# MEACA: Mobility and Energy Aware Clustering Algorithm for Constructing Stable MANETs

Yi Xu and Wenye Wang
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27606
Email: {yxu2, wwang}@ncsu.edu

*Abstract*— **Node clustering is a technique to mitigate the topology changes in mobile ad hoc networks (MANETs). It stabilizes the end-to-end communication paths and maximizes the path lifetime. It also improves the network scalability such that the routing overhead does not become tremendous in large-scale ad hoc networks. Its effectiveness, however, depends largely on the cluster stability which is measured by the lifetime of the cluster heads and the membership time of the cluster members. The existing clustering algorithms do not achieve this objective well. In this paper, we show that the cluster stability is fundamentally related to the mobility and the energy status of the nodes in the network and propose a new clustering algorithm using the node mobility and energy information. Our new algorithm maximizes the cluster stability by choosing the low mobility and high energy nodes to be the cluster heads and by keeping the constructed clusters unchanged to the extent of their maximum possible lifetime. In addition, the proposed clustering algorithm performs equally well in large-scale ad hoc networks as in the small ones. We also show that it is a near-optimal clustering algorithm.**

## I. Introduction

Self-organization is the characterizing feature of the mobile ad hoc networks. There are no infrastructure-like entities in the network that implement predefined control functions over the network. Every node is a peer to the others and they establish the network cooperatively. The communication between two non-neighboring nodes is realized through the relay of intermediate nodes that serve as routers for them. Due to the node mobility and the energy constraint, the network architecture is not always stable. A node participating in a route for others might quit before the end-to-end communication finishes, resulting in path failure and re-discovery. During the path failure time, the end nodes suffer from packet delays and packet losses. The path re-discovery introduces extra network overhead and thus consumes network bandwidth, which becomes significant when the node dynamics increase. Besides, scalability is another important issue of the ad hoc networks. As the node population grows, an efficient routing protocol that locates the destination node and establishes a path to it with affordable network resource expenses is desired. It has been proven that the reactive routing strategies [3], [4] perform better than the proactive strategies [1], [2] in highly dynamic networks. However, as network flooding is used in the reactive routing strategies for route discovery, scalability is still a challenging issue when the node population is large.

The cluster concept [5] was introduced as an approach to address the path stability and the network scalability problems. The path stability is defined to be the lifetime of a path and the network scalability measures the routing overhead increase in large networks over the small networks. In a clustered network, the nodes aggregate into different groups, named *clusters*. In some scenarios, the cluster as a whole is stationary, with the individual nodes moving around inside. In other scenarios when a group of nodes have common interest and similar movement traces, the cluster moves as a whole. In both cases, routing is viewed in two tiers: inter-cluster routing and intra-cluster routing. Determined by the specific mobility patterns, either the inter-cluster path or the intra-cluster path is stable over a long time. Thus the clustered network structure improves the path stability. Path discovery is also simplified with the clusters. The destination node is located cluster by cluster instead of through flooding every node in the network. Each cluster has a head node that knows the membership of its cluster, so it is sufficient to query only the cluster heads to discover the path.

Since the purposes of forming clusters are to stabilize the end-to-end communication paths and to improve the network scalability, the *cluster stability* must be considered, which is defined to be the lifetime of the cluster heads and the membership time of the cluster members. Unstable clusters could jeopardize both objectives. When a cluster is reformed, both the inter-cluster and the intra-cluster routes change. Also, the cluster membership changes and the re-clustering overhead offsets the benefit gained from the cluster-by-cluster path discovery. The widely used lowest-ID clustering algorithm [6], [7] achieves simple and fast clustering result, but does not guarantee cluster stableness. Cluster reforming takes place frequently as the node mobility increases. The fundamental reason is that the node ID does not reflect a node's suitability to become a cluster head, which would instead be more preferably determined by a node's mobility and energy status. Similarly, many other existing clustering algorithms also do not consider the *cluster stability* as the design objective and, therefore, experience frequent cluster changes.

We propose a new clustering algorithm in this paper. Our algorithm targets the *cluster stability* by using the node mobility and energy information as the basis for making clustering decisions. The contributions of this paper are:

1) we generalize the lowest-ID algorithm and reveal the underlying reasons that result in the cluster instability;
2) we design a new clustering algorithm that stabilizes the clusters better than the generalized lowest-ID algorithm;
3) we evaluate the optimality of our new algorithm and show it is near-optimal.

The rest of this paper is structured as follows. We describe the related work, generalize and investigate the lowest-ID algorithm in Sectioin II. We present the design of our new clustering algorithm in Section III. Section IV studies the performance of our new clustering algorithm in comparison to the generalized lowest-ID algorithm using simulations and evaluates the algorithm optimality. Finally, we conclude this paper in Section V.

## II. Related Work

The lowest-ID clustering algorithm is a simple and fast technique to form clusters. This algorithm uses the node ID, a unique node identifier, to determine which nodes to become cluster heads and which not. The nodes exchange their ID numbers periodically. The nodes having the lowest ID numbers in their neighborhood become the cluster heads and the other nodes become the cluster members. If a member node loses the contact with its head, it re-determines its new role and new head. In addition, a node changes its head and/or its role whenever it hears a head with a lower ID than its current head.

The weight-based clustering algorithm [8] is similar to the lowest-ID algorithm. Each node in the network is assigned a weight. The nodes with the highest weights in their respective neighborhood become the heads, and the other nodes become the members. A node changes its head and/or role if another head node heavier than its current head comes into its neighborhood. In an effort to alleviate the frequent cluster changes, the revised weight-based algorithm [9] relaxes the role update requirement such that the change takes place only when the new head is significantly heavier than the current one. Simulation shows the revised scheme enhances the *cluster stability* [10].

In fact, the lowest-ID algorithm and the weight-based algorithm can be viewed as two instances of a generalized lowest-ID algorithm. In this generalized lowest-ID algorithm, each node has a uniquely predefined identifying attribute that ranks the node's priority to become a cluster head. Lower attribute value indicates higher priority. In the lowest-ID algorithm, this attribute is the node ID. In the weight-based algorithm, this attribute is the inverse of the weight. The generalized lowest-ID algorithm has the cluster instability problem due to two reasons. First, the arbitrarily predefined ID attribute does not relate to a node's stability to be a cluster head. The node's stability is determined by its mobility and energy status, rather than its ID. As the result, the nodes with low IDs are probably not the best candidates for cluster heads. Second, the algorithm requires a node switch to a new head whenever the new head has lower ID than its current head, even in the case that the current cluster is still valid. This requirement terminates the existing cluster prematurely. The head selection using node IDs and premature head re-selection result in frequent cluster changes.

Node mobility is considered in [11] to determine the cluster heads. In this algorithm, every node monitors its speed relative to the neighbors and chooses the node with the lowest relative speed to be its head. This algorithm improves the *cluster stability*, but there is a potential convergence problem. Because the relative speed measurement on a node is not globally deterministic such that different nodes may have different measurements, the nodes may not be able to reach unanimous decisions on their roles. The Weighted Clustering Algorithm (WCA) [12] combines the node degree, transmission power, mobility, battery power and uses the weighted sum of these node status metrics to determine the cluster heads. The $(\alpha, t)$–Cluster algorithm [13] evaluates the intra-cluster reachability of the mobile nodes. Clusters are dynamically constructed to ensure path availability in each cluster.

Other clustering algorithms consider communication cost and energy consumption [14], [15], use node contention to select cluster heads [16]–[18], and bound cluster sizes [19]. The *cluster stability* is not the design objective of these algorithms.

In this paper, we target a new clustering algorithm that forms stable clusters. Stable clusters will have the benefits of providing stable end-to-end communication paths and enabling good network scalability. Our algorithm takes the node mobility and energy into account, but differs from the generic weight-based schemes in that we have designed the specific indicator metric to quantitatively measure each node's suitability to become a cluster head rather than using the general concept of weight. As mobility is the major cause for the network topology changes, the node mobility status takes the priority in our indicator metric and the node energy status plays the subsidiary role, which is different from the WCA where all the factors are mixed through using pre-assigned factor weights.

## III. The Mobility and Energy Aware Clustering Algorithm

Next, we define the clustering problem first. Then we present our clustering algorithm, named the Mobility and Energy Aware Clustering Algorithm (MEACA). Our algorithm uses the node mobility and energy status to evaluate and select the most stable nodes to be the cluster heads. Besides, it avoids premature cluster head re-selection to stabilize the formed clusters.

### A. Problem Definition

An ad hoc network is described by a node set $V = \{v\}$ and a node connectivity set $E = \{e_{ij}\}$. The node set $V$ gives all the nodes in the network and the connectivity set $E$ denotes all the 1-hop links among the nodes in $V$. We make a few assumptions on the network. First, every link is symmetric. Second, every node has an ID or node address that identifies the node uniquely. Third, every node is able to estimate its energy lasting time based on its energy usage. Fourth, every node reports its status accurately when the nodes exchange information.

We define a cluster to be a subset of $V$ where one node is selected to be the cluster head and the rest are affiliated members. The nodes in a cluster are geographically close to one another. The radius of a cluster is measured by the number of hops from the cluster head to the furthest member node in its cluster. The cluster radius is a tradeoff issue between the inter-cluster routing and the intra-cluster routing complexity. In this paper we define the cluster radius to be 1 hop. That is, every member node is in direct contact with its cluster head. Fig. 1 illustrates example clusters in an ad hoc network, where nodes $v_2$, $v_9$, $v_{12}$, $v_{15}$, and $v_{17}$ are cluster heads, and all the other nodes are cluster members.

The *cluster size* is defined to be the number of nodes in the cluster, counting both the cluster head and the cluster members. If

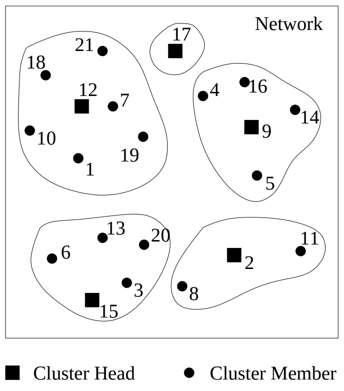


Fig. 1. An example of the clustered ad hoc network.

there are $N_h$ head nodes and $N_m$ member nodes in the network at a moment, the average *cluster size* is defined to be $\overline{s} = \frac{N_h+N_m}{N_h}$.

The *cluster stability* is evaluated in two metrics. The *cluster head lifetime* is the time duration when a node remains in the cluster head role. It starts when the node becomes a cluster head and ends when the node stops its head role. The *cluster membership time* is the time duration when a node remains affiliated to a cluster head. It starts when the member node has selected its cluster head and ends when it switches to another cluster head. To capture these two performance metrics accurately, we will use the averaged measurements of these two metrics in the later part of this paper.

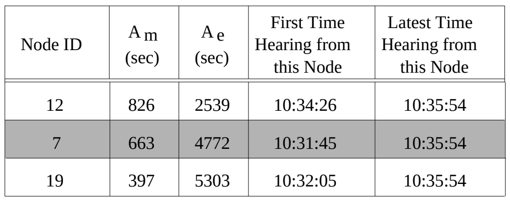
The objective of our clustering algorithm is to form stable clusters in ad hoc networks, where the stableness is measured quantitatively by the *cluster head lifetime* and the *cluster membership time*. To summarize, the algorithm forms clusters that satisfy the following requirements.

- Every node in the network becomes either a cluster head or a cluster member.
- Every node is associated with one and only one cluster.
- Every member node is 1-hop away from its cluster head.
- The average *cluster size* is maximized.
- The *cluster stability* is maximized.

## B. Basic Idea of MEACA

The Mobility and Energy Aware Clustering Algorithm (MEACA) clustering algorithm works in a distributed manner as in the lowest-ID algorithm. The nodes in the network have different priorities to become cluster head. They exchange their priority values to determine who will become the heads and who will become the members. Every node makes its own decision after having collected the priority values of all its neighbors. Because each node's priority value is globally deterministic, the nodes in the network are able to reach unanimous decisions on their roles, though each node decides independently.

Unlike the lowest-ID algorithm that fixes a node's priority level beforehand, MEACA sets the priority using the node's mobility and energy status. For this purpose, each node has a *mobility attribute* and an *energy attribute*. Both are kept up to date. When determining its cluster head, a node selects in its neighborhood the node with the relatively lowest mobility and highest energy using these two attributes. The selected node could be one of its



| Node ID | $A_m$ (sec) | $A_e$ (sec) | First Time Hearing from this Node | Latest Time Hearing from this Node |
|---|---|---|---|---|
| 12 | 826 | 2539 | 10:34:26 | 10:35:54 |
| 7 | 663 | 4772 | 10:31:45 | 10:35:54 |
| 19 | 397 | 5303 | 10:32:05 | 10:35:54 |

Fig. 2. An example neighborhood table and cluster head selection.

neighbors or the selecting node itself. After the cluster head has been determined, the member node registers itself with its cluster head. Re-clustering takes place only when a member node has lost contact to its head or a head node has lost contact to all its members. In other words, a node will not change its head and/or role as long as its current cluster remains valid.

## C. Algorithm Description

The MEACA algorithm requires two node attributes to determine a node's priority to become a cluster head. The *mobility attribute* $A_m$ measures a node's mobility stability. It is defined to be the sum of the neighboring time between the node and each of its current neighbors. If a node has many neighbors and it has been with these neighbors for a long time, its $A_m$ will have large value, indicating that it is a stable node in the mobility sense. The *energy attribute* $A_e$ measures the remaining time of a node before its energy is used up. High $A_e$ indicates that the node is stable in the energy sense. We assume that each node in the network is able to determine its $A_m$ and $A_e$ at any time.

**Advertisement of Attributes.** Every node broadcasts advertisements to its neighborhood periodically to exchange attributes. A node includes its up-to-date $A_m$ and $A_e$ in the advertisements. If the node has become a member node already, it sets $A_m$ and $A_e$ to null in the advertisements to inform other nodes not to choose it as their cluster head. Every node maintains a neighborhood table to keep the received advertisements. For each neighbor node, the table keeps the node ID, its $A_m$, its $A_e$, the time of receiving the first advertisement from it, and the time of receiving the latest advertisement from it. When an advertisement is received, the corresponding entry of the sending node in the table is updated. If a node does not receive advertisements from a neighbor node any more, the neighbor node is cleared from the neighborhood table. Thus the neighborhood table always keeps the current mobility and energy status information of the neighbor nodes. Fig. 2 illustrates the format of the neighborhood table. Every node can easily calculate its own $A_m$ from its neighborhood table.

**Formation of Clusters.** Initially all the nodes in the network are in the role-undecided state. The nodes use their neighborhood tables to determine their respective roles. A node chooses its cluster head as follows. First, it sorts the nodes in its neighborhood table from the highest $A_m$ to the lowest $A_m$, including itself. We denote the highest $A_m$ as $\max(A_m)$. Second, the node determines a mobility threshold $A_m^* = \alpha{\cdot}\max(A_m)$, where $\alpha{\in}(0, 1)$. The node uses the mobility threshold to eliminate the unstable nodes of which the $A_m$'s are lower than $A_m^*$. Third, in the remaining nodes, the node selects the one with the highest $A_e$ to be its cluster head. If the node selects itself, then it becomes a cluster head; otherwise, it becomes a member of the selected cluster head. Fig. 2 gives an example of the cluster head selection. There are three nodes in this example table. Assuming $\alpha = 0.8$, $A^*_m = 0.8 \times 826 = 660.8$, node 7 and 12 are shortlisted for further consideration. As node 7 has higher remaining energy, it is selected to be the cluster head.

**Finalization of Cluster Roles.** When a node has decided to become a cluster head, its role is finalized. If it chooses to become a member of another node, it sends a registration message to the selected node. The selected node acknowledges the registration if it has finalized its role to be a cluster head. After receiving the acknowledgment, the selecting node's member role is finalized. If the selected node has not made its decision yet, it ignores the registration. In this case, the selecting node will continue trying until it can register with the selected cluster head successfully.

**Reformation of Clusters.** After a node has determined its role, it starts to maintain a registration table. If the node is a cluster head, the registration table keeps the IDs of all its members. If the node is a cluster member, the registration table keeps the ID of its cluster head. The node uses its registration table to judge when it needs to re-cluster. The node will re-cluster only when its registration table becomes empty. In other words, the node will not re-cluster until it has lost contact to all of its members if it is a head node, or until it has lost contact to its head if it is a member node.

### D. Discussions

We have defined earlier the requirements for our clustering algorithm. Now we show that the MEACA algorithm satisfies these requirements.

- Every node becomes either a cluster head or a cluster member. This is true because every node can always locate a node in its neighborhood to be its cluster head. If the selected node is itself, the node becomes a head node. If the selected node is not itself, the node becomes a member node.
- Every node is associated with one and only one cluster. This is true since a node chooses one and only one node to be its cluster head, there is no cluster membership overlapping.
- Every member node is 1-hop away from its cluster head. This is true as the selected cluster head is from the neighborhood table in which all the nodes are 1-hop away.

The requirements on the *cluster size* maximization and the *cluster stability* maximization are algorithm optimization issues and we defer the discussion to Section IV.D.

## IV. Algorithm Evaluation

We evaluate the performance of the MEACA clustering algorithm in comparison to the generalized lowest-ID algorithm by simulations. We are interested in the algorithm optimality issues, including the *cluster size* and the *cluster stability*, and the *algorithm scalability* in different network sizes.

The simulation environment is set up as follows. We use the NS-2 simulator [20]. In the simulator, we define a new node mobility model that generates and maintains uniform node distribution in the network area throughout the simulation duration. The NS-2 default Random Waypoint Model is not used due to its non-uniform node distribution and speed decay problems [21]. In our node mobility model, a node alternates in the stationary and the moving states. In the stationary state, the node stays where it is for an exponentially-distributed random time. At the end of the stationary state, the node chooses a random direction and a random distance to travel. The direction and the distance are uniformly distributed. The node travels in a constant speed, which is also a uniformly-distributed random variable. If the node hits the network boundary before finishing its planned travel distance, it bounces back into the network to finish the remaining travel distance. When it reaches the destination, it transits into the stationary state again. The node energy lasting time is modeled as an exponentially-distributed random variable. When its energy is used up, the node stops its activity for some time to recharge its battery. The node joins the network again after recharging.

TABLE I
Mobility Pattern Configurations.

| Pattern | Pause Time (min) | Speed Range (m/s) | Maximum Distance (m) |
|---|---|---|---|
| 1 | 2 | (1,9) | 1000 |
| 2 | 4 | (1,7) | 1000 |
| 3 | 6 | (1,5) | 1000 |
| 4 | 8 | (1,3) | 1000 |

We have simulated the 120-node and the 240-node network sizes in an area of $2000\mathrm{m} \times 2000\mathrm{m}$. The node communication radius is 250m. Initially the nodes in the network are all in the role-undecided state. The clustering algorithm begins two minutes after the simulation starts to allow some time for the nodes to collect their neighborhood information. After the initial clustering, the nodes monitor their neighborhood continuously and re-cluster when necessary. We use four different mobility configurations to model different mobility levels. Table I lists their configuration details. The pause time is the duration when a node stays in the stationary state. The speed range gives the lowest and the highest possible speeds that a node can have when in the moving state. The maximum distance is the furthest distance that a node can travel in one moving state. Among them the Pattern-1 has the highest mobility and the Pattern-4 has the lowest mobility. The average node energy lasting time is 2 hours. In all the simulations $\alpha = 0.9$. Each measurement in the simulation is averaged over 10 simulation runs, with 1-hour simulation duration in each run.

### A. Cluster Size

We count the number of clusters and their average sizes every 10 minutes for the generalized lowest-ID algorithm and the MEACA algorithm. The results are shown in Fig. 3 and Fig. 4. From the figures, we see that the MEACA algorithm forms slightly more clusters than the lowest-ID algorithm. MEACA has 50–60 clusters, dependent on the node mobilities, while the lowest-ID has about 40 clusters throughout the simulation. Accordingly, MEACA has slightly smaller average *cluster size* than the lowest-ID. The average *cluster size* is 4–5 using MEACA, while it is about 6 using the lowest-ID. In the *cluster size* comparison, the lowest-ID is slightly better than the MEACA algorithm. This is because the lowest-ID algorithm keeps the cluster heads out of the direct communication range from one

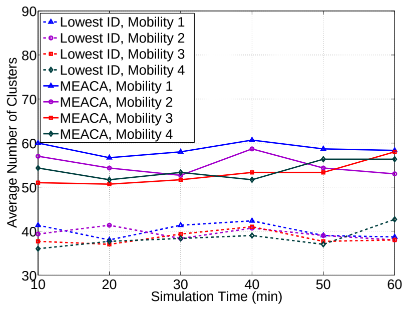


Fig. 3. Comparison of the average number of clusters, 240 nodes.

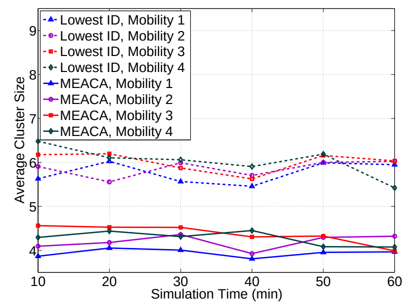


Fig. 4. Comparison of the average cluster sizes, 240 nodes.

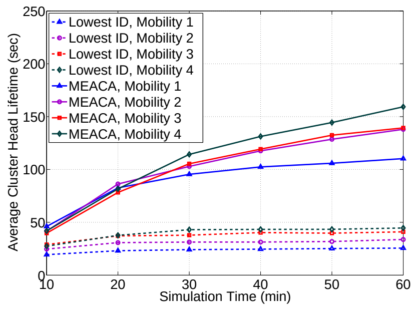


Fig. 5. Comparison of the cumulative average cluster head lifetime, 240 nodes.

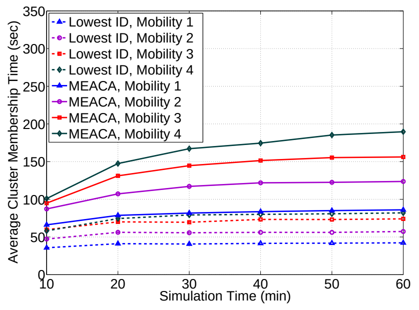


Fig. 6. Comparison of the cumulative average cluster membership time, 240 nodes.

another such that a cluster's coverage area is bigger than the MEACA algorithm on average. To see this, recall that in the lowest-ID algorithm a head node switches to be a member node when it hears another head with a lower ID. As the result, two cluster heads can never be in each other's neighborhood. In comparison, MEACA allows two cluster heads to be close to each other. This takes place when two cluster heads move toward each other and each is managing a cluster of affiliated member nodes. The average distance between cluster heads is then shorter in MEACA than in the lowest-ID. Therefore MEACA forms more clusters with smaller cluster sizes than the lowest-ID in the network. We also observe that the average *cluster size* in MEACA increases slightly when the node mobility decreases. As the nodes slow down, the chance that two cluster heads move close reduces, so the clusters become bigger on average. The lowest-ID algorithm, however, is invariable to node mobility in the *cluster size* measurement.

## *B. Cluster Stability*

*1) Cluster Head Lifetime:* Fig. 5 shows the cumulative average *cluster head lifetime* at 10-minute intervals. In the lowest-ID, a cluster head lasts for 20–40 seconds on average. In MEACA, the average *cluster head lifetime* increases and tends to stabilize as the simulation time goes. The initial *cluster head lifetime* is short due to the reason that the nodes in the network do not have enough time to have accurate mobility measurement and the selected cluster heads are not the most stable ones. As the simulation proceeds, the *cluster head lifetime* converges. At the simulation time of 1 hour, the average *cluster head lifetime* in MEACA is 110–160 seconds, which is significantly longer than the lowest-ID. MEACA has more stable cluster heads than the lowest-ID because it chooses the most stable nodes to be the cluster heads and reforms a cluster only when the existing cluster is broken. When a cluster head is selected, it is faithful to its role until all its member nodes have left it. Thus the cluster head achieves its maximum possible lifetime. The lowest-ID, on the contrary, changes a cluster head's role prematurely when another cluster head moves in. In both algorithms, we observe that the *cluster head lifetime* is shorter with high node mobility than with low node mobility. In MEACA high node mobility causes the member nodes to move away from their cluster heads fast, resulting in short *cluster head lifetime*. In the lowest-ID high node mobility increases the chance of two head nodes moving close and changing the role of one of them, therefore shortening the *cluster head lifetime*.

*2) Cluster Membership Time:* Fig. 6 compares the cumulative average *cluster membership time* at 10-minute intervals. Alike to the *cluster head lifetime*, MEACA has significantly longer *cluster membership time* than the lowest-ID. In the lowest-ID, a member node is affiliated to its cluster head for 40–80 seconds on average. In MEACA, the average *cluster membership time* converges after 30 minutes of the simulation time. The average is 80–200 seconds at the simulation time of 1 hour. MEACA achieves longer *cluster membership time* because it avoids premature cluster head re-selection. A member node stays with its head node as long as they are still in touch. The node's membership duration achieves the maximum in this way. In the lowest-ID, a member node can possibly switch between clusters prematurely when a new head node comes. We observe that node mobility affects a node's *cluster membership time* in a similar manner as in the *cluster head lifetime*. The node membership time is shorter with high node mobility than with low node mobility in both MEACA and

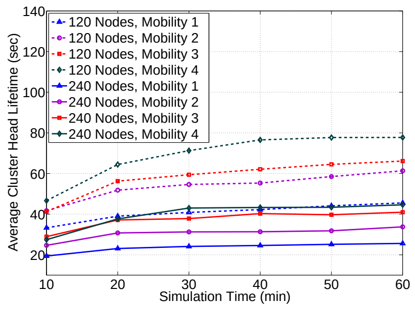


Fig. 7. Scalability of the lowest-ID algorithm: cluster head lifetime.

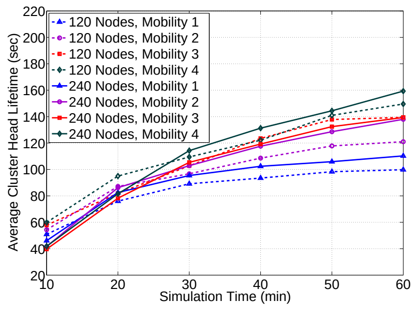


Fig. 9. Scalability of the MEACA algorithm: cluster head lifetime.

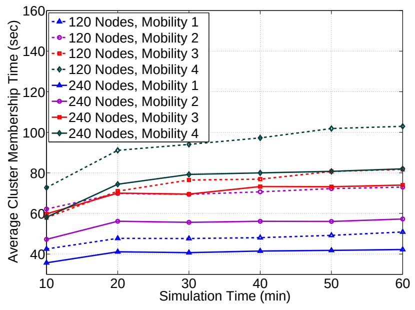


Fig. 8. Scalability of the lowest-ID algorithm: cluster membership time.

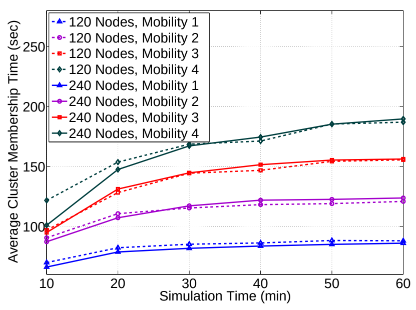


Fig. 10. Scalability of the MEACA algorithm: cluster membership time.

the lowest-ID algorithms. In MEACA, a node with high mobility tends to leave its cluster head fast, thus having short membership time. In the lowest-ID, a member node is likely to meet new cluster heads of lower IDs in high mobility environment, so its membership time with the original cluster head is short.

## C. Algorithm Scalability

*1) Scalability of the Lowest-ID Algorithm:* Fig. 7 and Fig. 8 give the simulation results of the average *cluster head lifetime* and the average *cluster membership time* using the lowest-ID algorithm in different network sizes. The two figures show that the clusters become less stable as the node population grows. For example, in the mobility Pattern-4, the average *cluster head lifetime* decreases from 78 seconds to 45 seconds when the node population increases from 120 nodes to 240 nodes, and the average *cluster membership time* decreases from 103 seconds to 82 seconds. The cluster stability degradation is accounted for by the increase of the premature re-clustering. As the node population grows, the node density increases, so the chance increases that a node hears a new cluster head with a lower ID than its current cluster head and thus re-clusters.

*2) Scalability of the MEACA Algorithm:* Fig. 9 and Fig. 10 compare the simulation results of the *cluster stability* using the MEACA algorithm in different network sizes. They show that the *cluster head lifetime* increases slightly when the node population grows, and the *cluster membership time* remains almost the same. The *cluster head lifetime* increases because the average *cluster size* becomes bigger as the node population increases, so it takes longer time for all the member nodes in a cluster to leave their head node. However, as the member nodes are independent from one another, the increase of *cluster size* does not change each member node's sojourn time in the cluster, so the *cluster membership time* does not change. Compared to the lowest-ID algorithm, MEACA has better scalability in large-sized networks.

## D. Algorithm Optimality

Till now we have demonstrated that the MEACA algorithm outperforms the lowest-ID algorithm in forming stable clusters. A further thought on the clustering problem naturally leads to the question: what is the optimal clustering algorithm? To answer this question, we must make it clear what desired properties a clustering algorithm should possess. First, the algorithm should cluster the network into small number of clusters. In 1-hop clusters, the increase of *cluster size* does not increase the intra-cluster routing complexity, except the demand for more memory space to manage the cluster membership. The complexity of the inter-cluster routing, however, is proportional to the total number of clusters in the network. Therefore, the optimal clustering algorithm should populate each cluster as much as possible to minimize the total number of clusters. We note that oversizing could consume the cluster head's energy very fast and shorten the cluster lifetime, so the *cluster size* maximization should be understood as being upper bound by the affordable sizes of the cluster heads. Second, because the cluster stableness is the basis for providing stable end-to-end communication paths and enabling good network scalability, the optimal clustering algorithm should stabilize each cluster. We identify the properties of the optimal clustering algorithm to be: 1) the algorithm maximizes the *cluster size* within the 1-hop cluster radius and the cluster head's affordable load limits; 2) the algorithm maximizes the *cluster head lifetime* and the *cluster membership time*.

We note that the *cluster size* is upper bounded by a limit determined by the node density such that any clustering algorithm achieves an average *cluster size* lower than this upper bound. In a network of $N$ nodes covering an geographical area of $a \times a$ $\mathrm{m}^2$, the node density is $\frac{N}{a^2}$. For $r$-meter node communication range, a cluster head covers an area of at most $\pi r^2$ $\mathrm{m}^2$ and forms a cluster of at most $\frac{\pi r^2 N}{a^2}$ nodes. Because in all the distributed clustering algorithms no entity has the global view of all the nodes' locations, the algorithms cannot guarantee sufficient spacing of the cluster heads to avoid cluster overlapping in their geographical coverages. As the result, some clusters achieve the maximum size but others do not. So the average *cluster size* is lower than the upper limit $\frac{\pi r^2 N}{a^2}$. Among all the distributed clustering algorithms, the lowest-ID has the largest *cluster size*, because it maximizes the inter-cluster distance by requiring the cluster heads to be out of the direct communication range of one another, which minimizes the cluster overlapping. The MEACA algorithm also maximizes the inter-cluster spacing when the clusters are initially constructed, but relaxes the maximum inter-cluster spacing requirement thereafter to allow the maximum *cluster stability*. The simulation resutls show that the average *cluster size* of MEACA is slightly smaller than the lowest-ID, indicating that MEACA is near-optimal in terms of the *cluster size*.

On the *cluster stability* aspect, the MEACA algorithm achieves the optimal *cluster head lifetime* and the optimal *cluster membership time*. The upper limits on the *cluster head lifetime* and the *cluster membership time* are determined by the node mobility and energy status. A cluster head can continue to be in the head role only when it remains in contact with its members. When it has lost the contact due to either mobility or energy reasons, it is not a cluster head any more. The only exception is that it cannot find another cluster head to join and has to be a cluster head to take care of itself. Similarly, a cluster member's membership ends naturally when it has lost the contact to its cluster head. These mobility and energy limits upper bound the achievable *cluster head lifetime* and *cluster membership time* of any clustering algorithm. MEACA does not change a cluster unless these limits are reached, so it achieves the longest possible *cluster head lifetime* and *cluster membership time*. The lowest-ID, in comparison, reforms a cluster before these mobility and energy limits are reached, so it is less optimal.

## V. Conclusions

In this paper we have studied the clustering problem in ad hoc networks. The clustering technique is used to manage large-scale ad hoc networks in an efficient and scalable way. We find that, due to different design goals, the clustering algorithms presented in the literature do not construct clusters stable enough to provide efficient end-to-end communication paths and to achieve good network scalability. The underlying reason is that they construct clusters without considering the node mobility and energy status, which is fundamentally related to the cluster stability. Based on this understanding, we have proposed the MEACA clustering algorithm that uses the node mobility and energy information to stabilize the clusters. Simulation results show that MEACA achieves longer lifetime of the cluster heads, longer membership time of the cluster members, and better algorithm scalability than the generalized lowest-ID algorithm, at the cost of slightly smaller cluster size. We have also shown that MEACA is an optimal algorithm in terms of the cluster stability and a near-optimal algorithm in terms of the cluster size. In the future work, we will investigate the MEACA algorithm further to evaluate the benefits of stabilizing clusters in the end-to-end communication performance, such as the packet throughput and the packet delay.

## References

[1] C.E. Perkins and P. Bhagwat. "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers". In *ACM SIGCOMM Computer Communications Review*, volume 24, pages 234–244, October 1994.

[2] T. Clausen and P. Jacquet. "Optimized Link State Routing Protocol (OLSR)". In *IETF RFC 3626*, October 2003.

[3] C.E. Perkins and E.M. Royer. "Ad-hoc On-Demand Distance Vector Routing". In *Proc. of IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[4] D.B. Johnson and D.A. Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks". In *Mobile Computing*, pages 153–181, 1996.

[5] P. Krishna, N.H. Vaidya, M. Chatterjee, and D.K. Pradhan. "A Cluster-Based Approach for Routing in Dynamic Networks". *ACM Computer Communication Review*, pages 49–64, April 1997.

[6] M. Gerla and J.T-C. Tsai. "Multicluster, Mobile, Multimedia Radio Network". *Wireless Networks*, pages 255–265, 1995.

[7] C.R. Lin and M. Gerla. "Adaptive Clustering for Mobile Wireless Networks". *IEEE Journal on Selected Areas in Communications (JSAC)*, 15(7):1265–1275, September 1997.

[8] S. Basagni. "Distributed Clustering for Ad Hoc Networks". In *Proc. of ISPAN, International Symposium on Parallel Architectures, Algorithms, and Networks*, pages 310–315, June 1999.

[9] S. Basagni. "Distributed and Mobility-Adaptive Clustering for Multimedia Support in Multi-Hop Wireless Networks". In *Proc. of VTC, IEEE Vehicular Technology Conference*, pages 889–893, September 1999.

[10] R. Ghosh and S. Basagni. "Limiting the Impact of Mobility on Ad Hoc Clustering". In *Proc. of the International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 197–204, October 2005.

[11] B. An and S. Papavassiliou. "A Mobility-Based Clustering Approach to Support Mobility Management and Multicast Routing in Mobile Ad-Hoc Wireless Networks". *International Journal of Network Management*, pages 387–395, 2001.

[12] M. Chatterjee, S.K. Das, and D. Turgut. "WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks". *Cluster Computing*, volumn 5, pages 193–204, 2002.

[13] A.B. McDonald and T.F. Znati. "A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks". *IEEE Jornal on Selected Areas in Communications*, 17(8):1466–1487, August 1999.

[14] O. Younis and S. Fahmy. "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks". *IEEE Transactions on Mobile Computing*, 3(4):366–379, October-December 2004.

[15] C.F. Hsin and M. Liu. "Partial Clustering: Maintaining Connectivity in a Low Duty-Cycled Dense Wireless Sensor Network". In *Proc. of IPDPS, the IEEE International Parallel and Distributed Processing Symposium*, pages 1–8, 2005.

[16] K. Xu, X. Hong, and M. Gerla. "Landmark Routing in Ad Hoc Networks with Mobile Backbones". *Journal of Parallel and Distributed Computing*, 63(2):110–122, February 2003.

[17] T.C. Hou and T.J. Tsai. "An Access-Based Clustering Protocol for Multihop Wireless Ad Hoc Networks". *IEEE Journal on Selected Areas in Communications (JSAC)*, 19(7):1201–1210, July 2001.

[18] L. Ramachandran, M. Kapoor, A. Sarkar, and A. Aggarwal. "Clustering Algorithms for Wireless Ad Hoc Networks". In *Proc. of ACM DIALM Workshop*, pages 54–63, 2000.

[19] R. Ramanathan and M. Steenstrup. "Hierarchically-Organized Multihop Mobile Wireless Networks for Quality-of-Service Support". *Mobile Networks and Applications*, pages 101–119, 1998.

[20] The Network Simulator NS-2. http://www.isi.edu/nsnam/ns/.

[21] J. Yoon, M. Liu, and B. Noble. "Random Waypoint Considered Harmful". *Proc. of IEEE INFOCOM*, March 2003.