A Mobility-Resistant Efficient Clustering Approach for Ad Hoc and Sensor Networks

Jason H. Li[a], Miao Yu[b], Renato Levy[a], Anna Teittinen[a]
jli@i-a-i.com, mmyu@glue.umd.edu, rlevy@i-a-i.com, annat@i-a-i.com
[a]Intelligent Automation, Inc., Rockville, MD, 20855, USA
[b]Department of Mechanical Engineering, University of Maryland, College Park, MD 20742, USA.

This paper presents a Mobility-Resistant, Efficient Clustering Approach (MRECA) for ad hoc and sensor networks. MRECA can provide robustness against moderate node mobility and at the same time render energy-efficiency. The identified clusterheads cover the whole network and each node in the network can determine its cluster and only one cluster. The algorithm terminates in deterministic time without iterations, and each node transmits only one message during the algorithm. We prove analytically the correctness and complexity of the algorithm, and simulation results demonstrate that MRECA is energy-efficient, resilient against node mobility, and robust against synchronization errors[*]

I. Introduction

Advances in wireless technology and portable computing along with demands for greater user mobility have provided a major impetus toward development of self-organizing wireless multi-hop networks, referred to as *ad hoc networks,* composed of a possibly very large number of nodes. These nodes can be either static or mobile, and are usually constrained for the most critical resources, such as power and computation capabilities. An ad hoc network is comprised of wireless nodes and requires no fixed infrastructure.

Communication between arbitrary endpoints in an ad hoc network typically requires routing over multiple-hop wireless paths due to the limited wireless transmission range. Without a fixed infrastructure, these paths consist of wireless links whose endpoints are likely to be moving independently of one another. Consequently, mobile end systems in an ad hoc network are expected to act cooperatively to route traffic and adapt the network to the dynamic state of its links and its mobility patterns. Unlike fixed infrastructure networks where link failures are comparatively rare events, the rate of link failure due to node mobility is the primary obstacle to routing in ad hoc networks [15].

A closely related area of ad hoc networks is *wireless sensor networks (WSNs)* [1], which comprise of a higher number of nodes scattered over some region. Sensor nodes are typically less mobile, and more densely deployed than mobile ad hoc networks. Sensor nodes are usually heavily resource-constrained (especially on power), irreplaceable, and become unusable after failure or energy depletion. Thus it is crucial to devise novel energy-efficient solutions for topology organization and routing that are scalable, efficient and energy conserving in order to increase the overall network longevity.

Given the potentially large number of mobile devices, scalability becomes a critical issue. Among the solutions proposed for scaling down networks with a large number of nodes, network clustering is among the most investigated. The basic idea is to group network nodes that are in physical proximity and thereby logically organize the network into groups with smaller sizes, and hence simpler to manage.

Clustering protocols have been investigated for ad hoc and sensor networks in the literature [2][3][12][14][20][21]. While these strategies differ in the criteria used to organize the clusters, clustering decisions in each of these schemes are based on static views of the network at the time of each topology change; none of the proposed schemes, even equipped with some local maintenance schemes, is satisfactorily resistant to node mobility beyond rare and trivial node movement. However, node mobility is of great importance in ad hoc sensor networks. One motivating example

[*]A previous paper with preliminary results [13] has appeared in the Proceeding of the International Conference on Mobile Ad Hoc and Sensor Networks (MSN05). This paper greatly extends [13] with respects to various aspects, including local maintenance, performance evaluation under various node transmission ranges and node speeds, and synchronization error resilience. This submission presents the complete paper.

is related to battlefield surveillance, where sensor nodes can move and organize among themselves (while moving) to form ad hoc networks. Similar scenarios also exist in disaster relief and search-and-rescue applications. One of the objectives of this work is to propose a clustering protocol that is resilient against mild to moderate mobility where each node can potentially move.

In [22], a hybrid energy-efficient distributed (HEED) clustering approach is presented for ad hoc sensor networks, and the key ideas can be summarized as two-folds. First, clusterhead selection is primarily based on the residual energy of each node. Second, the clustering process entails a number of rounds of iterations; each iteration exploits some probabilistic methods for nodes to elect to become a clusterhead. HEED is a fully distributed protocol and it ensures that each node can either elect to become a clusterhead or it joins a cluster within its range. Further, it has been shown that HEED outperforms generic clustering protocols on various aspects. While HEED is one of the most recognized energy-efficient clustering protocols, it is pointed out that the clustering performance can be further enhanced. In this work, we present a distributed, Moblity-Resistant and Energy-efficient Clustering Approach (MRECA) that has better clustering efficiency. The protocol terminates without rounds of iterations as required by HEED, which makes MRECA a less complex and more efficient algorithm. Further, MRECA's efforts of minimizing control overhead render even smaller overhead than HEED, which enables better energy-efficiency.

The remainder of this paper is organized as follows. Section II describes the network model and states the problem that is addressed in this work. Section III presents the MRECA protocol with correctness and complexity analysis. Performance evaluation is presented in Section IV, followed by the descriptions on relevant work in Section V. We conclude the paper in Section VI.

II. Problem Statement

An ad hoc wireless network is modeled as a set V of nodes that are interconnected by a set E of full-duplex directed communication links. V and E are changing over time when nodes move, join, and leave. Two nodes are neighbors and have a link between them if they are in the transmission range of each other [6]. Neighboring nodes share the same wireless media, and each message is transmitted by a local broadcast.

Nodes within an ad hoc network may move at any time without notice, but we assume that the node speed is moderate with respect to the packet transmission latency and wireless transmission range of the particular underlying network hardware in use. Nodes may join, leave, and rejoin an ad hoc network at any time and any location; existing links may disappear, and new links may be formed as the nodes move. Note that we do not require the ad hoc networks to be either static or quasi-stationary; any node in the network can move independently with some mild to moderate speed. It is our goal that the clustering protocol can still generate decent clusters under such mobility.

Let the clustering duration T_C be the time interval taken by the clustering protocol to cluster the network. Let the network operation interval T_O be the time needed to execute the intended tasks. In many applications, $T_O >> T_C$, which implies that the formed clusters need to be maintained during the operation period in order to reap the advantage of clustering. In general, nodes that travel rapidly in the network may degrade the cluster quality because they alter the node distribution in their clusters and make the clusters unstable, possibly long before the end of T_O. However, research efforts on clustering should not be restricted only within the arena of static or quasi-stationary networks where node movements are rare and slow—some local maintenance mechanisms suffice to tackle such problems [12][14][20]. Rather, for those applications where T_O is not much longer than T_C, we propose in this work an efficient protocol that generates clusters in *ad hoc networks* with mild to moderate node mobility.

In our model for *sensor networks*, though, the sensor nodes are assumed to be quasi-stationary and all nodes have similar capabilities. Nodes are location unaware and will be left unattended after deployment. Recharging is assumed not possible and therefore, energy-efficient sensor network protocols are required for energy conservation and prolonging network lifetime. For clustering, in particular, every node can act as both a source and a server (clusterhead), and a node only knows about the servers that are within its reachable range. A node may fail if its energy resource is depleted, which motivates the need for rotating the clusterhead role in some fair manner among

all neighboring nodes for load balancing.

The problem of clustering is then defined as follows. For an ad hoc or sensor network with nodes set V, the goal is to identify a set of clusterheads that cover the whole network. Each and every node v in set V must be mapped into exactly one cluster, and each ordinary node in the cluster must be able to directly communicate to its clusterhead. The clustering protocol must be completely distributed meaning that each node independently makes its decisions based only on local information. Further, the clustering must terminate fast and execute efficiently in terms of processing complexity and message exchange. Finally, the clustering algorithm must be resistant to moderate mobility (in ad hoc networks) and at the same time renders energy-efficiency, especially for sensor networks.

III. MRECA Clustering Algorithm

The MRECA algorithm structure is somewhat similar to that presented by Lin and Gerla [14] and HEED protocol [22] in that each node broadcasts its decision as the clusterhead in the neighborhood based on some local information and score function. The difference between MRECA and the protocols in [14] and [22] lies in when and how the nodes make such decisions and how the score gets computed. In [14] the score is computed based on node identifiers, and each node holds its message transmission until all its neighbors with better scores (lower ID) have done so. Each node stops its execution of the protocol if it knows that every node in its closed neighborhood (including itself) has transmitted. HEED utilizes node residual energy as the first criterion and takes a cost function as the secondary criterion to compute the score, and each node probabilistically propagates tentative or final clusterhead announcements depending on its probability and connectivity. The execution of the protocol at each node will terminate when the probability of self-election, which gets doubled in every iteration, reaches 1.

It is assumed in [14] that the network topology does not change during the algorithm execution, and therefore it is valid for each node to wait until it overhears the transmission from every higher-scored neighbor. With some node mobility, however, this algorithm can halt since it is possible that an initial neighboring node leaves the transmission range for a node, say v, so that v cannot overhear its transmission. v then has to wait endlessly according to the stopping rule.

Similar assumption exists in HEED so that each node can experience rounds of iterations of tentative or final clusterhead announcements before entering the finalizing phase to choose its cluster. However, it is observed that the rounds of iterations are not necessary and can potentially harm the clustering performance due to the possibly excessive number of transmitted announcements.

But [14] does provide important insights on how the distributed clustering should be performed among neighboring nodes: those nodes with better scores should announce themselves earlier than those with worse scores. We adopt this idea in MRECA and we utilize a score function that captures node residual energy, connectivity and identifier. Each node does not need to hold its announcement until its better-scored neighbors have done so; each node simply calculates a normalized delay based on its score and transmits according to the computed delay. Each node does not need to overhear every neighbor in order to stop; rather, each node can terminate its execution in a pre-determined time, estimated based on its computing capability and node mobility. Further, each node only transmits one message, rather than going through rounds of iterations of probabilistic message announcements. Given the common belief and the fact that it is communication that consumes far more energy in sensor nodes compared with sensing and computation, such savings on message transmissions lead to better energy efficiency.

III.A. MRECA Operation

Each node periodically transmits a Hello message to identify itself, and based on such Hello messages, each node maintains a neighbor list. Define the score function at each node as score $= w_1 E + w_2 C + w_3 I$, where E stands for the node residual energy, C stands for the node connectivity, I stands for the node identifier (for tie breaking), and weights follow $\sum_{i=1}^{3} w_i = 1$. Values of different factors (i.e. E, C and I) will be mapped to some normalized values first, and the weights can be assigned according to the relative importance of each factor—we put higher weight on node residual energy in our work. The computed score is then used to compute the delay for a node to announce itself as the clusterhead. The higher the score, the sooner the node will transmit. The

computed delay is normalized between 0 and a certain upper bound $D_{\max}$, which is a key parameter that needs to be carefully selected in practice, such as the DIFS parameter in IEEE 802.11 MAC protocol. After the clustering starts, the procedure will terminate after time T_{stop}, which is another key parameter. The selection of parameters $D_{\max}$ and T_{stop} needs to consider various issues including node density, node computation capability, mobility and synchronization drifts. In our work, we have not computed such parameters analytically; the trial-and-simulation approach has been used. In this regard, the default values of the DIFS parameter in IEEE 802.11 MAC protocol have been used as references. We have tested $D_{\max} = 10 - 50\,\text{ms}$ and $T_{\text{stop}} = 1 - 2\,\text{s}$, and the protocol works well.

The distributed clustering algorithm at each node is illustrated in the pseudo code fragments. Essentially, clustering is done periodically and at each clustering epoch, each node either immediately announces itself as a potential clusterhead or it holds (schedules) for some delay time.

Upon receiving clustering messages, a node needs to check whether the node ID and the cluster ID embedded in the received message are the same; same node and cluster ID means that the message has been transmitted from a clusterhead. Further, if the receiving node does not belong to any cluster, and the received score is better than its own, the node can mark down the advertised cluster and wait until its scheduled announcement to send its message.

If the receiving node currently belongs to some cluster, and the received score is better than its own score, two cases are further considered. First, if the current node receiving a better-scored message is not a clusterhead itself, as an ordinary node, it can immediately mark down the best cluster so far (line 8 in II) and wait until its scheduled announcement. This node will stay in its committed cluster after its announcement. On the other hand, if the current node is a clusterhead itself, receiving a better scored message (due to variant delays and/or synchronization drifts) means that this node may need to switch to the better cluster. However, cautions need to be taken here before switching since the current node, as a clusterhead, may already have other nodes affiliated with it. Therefore, inconsistencies can occur if it rushes to switch to another cluster. In our approach, we simply mark the necessity for switching (line 7 in II) and defer it to the finalizing phase, where it checks to make sure that no other nodes are affiliated with this node in the cluster as the head, before switching can occur. It is noted that the switch process mandates that a node needs to leave a cluster first before joining a new cluster. Further, it is important to point out that since each node announces itself according to the computed score, this second case is really the exception, rather than the normal case. For example, lower scored nodes may transmit earlier when synchronization drifts among nodes are large. We include such exception handling in MRECA to achieve better robustness. In this (rare) case, the conversion procedure incurs one more message transmission for the converted node. In normal operations, however, each node transmits only one message.

In the finalizing phase, where each node is forced to enter after T_{stop}, each node checks to see if it needs to convert. Further, each node checks if it already belongs to a cluster and will initiate a new cluster with itself as the head if not so.

```
I. START-CLUSTERING-ALGORITHM()
1  myScore = w1E + w2C + w3I;
2  delay = (1000 − myScore)/100;
3  if (delay < 0)
4      then bcastClstr (myId, myCid, myScore);
5      else  delayAnnouncement ();
6  Schedule clustering termination.

II. RECEIVE-CLUSTERING-MESSAGE(id, cid, score)
1  if (id == cid)
2      then if (myCid == NULL)
3               then if (score > myScore)
4                        myCid = cid;
5                        elseif (score > myScore)
6                           then if (myId == myCid)
7                                    needConvert = true;
8                           else  markBestCluster();

III. ACTUAL-ANNOUNCEMENT()
1  bcastClstr (myId, myCid, score);

IV. FINALIZE-CLUSTERING-ALGORITHM()
1  if (needConvert)
2      then if (!amIHeadforAnyOtherNode ())
3               then convtToNewClst ();
4  if (myCid == NULL)
5      then myCid = cid;
6           bcastClstr (myId, myCid, score);
```

III.B. Correctness and Complexity

The protocol described above is completely distributed, and to prove the correctness of the algorithm, we need to show that 1) the algorithm terminates; 2) every node eventually determines its cluster; and 3) in a cluster, any two nodes are at most two-hops away.

Theorem III.1 *Eventually MRECA protocol terminates.*

Proof. After the clustering starts, the procedure will stop receiving messages after time T_{stop}, and enter the finalizing phase. Subsequently, the algorithm will terminate.

In MRECA, a node does not need to wait (possibly in vain as in [14]) to transmit or terminate, nor does it need to go through rounds of probabilistic announcements as in HEED.

Theorem III.2 *At the end of Phase IV, every node can determine its cluster and only one cluster.*

Proof. Suppose a node does not determine its cluster when entering Phase IV. Then the condition at line 4 holds and the node will create a new cluster and claims itself as the clusterhead. So every node can determine its cluster. Now we show that every node selects only one cluster. A node determines its cluster by one of the following three methods. First, it claims itself as the clusterhead; second, it joins a cluster with a better score when its cluster is undecided; and third, it converts from one cluster to another. The first two methods do not make a node join more than one cluster, and the switch procedure checks for consistency and mandates that a non-responsible node (a node not serving as the head for a cluster) can only leave the previous cluster first before joining the new cluster. As a result, no node can appear in two clusters.

One may argue that Theorem III.2 does not suffice for clustering purposes. For example, one can easily invent an algorithm such that every node creates a new cluster and claims itself as the clusterhead; obviously Theorem 2 holds. However, our algorithm does much better than such trivial clustering. Most of the clusters in our algorithm are formed when executing line 4 or line 8 in Phase II, which means joining clusters with better-scored heads. This is due to the fact that the initial order of clusterhead announcements is determined using the insight that *better-scored nodes should announce earlier.*

Theorem III.3 *When clustering finishes, any two nodes in a cluster are at most two-hops away from each other.*

Proof. The proof is based on the mechanisms by which a node joins a cluster. A node, say v, joins a cluster with head w only if v can receive an announcement from w with a better score. In other words, all ordinary nodes are within one-hop from the clusterhead and the theorem follows.

Now we analyze why the algorithm is more robust under node mobility. The key idea that can render such resilience is to abandon the wait-and-send mechanism in [14]; instead, *one* round announcement is scheduled at each node based on scores and the algorithm termination is forced. With carefully selected parameters, the algorithm can terminate much faster and be able to adapt to moderate node mobility. We will verify such resilience in Section IV. To show that the algorithm is energy-efficient, we prove that the communication and time complexity is low.

Theorem III.4 *In MRECA, each node transmits only one message during the operation.*

Proof. In the method that broadcasts cluster information (`bcastClstr` in the Pseudo code), a Boolean variable `iAlreadySent` (not shown in the Pseudo code) ensures that each node cannot send more than once. Now we show that each node will eventually transmit. In Phase I execution when nodes start the clustering, each node either transmits immediately or schedules a delayed transmission, which will get executed at line 1 in Phase III. So each node will eventually transmit.

Theorem III.5 *The time complexity of the algorithm is $O(|V|)$.*

Proof. From Phase II operations, each received message is processed by a fixed number of computation steps without any loop. By Theorem III.4, each node only sends one message and therefore there are only $|V|$ messages in the system. Thus the time complexity is $O(|V|)$.

III.C. Local Maintenance

MRECA clustering is periodically triggered in order to distribute energy consumption among network nodes and adapt to the mobile nature of

ad hoc networks. However, between clustering periods, if most nodes are static and only very few nodes can move slowly, re-clustering may be too expensive. We propose the local maintenance scheme to handle such rare and slow movements.

Essentially, two kinds of events can be detected with such movement: link up and link down. Link up events can be detected when new Hello messages are received from a new node, and link down events can be detected when the expected Hello messages are missing. Such events can be handled according to the roles of the detecting nodes.

For link up event, if the detecting node is not a clusterhead, then it does not need to handle this event. But if the detecting node is itself a clusterhead, it will invite the other node to join its cluster by sending an invitation message (via broadcast) with necessary cluster information. After receiving the invitation, if the node is a clusterhead itself, it will ignore the message; otherwise, it will decide whether or not to join the inviting cluster depending on its current affiliation. If the receiving node does not belong to any other cluster, e.g. a newly powered-up node, it can simply join the cluster; otherwise, it needs to check if it can still contact its current clusterhead. If yes, no need to switch; otherwise, it should join the inviting cluster. The following pseudo codes illustrate such ideas.

```
LINK-UP-DETECT(DetectingNode D)
1  if (D is clusterHead)
2      then inviteToMyCluster(clusterInfo);
3      else do nothing.

LINK-UP-RECEIVING-INVITATION(clusterInfo)
1  if (thisNode != clusterHead)
2      then if (thisNode.clstName==NULL)
3                then acceptInvitation ();
4           elseif (checkPriorClstHead()==OK)
5                then ignoreInvitation ();
6                else acceptInvitation ();
7      else ignoreInvitation();
```

For link down event, the behaviors of the detecting nodes again depend on their roles in the cluster, as shown in the following pseudo codes. For a clusterhead, it will check if the other node resides in the same cluster with itself: if yes, the other node cannot be a clusterhead and it can simply removes the other node from the cluster; otherwise, it does not need to do anything. But if the detecting node is just an ordinary node, it does not need to do anything special to tackle the link down event, given that the algorithm generates only 2-hop clusters. Note that if we aim to generate k-hop clusters, with k > 2, then such receiving node may need to propagate such link down messages so that other nodes can be informed, and the clusterhead can check if the k-hop condition in the cluster still holds.

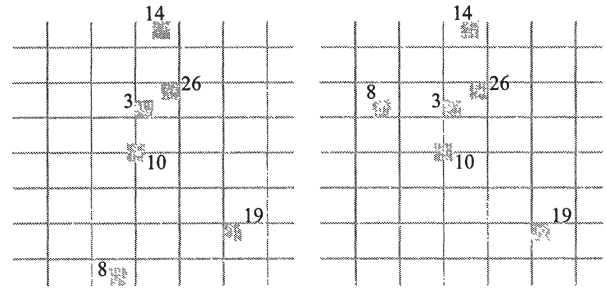

Figure 1: Local maintenance example.

```
LDOWN-DETECT(DetectingNd D, otherNd O)
1  if (D is clusterHead)
2      then if (nodeInMyCluster(O))
3               DRemovesNodeFromCluster(O);
4      else do nothing.
```

Fig.1 shows a simple example, where initially node 8 does not belongs to the top cluster (it is in its own cluster), and as node 8 moves into the top region, it gets invited into the top cluster.

Note, however, that the described local maintenance procedures do not abide by the algorithm shown in Section III.A and may produce a poor quality of cluster structure after repeated use, in which case re-clustering will execute the MRECA algorithm and reconstruct quality clusters.

IV. Performance Evaluation

In this section, we evaluate the MRECA protocol via simulations. We use an in-house simulation tool called agent-based ad hoc network simulator (NetSim) to implement our protocol and the protocols proposed by Krishna et al. [12], Lin and Gerla [14], and HEED [22] for comparisons. Compared with other network simulators (for instance ns-2), the most important feature of NetSim is its capability of handling massive ad hoc wireless networks and sensor networks. In our work, we use network allocation vector (NAV) based protocol (i.e. IEEE 802.11) for medium access control.

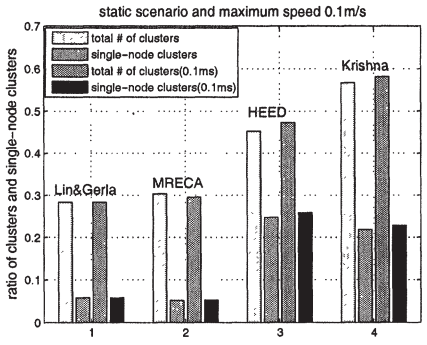

Figure 2: Ratio of number of clusters.

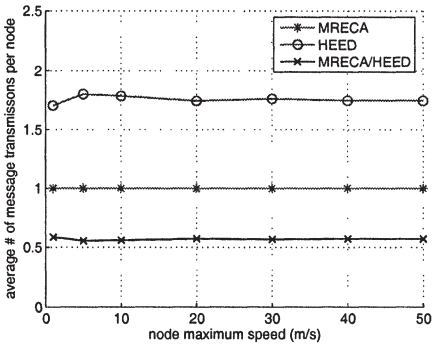

Figure 3: Average number of transmissions per node.

In general, for any clustering protocol, it is undesirable to create single-node clusters. Single-node clusters arise when a node is forced to represent itself (because of not receiving any clusterhead messages). A cluster may also contain a single node if this node decides to act as a clusterhead and all its neighbors register themselves with other clusterheads. While other protocols will generate lots of single-node clusters as node mobility gets more aggressive, our algorithm shows much better resilience under such situations.

In our simulations, random graphs are generated so that nodes are randomly dispersed in a 1000m $\times$ 1000m region and the transmission range of each node is bound to 250m. We investigate the clustering performance under different node mobility patterns, and the node speed ranges from 0 to 50 m/s. In particular, we simulate the following scenarios with maximum node speed set as 0, 0.1, 1, 5, 10, 20, 30, 40, and 50 m/s. For each scenario, each node takes the same maximum speed and a large number of random graphs are generated. Simulations are run and results are averaged over these random graphs.

We have considered the following metrics for performance comparisons: 1) the average overhead (in number of protocol messages); 2) the ratio of the number of clusters to the number of nodes in the network; 3) the ratio of the single-node clusters to the number of nodes in the network; and 4) the average residual energy of the selected clusterheads.

We first look at static scenarios where nodes do not move and the quasi-stationary scenarios where the maximum node speed is bounded at 0.1m/s. We compare four protocols with respect to the ratio of the number of clusters, and the number of single-node clusters, to the number of nodes in the network. We choose [14] proposed by Lin & Gerla (referred to as LIN) as a representative for those general clustering protocols [2][3], and choose Krishna's algorithm (KRISHNA) [12] to represent dominating-set based clustering protocols [16]-[21]. For energy-aware protocols, we choose HEED [22] to compare with MRECA.

Fig. 2 shows that KRISHNA has the worst clustering performance with the highest cluster-to-nodes ratio, while MRECA and LIN possess the best performance. HEED performs in between. In addition, all four protocols perform consistently under (very) mild node mobility.

During our simulations, both LIN and KRISHNA fail to generate clusters as we increase the maximum node speed. This is expected. In LIN, a node will not transmit its message until all its better-scored neighbors have done so; the algorithm will not terminate if a node does not receive a message from each of its neighbors. Node mobility can make the holding node wait forever. In KRISHNA, in order to compute clusters, each node needs accurate information of the entire network topology, facilitated by network-wide link state update which by itself is extremely vulnerable to node mobility. In contrast, we found that both HEED and MRECA are quite resilient to node mobility in that they can generate decent clusters even when each node can potentially move independently of others. The following figures compare the performance of MRECA and HEED under different node mobility.

Fig. 3 shows that for MRECA, the number of protocol messages for clustering remains *one* per node, regardless of the node speed, as proven in Theorem III.4. For HEED, the number of protocol messages is roughly 1.8 for every node speed, and a node running MRECA transmits about 56% number of messages as that in HEED (shown as MRECA/HEED in Fig. 3). The fact

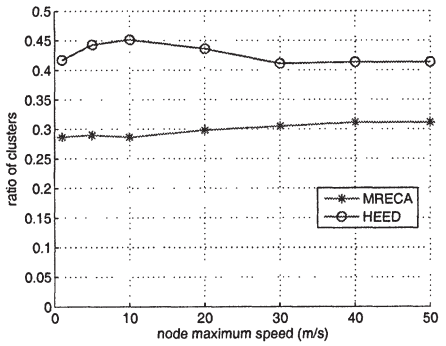

Figure 4: Ratio of clusters to total number of nodes.

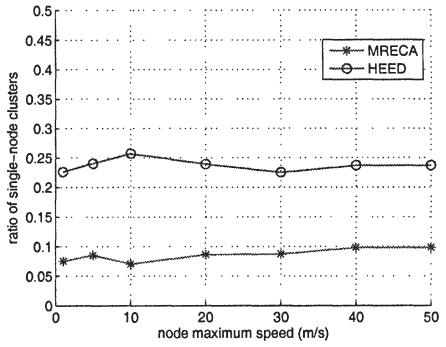

Figure 5: Single-node clusters/total number of nodes.

that HEED incurs more message transmissions is due to the possibly many rounds of iterations (especially when node power is getting reduced), where each node in every iteration can potentially send a message to claim itself as the candidate clusterhead [22]. Reducing the number of transmissions is of great importance, especially in sensor networks, since it would render better energy efficiency and fewer packet collisions (e.g. CSMA/CA type MAC in IEEE 802.11). Fig. 4 and Fig. 5 illustrate the ratio of the number of clusters and single node clusters to the total number of nodes in the network. In both cases, MRECA outperforms HEED.

Note that both MRECA and HEED perform quite *consistently* under different maximum node speed and this is not coincident: a node in both MRECA and HEED will stop trying to claim itself as the potential clusterhead after some initial period (delayed announcement in MRECA and rounds of iterations in HEED) and enter the finalizing phase. As a result, the local information gathered, which serves as the base for clustering, is essentially what can be gathered within the (roughly invariant) initial period which leads to consistent behaviors under different node mobility. It is this consistency in performance that we conclude that both MRECA and HEED are resilient to node mobility.

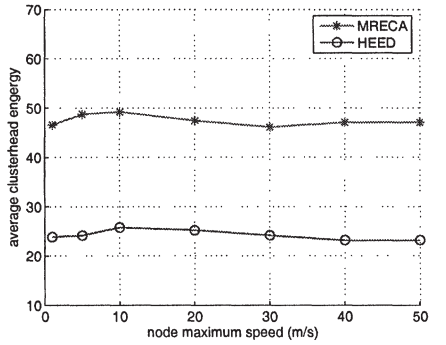

Figure 6: Average clusterhead energy.

Further, we compare MRECA and HEED with respect to the (normalized) average clusterhead energy in Fig. 6. Again both MRECA and HEED perform quite consistently and MRECA outperforms HEED with about doubled the clusterhead residual energy. This is in accordance with Fig. 5 where MRECA consistently incurs fewer message transmissions than HEED. Note that the comparisons here are only informative than quantitative without detailed modeling of lower layer protocols and power expenses. However, the key insights are certainly conveyed. Particularly, in sensor networks, sending fewer messages by each node in MRECA while achieving the intended goal means energy-efficiency and longer node lifetime, since transmission typically consumes orders of magnitude more energy than processing does.

In addition, HEED may possess an undesirable feature in its protocol operation over time. Suppose no re-charging exists in a sensor network. Over time, the energy of each node fades. The decrease of residual energy leads to a uniformly smaller probability of transmission in HEED for each node, which implies more rounds of iterations overall. As a result, more announcements could be sent and more energy could be consumed, which can lead to more messages sent and more energy consumed in the next round of clustering. MRECA, on the contrary, does not possess this potential drawback even with energy fading, since each node only sends *one* message during the operation.

We further extend our simulations to investigate how MRECA performs under different node speeds and transmission ranges. Fig. 7 shows that MRECA performs quite consistently in terms

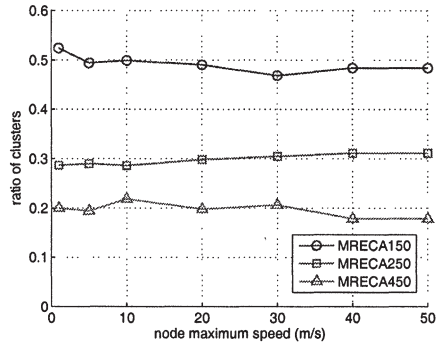

Figure 7: Cluster ratio under different speeds.

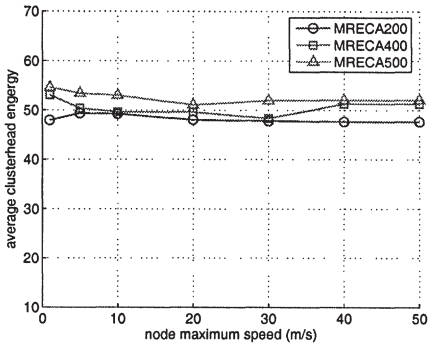

Figure 9: Average energy under different speeds.

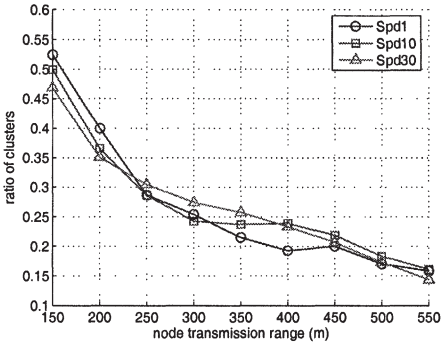

Figure 8: Cluster ratio under different ranges.

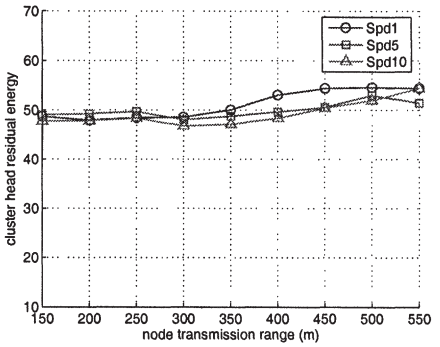

Figure 10: Average energy under different ranges.

of the ratio of clusters under various node speeds; the larger the transmission range, the lower the cluster ratio (as expected). Such observations can also be validated in Fig. 8, where the cluster ratio curves under different node speeds track each other quite closely, and the ratio of clusters decreases as the transmission range increases. Such observations further verify the unique resilience of MRECA against node mobility.

Fig. 9 and Fig. 10 illustrate the average clusterhead (residual) energy under different node speeds and transmission ranges. It can be seen that the clusterhead residual energy are quite consistent under different node speeds (Fig. 9). This is in accordance with the fact that, regardless of node mobility, each node only sends one message in the clustering process. Note that, however, we do not include a more elaborate power consumption model to illustrate the residual energy under different transmission ranges. Using a uniform power model, the residual energy curves track each other fairly closely. Better power models will be incorporated in future work.

In general, it is undesirable to extend a node's transmission range unnecessarily in wireless ad hoc and sensor networks. On the other hand, larger range incurs better clustering performance. To tackle this trade-off, we propose the energy-conservative approach: select the least amount of energy that brings about the largest performance improvement. For example, from Fig. 7 we can tell that the ratio of clusters drops from about 0.5 to about 0.3 with range increases from 150m to 250m, while the ratio drops only to about 0.2 if the range increases further to 450m. As a result, it might not be worthwhile to increase the range over around 250m. This insight can be further validated by Fig. 8, where we should choose the range with the steepest slope in the figure, indicating the greatest improvement on the clustering performance. Again, we need to choose the range around 250m. Such energy-conservative approach is not only of simulation interests; practical deployment of MRECA should follow such insights.

It can be observed that in MRECA the dispersed delay timers for clusterhead announcements assume the existence of a global synchronization system. While this might not be a problem or a constraint for many military communication equipment, synchronization could become trickier for less equipped devices (e.g. commercial PDAs and nodes in the sensor networks).

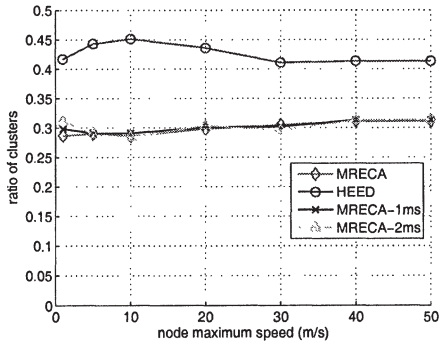

Figure 11: Resilience against sync drifts–cluster ratio.

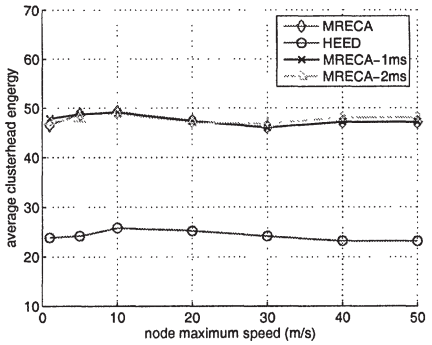

Figure 12: Resilience against sync drifts–energy.

However, we can show that our MRECA scheme is in fact quite resilient against synchronization drifts. It has been shown in the literature [8] that synchronization errors can be controlled within the range of $10\,\mu s$ (with minimum efforts) for nodes in sensor networks, which have largely the most stringent computing and communicating resources. We further relax this time range and put up to $2\,ms$ of errors on the delay timers. Fig. 11 and Fig. 12 illustrate the simulation results. We can easily observe that with $1\,ms$ and $2\,ms$ synchronization errors, the protocol performance tracks the case of a perfect synchronization in an indistinguishable manner.

Finally, all the previous simulation results are obtained assuming perfect wireless channels, i.e. no packet loss will occur. We now investigate the performance of MRECA under lossy wireless channels, e.g. due to irregular radio propagation induced by mobility. We simulate different loss rates and the results are shown in Fig. 13. As can be seen, the protocol performance in terms of cluster ratio only decreases in a very graceful manner, even with 5% channel loss, which is quite severe in the sense of wireless transmission.

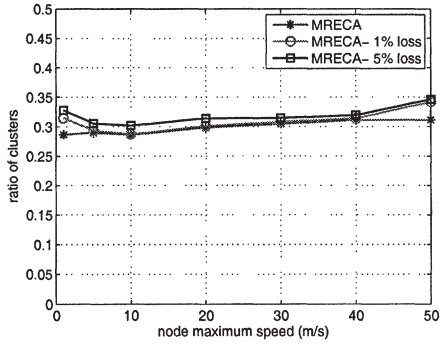

Figure 13: Resilience against losses–cluster ratio.

V. Related Work

Cluster organization has been investigated for ad hoc networks since their appearance. Among scalable routing mechanisms in ad hoc and sensor networks, dominating-set-based clustering [16]-[21] surfaces as one of the most promising approaches. A subset of vertices in an undirected graph is a dominating set if every vertex not in the subset is adjacent to at least one vertex in the subset. Moreover, this dominating set should be connected for ease of the routing process. The main advantage of dominating-set-based routing is that it simplifies the routing process to the one in a smaller subnetwork generated from the connected dominating set (CDS).

Krishna et al. [12] proposed a scheme that organizes the topology into clusters for routing in a dynamic network. A k-cluster is defined to be a subset of nodes which are mutually reachable by a path of length at most k, for some fixed k. A k-cluster with k = 1 is a clique. During the cluster formation, each node needs information of the entire network topology. As a result, for larger networks the amount of information to be updated at each node imposes significant overhead on the communication bandwidth. For mobile ad hoc networks, clique formation usually results in very small clusters, unless the network is really dense.

Sivakumar et al. [16][17][18] proposed a series of 2-level hierarchical routing algorithms for ad hoc wireless networks. The idea is to identify a subnetwork that forms a minimum connected dominating set (MCDS). Each node in the subnetwork is called a spine node and keeps a routing table that captures the topological structure of the whole network. In this approach, a connected dominating set is found by growing a tree T starting from a vertex with the maximum node

degree. Then, a vertex v in T that has the maximum number of neighbors not in T is selected. Finally, a spanning tree is constructed and non-leaf nodes form a connected dominating set.

In [20] and [21] the authors proposed a series of simple and efficient algorithms that can quickly build a backbone directly in ad hoc networks. This approach uses a localized algorithm called the *marking process* where hosts interact with others in restricted vicinity. The resultant dominating set derived from the marking process is further reduced by applying two dominant pruning rules. The low complexity of this algorithm translates into a low communication and computation cost; but the algorithm tends to create very large CDSs.

Basagni [2] proposed to use nodes' weights instead of lowest ID or node degrees in clusterhead decisions. Weight is defined by mobility related parameters, such as speed. Basagni [3] further generalized the scheme by allowing each clusterhead to have at most k neighboring clusterheads and described an algorithm for finding a maximal weighted independent set in wireless networks. Ref. [9] generalized the cluster definition so that a cluster contains all nodes that are at distance at most k hops from the initiator. They proposed to combine connectivity and node ID for choosing the clusterhead. In [4], the authors compared several clustering and backbone formation protocols in large scale ad hoc networks.

Recently, Chan [5] described an emergent clustering algorithm for highly uniform cluster formation (ACE). ACE tries to minimize overlap between clusters, and it claims good performance in three rounds of algorithm iterations. But a better way is to efficiently generate clusters without overlap, rather than reducing it—our MRECA algorithm induces no overlap. Further, MRECA achieves good performance with only one round of iteration.

One of the first protocols that use clustering for network longevity is the Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol [10]. In LEACH, a node elects to become a clusterhead randomly according to a target number of clusterheads in the network and its own residual energy, and energy load get evenly distributed among the sensors in the network. LEACH clustering proved to be 4 to 8 times more effective in prolonging the network lifetime than direct communication or minimum energy transfer. A limitation of this scheme is that it requires all current clusterheads to be able to transmit directly to the sink. Improvements to the basic LEACH algorithms include multi-layer LEACH-based clustering and the optimal determination of the number of clusterheads that minimizes the energy consumption throughout the network.

VI. Conclusion and Future Work

In this paper we present a distributed, efficient clustering algorithm that works with resilience to node mobility and synchronization errors, and at the same time renders energy efficiency. The algorithm terminates fast, has low time complexity, and generates non-overlapping clusters with good clustering performance. Our approach is applicable to both mobile ad hoc networks and energy-constrained sensor networks. The clustering scheme provides a useful service that can be leveraged by different applications to achieve scalability. For example, our approach can be effective for sensor applications requiring efficient data aggregation and prolonged network lifetime, such as environmental monitoring, and efficient data fusion in many applications.

Our future work includes more extensive simulations on larger scale networks with elaborate power models, extensions to k-hop clusters, and integration of clustering with network applications, for example cooperative intrusion detection and multi-tiered secure group communications.

Acknowledgement

The authors are thankful to the reviewers for their valuable suggestions. This work was supported by the Air Force Research Laboratory, USA, grant FA8750-05-C-0161.

References

[1] I. F. Akyildiz, W. Su, Y. Sanakarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," Computer Networks, vol. 38, no. 4, pp. 393-422, 2002.

[2] S. Basagni, "Distributed clustering for ad hoc networks," in Proc. of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks.

[3] S. Basagni, D. Turgut, and S. K. Das, "Mobility-adaptive protocols for managing large ad hoc networks,"

in Proc. of the IEEE International Conference on Communications, 2001.

[4] S. Basagni, M. Mastrogiovanni, and C. Petrioli,"A performance comparion of protocols for clustering and backbone formation in large scale ad hoc networks," in Proc. Int'l conference on Mobile Ad Hoc and Sensor Systems, 2004.

[5] H. Chan and A. Perrig, " ACE: An Emergent Algorithm for Highly Uniform Cluster Formation", in Proc. of the First European Workshop on Sensor Networks(EWSN), January 2004.

[6] B. N. Clark, C. J. Colburn, and D. S. Johnson,"Unit disk graphs," Discrete Mathematics, vol. 86, pp. 165-167, 1990.

[7] F. Dai and J. Wu,"An extended localized algorithms for connected dominating set formation in ad hoc wireless networks," IEEE Trans. on Parallel and Distributed Systems, vol. 15, no. 10, 2004.

[8] J. Elson and L. Girod and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts", In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), December 2002.

[9] F. Garcia Nocetti, J. Solano Gonzales, and I. Stojmenovic, "Connectivity based k-hop clustering in wireless networks," TelecommunicationSystems, vol. 22, no. 1-4, pp. 205-220, 2003.

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan,"Energy efficient communication protocol for wireless microsensor networks," in Proc. of the 3rd Annual Hawaii International Conference on System Sciences, 2000, pp. 3005-3014

[11] U. C. Kozat, G. Kondylis, B. Ryu, and M. K. Marina,"Virtual dynamic backbone for mobile ad hoc networks," in Proc. of the IEEE International Conference on Communications (2001).

[12] P. Krishna, N.N. Vaidya, M. Chatterjee and D.K. Pradhan, "A cluster-based approach for routing indynamic networks", ACM SIGCOMM Computer Communication Review 49 (1997) 49-64.

[13] J. H. Li, M. Yu, and R. Levy, " Distributed Efficient Clustering Approach for Ad Hoc and Sensor Networks" in Proc International Conference on Mobile Ad hoc and Sensor Networks, 2005.

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

[15] A. B. McDonald and T. Znati, "A mobility-based framework for adaptive clustering in wireless ad hoc networks," IEEE Journal on Selected Areas in Communications, vol. 17, no. 8, pp. 1466-1487, August 1999.

[16] R. Sivakumar, B. Das, and V. Bharghavan,"The clade vertebrata: Spines and routing in ad hoc networks," in Proc. of the IEEE Symposium on Computer Communications (ISCC'98).

[17] R. Sivakumar, B. Das, and B. V.,"Spine-based routing in ad hoc networks," ACM/Baltzer Cluster Computing Journal, vol. 1, pp. 237-248, November 1998.

[18] R. Sivakumar, P. Sinha, and V. Bharghavan,"CEDAR: A core-extraction distributed ad hoc routing algorithm," IEEE Journal on Selected Areasin Communications, vol. 17, no. 8, pp. 1454-1465, 1999.

[19] I. Stojmenovic, M. Seddigh, and J. Zunic,"Dominating sets and neighbors elimination-based broadcasting algorithms in wireless networks," IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 1, pp. 14-25, 2002.

[20] J. Wu and H. Li,"On calculating connected dominating sets for efficient routing in ad hoc wireless networks," Telecommunication Systems, vol. 18,no. 1/3, pp. 13-36, September 2001.

[21] J. Wu, F. Dai, M. Gao, and I. Stojmenovic,"On calculating poweraware connected dominating sets for efficient routing in ad hoc wireless networks," Journal of Communications and Networks, vol. 4, no. 1, pp.1-12, March 2002.

[22] O. Younis, S. Fahmy,"HEED: A Hybrid, Energy-Efficient,Distributed Clustering Approach for Ad Hoc Sensor Networks," IEEE Trans. on Mobile Computing, VOL. 3, NO. 4, 2004