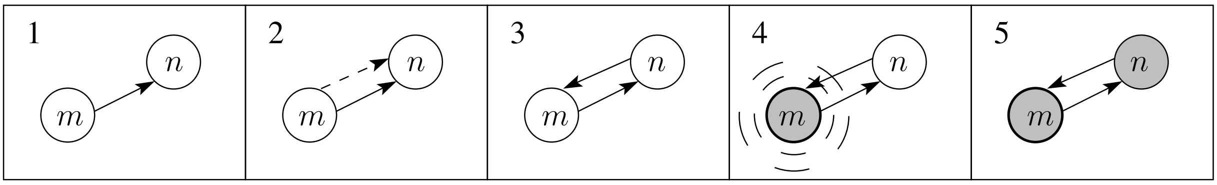
To evaluate ODGMBC's functionality and performance, a prototype module for the ad hoc network simulator GloMoSim v2.03 [2] was implemented. The deployment scenario is a pedestrian precinct. There are 300 pedestrians (nodes) dispersed over a quadratic area measuring 2000 m × 2000 m. Each pedestrian is equipped with an 802.11-capable device having a radio range of 250 m. The node speed ranges from 0.5 m/s to 2.0 m/s. A total of 215 nodes moves individually, whereas the remaining 85 nodes are divided among six groups with diameters from one to three hops. For node movement, the "Reference Point Group Mobility" (RPGM) model [11] was used. The group centers move according to the "Random Direction" model [12], with the mean epoch time set to 30 seconds. Individual motion is also generated using the RPGM model by creating groups with only one node. These nodes' positions and movements are identical to those of the respective group centers.

Network traffic is generated using a simple model: There is a fixed number of connections with fixed endpoints (per simulation run). Connection endpoints are chosen independently and identically distributed (i.i.d.) among all 300 nodes using a uniform random distribution. The connections start uniformly i.i.d. within the first 20% of simulated time, which was chosen to be 600 seconds (10 minutes). They stop simultaneously with the simulation. Over each connection, packets of 512 bytes length are sent with a constant rate of two packets per second from the first to the second endpoint. Transmission takes place during the full connection duration.

A load of 30 concurrent connections was put on the network, resulting in a maximum packet loss of 5%, which was confirmed in additional simulation runs. Routes are determined with AODV and its default parameters. Note that each of the results presented was obtained combining three i.i.d. random samples of the mobility scenario and three i.i.d. random samples of the traffic scenario. All results are averaged over the resulting 9 simulation runs, leading to reasonable statistical significance with comparably low simulation costs.

Although they may be chosen independently, the period length of the NRP and the period length of the leader beaconing were set to equal values, respectively. This decision will be re-evaluated in our future work, but the independent choice has the drawback of multiplying the non-negligible simulation costs with another factor.

ODGMBC was simulated with period lengths of 5 s and 60 s. For comparison, GMBC was simulated with the respec-



m recognizes n as a neighbour

m sends *NEIGHBOUR-INFORM* to n

n regards m as a neighbour

m declares itself leader, sends *LEADER-BEACON*

m and n form a cluster

Fig. 3. Example: Cluster creation with ODGMBC

tive parameters set to reflect period lengths of 5 s and 30 s. In both cases, the maximum cluster radius was set to be 3 hops. To recognize a node as a neighbour, one received packet per 6 seconds suffices.

B. Evaluated Performance Indicators

There are several performance indicators of interest for the evaluation of a clustering protocol (see also [13]). The number of leaders in the network and the average size of clusters give hints whether the protocol works properly. By taking into account the "noise" in both figures, a first estimate of cluster stability is obtained. As mentioned before, cluster stability is of high importance for the targeted application, namely automatic addressing and hierarchical routing. Therefore, the change frequency of cluster leaders and cluster members are additionally inspected. The mean duration of nodes being in the leader state and the mean sojourn time of nodes in different clusters also help in evaluating cluster stability.

Also, the overhead incurred by the clustering protocol has to be measured. It is normalized both as packets per period and bytes per second to be comparable across different choices of the period values.

C. Simulation Results

The results of the simulations made are described in the following sections.

1) Number of Leaders and Cluster Size: First, compare the number of leaders and average cluster size with ODGMBC using a 5 s period (see figure 4) as opposed to those of GMBC with the same period (see figure 5). The slowly ascending curve showing ODGMBC's average cluster size suggests that the protocol works as expected: It reacts to the increasing traffic load. Compared to GMBC, the curve showing the number of leaders with ODGMBC is considerably smoother. This is a first hint that cluster stability is better with ODGMBC than with GMBC.

The peak at the beginning of GMBC's leader curve is the result of a large number of leader election conflicts. They arise because all nodes nearly simultaneously begin clustering. Due to the high variability, clusters tend to be smaller and more leaders are present than with ODGMBC.

2) Leader Change Frequency: Comparing the leader change frequencies in both configurations (see figure 6), it becomes obvious that GMBC performs considerably worse (1.54 changes/second) than ODGMBC in this point (0.21 changes/second). Also see table I for the other configurations mentioned before and averages of the performance indicators regarding stability and overhead.

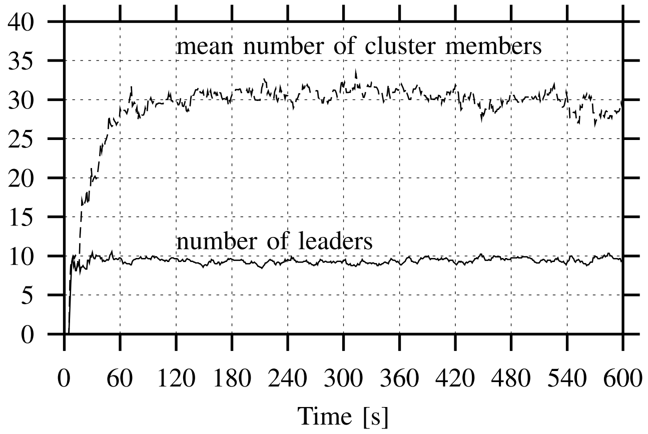


Fig. 4. Number of clusters and cluster size (ODGMBC, 5 s)

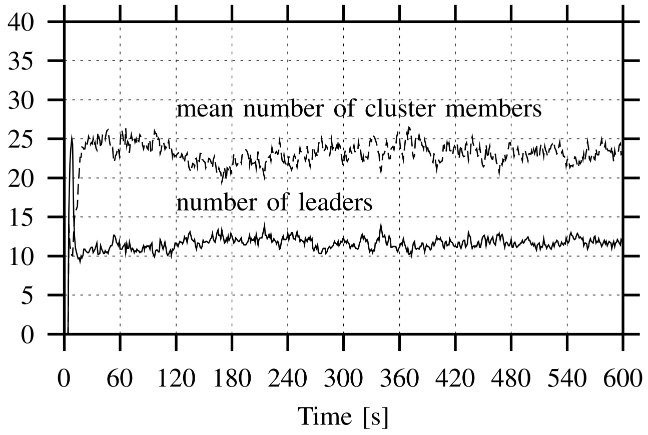


Fig. 5. Number of clusters and cluster size (GMBC, 5 s)

There is a simple explanation for the high stability of clusters with ODGMBC. Other than with GMBC, nodes "stick" to their leaders as long as possible – they only change their clusters when their previous leaders' beacons have not been overheard recently (refer to section II-C.2). Also in contrast to GMBC, not every node has the ability to declare itself leader in ODGMBC. Remember from section II-C.2 that

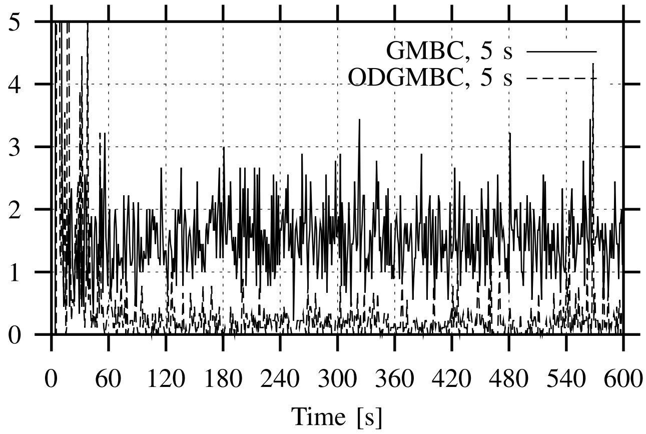


Fig. 6. Leader changes per second

only nodes which have recognized other nodes as neighbours *themselves* are allowed to do so. Last, because nodes keep a list of recently overheard leaders, they can immediately join an existing cluster when they need to. With GMBC, nodes do not keep a comparable list and therefore *always* create a new cluster in these situations. Many conflicts arise due to this design decision.

3) Cluster Change Frequency: Regarding the cluster change frequency, ODGMBC again clearly outperforms GMBC (see figures 7, 8). The same reasons as for the leader change frequency apply here.

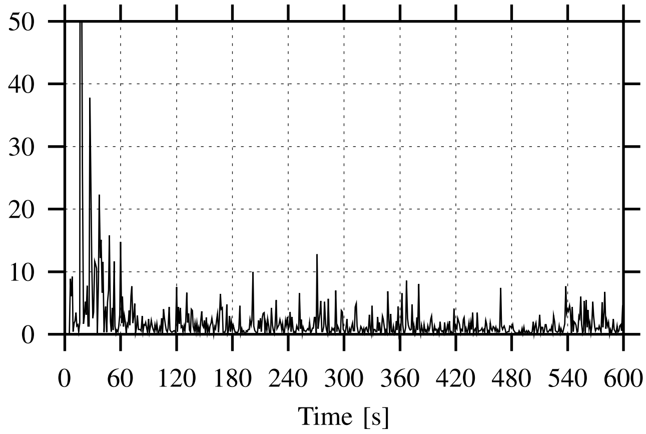


Fig. 7. Cluster changes per second (ODGMBC, 5 s)

4) Stability and Overhead: Table I gives an overview of the stability and overhead indicators for the simulated configurations. Once more, it can be seen that ODGMBC is much better than GMBC in terms of the cluster sojourn time and average time of nodes being in the leader state. Statistically, the significance of the difference could not be rejected at an error level of 5%.

For both protocols, the increase of the period durations lead to generally better performance indicators. This result has to be interpreted carefully. It only shows that, as expected, the less often status updates are made (i.e. packet are sent), the less often the state actually changes (i.e. nodes change their clusters). The longer the periods are, the more the risk of having stale state at the nodes increases. There is an obvious trade-off between the update frequency (which clearly is correlated with the generated overhead) and the actuality of state.

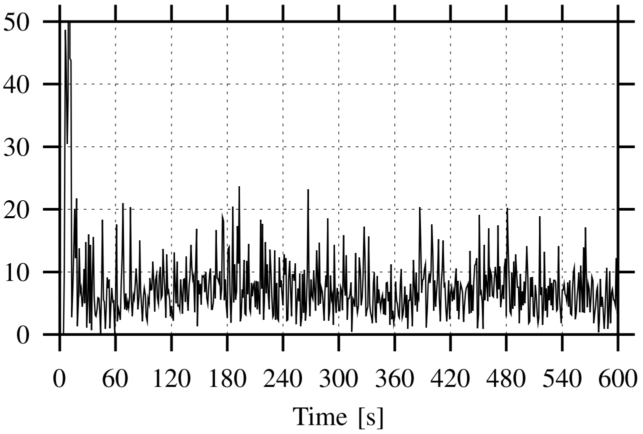


Fig. 8. Cluster changes per second (GMBC, 5 s)

Compared to GMBC, ODGMBC's overhead is slightly higher in the tested scenario. This mainly follows from the high overall traffic load. As ODGMBC is a reactive protocol, the overhead should be proportional to the network load. ODGMBC has to send more messages than GMBC because of its neighbourhood detection scheme. Also, these messages are quite long because the MAC addresses of the recognized neighbours are appended. In the scenario, nodes have 15 neighbours on average and MAC addresses are 48 bits long.

To sum up, our simulation results indicate that ODGMBC is working as desired. It reactively builds clusters of high stability. In its current design, protocol overhead is a drawback in dense traffic scenarios due to the neighbourhood detection scheme, but some ideas to reduce it are outlined in the last section.

IV. Related Work

For a thorough yet slightly dated review of clustering protocols and the goals associated with them, see [3]. [4] and [5] present some simple one-hop clustering mechanisms. The only reactive clustering protcol known to us is introduced in [1]. However, it presents a one-hop clustering protocol with the goal of collision reduction in flooding. Effects of different leader election rules on cluster stability are examined in [13].

Our work in part benefited from the insights provided in [10]. As mentioned before, "Random Competition based Clustering" strongly influenced our leader election procedure. The notion of group mobility was defined with the long-term stability of physical node neighbourhoods. These neighbourhoods themselves are defined by physical (radio) links. Physical link stability has been considered as a route selection

Configuration	Leader Changes [1/s]	Cluster Changes [1/s]	Leader Stability [s]	Cluster Sojourn Time [s]	Byte/s	Packets/Period
ODGMBC, 5 s	0.21	1.42	46.07	100.47	38.96	3.88
ODGMBC, 60 s	0.07	1.07	70.75	181.71	5.80	9.10
GMBC, 5 s	1.54	7.21	14.53	28.96	6.55	1.93
GMBC, 30 s	0.36	1.79	40.26	89.94	5.90	10.40

TABLE I

SIMULATION RESULTS – STABILITY AND OVERHEAD

metric by several independent research groups. See [9], [14] and [15] for results.

V. CONCLUSION AND FUTURE WORK

In this paper, we have investigated whether a reactive clustering protocol with group mobility is feasible. A possible solution has been presented. The protocol was implemented and tested by means of simulation. It could be shown that the approach indeed yields stable clusters. The key improvement of ODGMBC over a proactive clustering protocol is that it does not consume any bandwidth when no clusters are currently needed.

Nevertheless, there are several aspects of enhancements left. Some possible approaches for reducing the protocol overhead are:

- One possibility is to reduce the size of the *NEIGHBOUR-INFORM* messages by removing the MAC addresses from them. Now nodes take part in the clustering process which have not been considered before. These are the nodes whose packet counts do not exceed the threshold level. This change could compromise cluster stability because the number of potential leaders increases. More conflicts arise during the initial election phase. The effects of this subtle change have to be evaluated in detail.
- Another option is not to use *NEIGHBOUR-INFORM* messages at all. This approach might sound strange at first sight, as the whole clustering protocol relies on these messages. However, in the design process, it was not taken into account that nodes could count how many packets they sent per period *themselves*. Thus, they would automatically know whether there is the need for clustering without sending any additional message. The total overhead would reduce to that of the *LEADER-BEACON* messages. By this modification, the detection of unidirectional links is no more possible and packet loss is not considered. Again, care has to be taken for the possible changes in cluster stability.

Apart from these protocol alterations, another direction in future research has to be the coupling of ODGMBC with an address auto-configuration mechanism. A large degree of interplay between the two mechanisms of clustering and auto-configuration is required. See e.g. [16] for a discussion of issues arising when using auto-configuration. Then, a routing protocol and location management can be layered on top. These problems are both challenging and interesting steps in future research.

ACKNOWLEDGEMENT

Many thanks to the anonymous reviewer who provided valuable comments. Funding was provided by the German Fed. Ministry of Education and Research as part of *IPonAir*.

REFERENCES

[1] T. J. Kwon and M. Gerla, "Efficient flooding with passive clustering (pc) in ad hoc networks," *ACM Computer Communication Review*, vol. 32, no. 1, 2002.

[2] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla, "Glomosim: A scalable network simulation environment," UCLA Computer Science Department, Tech. Rep. 990027, 1999.

[3] M. Steenstrup, "Cluster-based networks," in *Ad Hoc Networking*, C. E. Perkins, Ed. Upper Saddle River, NJ: Addison-Wesley/Pearson Education, 2000, pp. 75–138.

[4] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, 1997.

[5] S. Basagni, "Distributed clustering for ad hoc networks," in *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, A. Y. Zomaya, D. F. Hsu, O. Ibarra, S. Origuchi, D. Nassimi, and M. Palis, Eds. Perth/Fremantle, Australia: IEEE Computer Society, 23.–25. June 1999, pp. 310–315.

[6] P. Krishna, N. Vaidya, M. Chatterjee, and D. Pradhan, "A cluster-based approach for routing in dynamic networks," *ACM SIGCOMM Computer Communication Review*, pp. 49–65, Apr. 1997.

[7] D. J. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Transactions on Communcations*, vol. COM-29, no. 11, pp. 1694–1701, Nov. 1981.

[8] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *ACM Wireless Networks*, vol. 8, no. 2, pp. 153–167, Mar. 2002.

[9] C.-K. Toh, *Ad hoc mobile wireless networks: protocols and systems*. Upper Saddle River, New Jersey: Prentice Hall PTR, 2002.

[10] K. Xu, X. Hong, and M. Gerla, "An ad hoc network with mobile backbones," in *Proceedings of the IEEE International Conference on Communications (ICC)*, New York, NY, Apr. 2002.

[11] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, "A group mobility model for ad hoc wireless networks," in *Proceedings of ACM/IEEE MSWiM 1999*, Seattle, WA, Aug. 1999.

[12] C. Bettstetter and C. Wagner, "The spatial node distribution of the random waypoint mobility model," in *Proceedings of the 1st German Workshop on Mobile Ad-Hoc Networks (WMAN)*, ser. GI Lecture Notes in Informatics, no. P-11, Ulm, 25.–26. Mar. 2002, pp. 41–58.

[13] C. Bettstetter and R. Krausser, "Scenario-based stability analysis of the distributed mobility-adaptive clustering (dmac) algorithm," in *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Long Beach, CA, USA, 4.–5. Oct. 2001, pp. 232–241.

[14] X. Hong and M. Gerla, "Dynamic group discovery and routing in ad hoc networks," in *Proceedings of the First Annual Mediterranean Ad Hoc Networking Workshop (Med-hoc-Net 2002)*, Sardegna, Italy, Sept. 2002.

[15] M. Gerharz, C. de Waal, M. Frank, and P. Martini, "Link stability in mobile wireless ad hoc networks," in *Proceedings of the IEEE Conference on Local Computer Networks (LCN)*, Tampa, Florida, Nov. 2002.

[16] K. Weniger, "Passive duplicate address detection in mobile ad hoc networks," in *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, vol. 3, Mar. 2003, pp. 1504–1509.