Mobility Resistant Clustering in Multi-Hop Wireless Networks

Miao Yu
Department of Mechanical Engineering, University of Maryland, College Park, MD, USA
Email: mmyu@eng.umd.edu

Jason H. Li and Renato Levy
Intelligent Automation Inc., Rockville, MD, USA
Email: {jli, rlevy}@i-a-i.com

***Abstract*—This paper presents a Distributed Efficient Clustering Approach (DECA) for mobility-resistant and energy-efficient clustering in multi-hop wireless networks. The clusterheads cover the whole network and each node in the network can exclusively determine the single cluster it belongs. DECA is fully distributed, each node transmits only one message during clustering operation, and the algorithm terminates in deterministic time without iterations. Theoretical results show the correctness of DECA, and extensive simulation results demonstrate that DECA is energy-efficient and robust against node mobility.**

***Index Terms*—clustering, ad hoc networks, wireless sensor networks, mobility, performance evaluation**

I. INTRODUCTION

Multi-hop ad hoc networks provide a flexible means of communication when there is no infrastructure or the existing infrastructure is inconvenient or expensive to use. Without a fixed infrastructure, routing 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 mobility patterns. Unlike infrastructure networks where link failures are relatively rare events, the rate of link failure due to node mobility is the primary obstacle to routing in ad hoc networks [1].

With the potentially large number of mobile devices in such networks, 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.

Based on "A Distributed Efficient Clustering Approach for Ad Hoc and Sensor Networks", by J. H. Li, M. Yu, and R. Levy, which appeared in the Proceedings of the first International Conference on Mobile Ad Hoc and Sensor Networks, Wuhan, China, December 2005.

Clustering protocols have been investigated extensively for multi-hop wireless networks in the literature [2]-[6]. While these strategies differ in the criteria used to organize the clusters, clustering decisions in these schemes are based on static views of the network; none of these schemes, even equipped with local maintenance remedies, is satisfactorily resistant to node mobility beyond rare and trivial node movements.

Another important category of multi-hop wireless networks is called *wireless sensor networks* [7], which comprise of a larger number of nodes (in thousands and more) scattered over some region. Sensor nodes are typically less mobile, and more densely deployed than ad hoc networks. The sensor nodes gather data from the environment and they can perform various kinds of activities including collaborative processing of sensor data and performing some synchronized actions based on the gathered sensor data. Typically, sensor nodes are heavily resource-constrained (especially on power), irreplaceable, and become unusable after failure or energy depletion. It is thus 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.

Clustering techniques can facilitate in reducing energy consumption. Network lifetime can be prolonged through reducing the number of nodes contending for channel access, summarizing information at the clusterheads, and routing through an overlay among clusterheads, which has a relatively small network diameter. In this article, we present a distributed, efficient clustering approach (DECA) that outperforms the state-of-the-art in terms of energy efficiency and possesses the advantages of better clustering overhead and resilience against node mobility.

The remainder of this paper is organized as follows. Section II describes the related work, followed by the revisit of the DECA protocol in Section III. Performance evaluation is presented in Section IV, and various issues and application scenarios are discussed in Section V. The paper is concluded in Section VI.

II. RELATED WORK

Clustering algorithms have been investigated for ad hoc networks since their appearance. The first solutions [8][9] aimed at partitioning the nodes into clusters, each with a clusterhead and some ordinary nodes, so that the clusterheads form an independent set, i.e., a set whose nodes are never neighbors among themselves. In general, the sizes of the constructed clusters should not be very small so that the efforts for clustering and creating network hierarchies can justify.

Among scalable routing mechanisms in multi-hop wireless networks, dominating set based clustering [10]-[14] serves 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 routing within the induced graph of dominating vertices. The main advantage of dominating set based routing is that it simplifies the routing problem to a smaller subnetwork generated from the connected dominating set.

Using the idea of the connected dominating set (CDS), Krishna et al. [4] proposed to dynamically organize the network topology into clusters for routing. A k-cluster is defined as a subset of nodes that are mutually "reachable" by a path length of at most k for some fixed k. A k-cluster with k = 1 is a *clique*. During the cluster formation, the network is viewed as a dynamically growing system, and each node needs information of the entire network topology. As a result, for "large" networks, the amount of information to be updated at each mobile node is significant, which imposes much overhead on the communication bandwidth. For mobile ad hoc networks, clique formation usually results in small clusters, unless the network is really dense.

Sivakumar et al. [10]-[12][15] 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. The main drawback of this algorithm is that it still needs a non-constant number of rounds to determine a connected dominating set [14].

Wu et al. [14][16] proposed a series of simple and efficient localized 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 pruning rules. The low complexity of such algorithms translates into low communication and computation cost; but the algorithms tend to create large CDSs.

Instead of constructing connected dominating sets, Lin and Gerla [5] used node ID numbers to build clusters of nodes that are reachable by two-hop paths. The distributed clustering algorithm is initiated by all nodes that have the lowest ID numbers among their neighbors. The cluster initiators broadcast their decision to all their neighbors. If all the lower ID neighbors sent their decisions and none declared itself as a cluster initiator, the node decides to create its own cluster and broadcasts its own ID as the cluster ID. Otherwise, it chooses a neighboring cluster with the lowest ID, and broadcasts such decision. Ref. [17] generalizes the cluster definition so that a cluster contains all nodes that are at distance at most k hops from the initiator.

Similar to [5], Basagni [2] proposed to use nodes' weights instead of the lowest ID or node degrees in clusterhead decisions. Weight is defined by mobility related parameters, such as speed. Ref. [3] further generalized the scheme by allowing each clusterhead to have at most k neighboring clusterheads and described an algorithm of finding a maximal weighted independent set in wireless networks.

In [6], a hybrid energy-efficient distributed (HEED) clustering protocol was presented for ad hoc sensor networks. HEED utilizes node residual energy as the first criterion and takes a cost function as the second criterion to compute the score. Each node probabilistically propagates tentative or final clusterhead announcements depending on its probability and connectivity. The clustering process entails a number of rounds of iterations, and the execution of the protocol at each node will terminate when the probability of self-election, which is doubled in every iteration, reaches 1. It has been shown that HEED outperforms generic clustering protocols on various aspects.

One of the first protocols that use clustering for extending network lifetime was the Low-Energy Adaptive Clustering Hierarchy (LEACH) [18]. 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. Energy load gets evenly distributed among the sensors in the network. LEACH clustering proved to be 4 to 8 times more effective in extending the network lifetime than direct communication or minimum energy transfer (shortest path routing). 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.

None of the above algorithms intends to handle the scenarios that all the nodes in the network can potentially move. DECA protocol tackles such problems. The initial idea was proposed in Ref. [19], and a subsequent study focused on clustering performance with lossy wireless channels and synchronization errors [20]. This article enriches DECA protocol with more extensive results and insights under different node transmission ranges.

III. DECA CLUSTERING ALGORITHM

A. 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 communication links. Each node has a unique identifier and has at least one transmitter and one receiver. Two nodes are neighbors and have a link between them if they are in the transmission range of each other [21]. Neighboring nodes share the same wireless media, and each message is transmitted through a local broadcast.

Nodes within an ad hoc network may move at any time without notice, but it is assumed that the node speed is moderate with respect to the packet transmission latency and the transmission range of the particular underlying network hardware. Nodes may join, leave, and rejoin an ad hoc network at any time, existing links may disappear, and new links may be formed as the nodes move.

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 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 [5][14]. Rather, for applications where T_O is not much longer than T_C, it is proposed in this work an efficient protocol that can generate descent clusters under mild to moderate node mobility.

The problem of clustering is then defined as follows. For a multi-hop wireless network with node 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, that is, 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.

B. Overview of DECA

The DECA algorithm structure is somewhat similar to that presented in [5] and the HEED protocol [6] in that each node broadcasts its decision as the clusterhead in the neighborhood based on some local information and score function. The difference between DECA and these two protocols lies in when and how the nodes make such decisions and how the score gets computed.

In [5] the score is computed based on node identifiers, and each node holds its message transmission until all its neighbors with lower IDs have done so. Each node stops its protocol execution if it knows that every node in its neighborhood has transmitted. It is assumed that the network topology does not change during the algorithm execution, and it is thus valid for each node to wait until it overhears every higher-scored neighbor transmitting. With some node mobility, however, this algorithm can halt since it is quite possible that an initial neighboring node leaves the transmission range for a node, say v, so that v cannot overhear its transmission. Node v then has to wait endlessly according to the stopping rule.

HEED also assumes static network topology so that each node can experience rounds of iterations of tentative or final clusterhead announcement before entering the finalizing phase to choose its cluster. Under node mobility, HEED will not halt. However, we observe that the iterations are not necessary and can potentially harm the clustering performance due to the possibly excessive number of announcements during iterations.

But Ref. [5] does provide important insights on how distributed clustering should be performed among neighboring nodes: those nodes with better scores should announce themselves earlier. We adopt this idea in DECA—actually many clustering protocols use ideas similar to this [2][3][9]—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 announcement. Given the fact that it is communication that consumes far more energy in sensor nodes compared with sensing and computation, such savings on message transmission lead to better energy efficiency.

C. DECA Operation

Each node periodically transmits a Hello message to identify itself, and based on such Hello messages, each node maintains a neighbor list. Define for each node the score function 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, and the weights follow $\sum_{i=1}^{3} w_i = 1$. The computed score is then used to compute the delay for this 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, like the DIFS parameter in IEEE 802.11. After the clustering starts, the procedure will terminate after time T_{stop}, another key parameter whose selection needs to take the

node computation capability and mobility into consideration. In the simulation, we choose $D_{\max}$ between $10\text{ ms} - 50\text{ ms}$ and T_{stop} between $1\text{ s} - 2\text{ s}$, and the protocol works well.

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

Upon receiving clustering messages, a node first checks 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 will mark down the advertised cluster and wait until its scheduled time 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. We include such exception handling for better robustness.

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. StartClusteringAlgorithm()
 1  myScore = w1E + w2C + w3I;
 2  delay = (1000 – myScore)/100;
 3  if (delay < 0)
 4        broadcastCluster(myId,myCid,myScore);
 5  else  delayAnnouncement ();
 6  Schedule clustering termination.

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

III. ActualAnnouncement ()
 1  broadcastCluster (myId, myCid, score);

IV. FinalizeClusteringAlgorithm ()
 1  if (needConvert)
 2     if (!amIHeadforAnyOtherNode ())
 3        convertToNewCluster ();
 4  if (myCid == NULL)
 5     myCid = cid;
 6     broadcastCluster (myId, myCid, score);
```

D. Correctness and Complexity

The DECA protocol described above is completely distributed. To show the correctness and efficiency of the algorithm, the following theoretical results have been presented in our previous study [20].

Theorem 1. Eventually DECA terminates.

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

Theorem 3. When clustering finishes, any two nodes in a cluster are at most two-hops away.

Theorem 4. In DECA, each node transmits only one message during the operation.

Theorem 5. The time complexity of DECA is $O(|V|)$.

IV. PERFORMANCE EVALUATION

In this section, we use an in-house simulation tool called agent-based ad-hoc network simulator (NetSim) to implement our protocol and the algorithms proposed by Krishna et al. [4], Lin and Gerla [5], and HEED [6] 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 simulations, random graphs are generated so that nodes are randomly dispersed in a $1000\text{m} \times 1000\text{m}$ region. All nodes have the same transmission radius, which ranges from 150m to 450m with an increment of 50m. To study mobility resistance, the transmission range is set to 250m, and we investigate the clustering performance under different node speed ranges. 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 speed, 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.

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 clustering algorithms typically generate lots of single-node clusters as node mobility gets more aggressive, our algorithm shows much better resilience in such situations.

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 total number of nodes in the network; 3) the ratio of the number of single-node clusters to the total number of nodes in the network; and 4) the average residual energy of the 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 choose Ref. [5] (referred to as LIN) as a representative for those general clustering protocols [2][3], and choose Ref. [4] (KRISHNA) to represent dominating-set based clustering protocols [10]-[14]. For the state-of-the-art, we choose HEED [6] to compare with DECA.

From Fig. 1 it is easy to observe that KRISHNA has the worst clustering performance with respect to cluster ratios, while DECA and LIN have the best performance. HEED performs in between. In addition, all four protocols perform quite consistently under (very) mild node mobility (i.e., 0.1m/s maximum speed).

During our simulations, both LIN and KRISHNA fail to generate clusters as the node speed increases. This is expected. In LIN, 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, which by itself is extremely vulnerable to node mobility. In contrast, we found that both HEED and DECA 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 DECA and HEED under different node mobility.

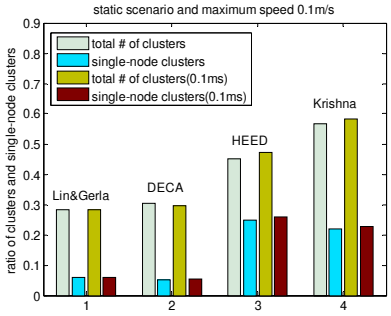


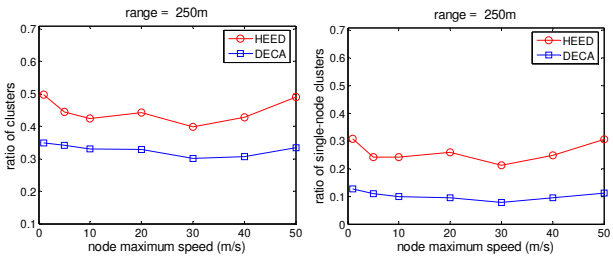
Figure 1. Ratio of clusters for four protocols.

Fig. 2 shows the ratio of the number of clusters and single-node clusters to the total number of nodes in the network. All nodes have the same transmission range of 250m. In both cases, DECA significantly outperforms HEED, with performance gains around 40% in Fig. 2(a) and 200% in Fig. 2(b).

Fig. 3(a) shows that for DECA, the number of protocol messages for clustering remains *one* per node, regardless of the node speed, as proven in Theorem 4. For HEED, the number of protocol messages is roughly 1.7−2 for every node speed. The fact that HEED incurs more message transmissions is due to the possibly many rounds of iterations, where each node in every iteration can potentially send a message to claim itself as the candidate clusterhead [6].

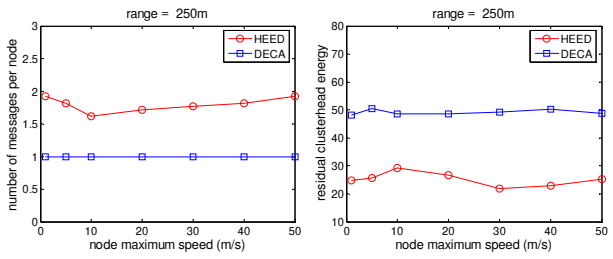
Fig. 3(b) compares DECA and HEED with respect to the (normalized) average clusterhead energy. Again, DECA outperforms HEED with about twice the clusterhead residual energy. This is in accordance with Fig. 3(a) where DECA consistently incurs fewer message transmissions than HEED. Reducing the number of transmissions is of great importance, especially in sensor networks, since it would render better energy efficiency and fewer packet collisions, e.g. in IEEE 802.11 MAC.

We extend our simulations to investigate how DECA and HEED perform under different node speeds and transmission ranges. Fig. 4(a) shows that DECA performs quite consistently in terms of cluster ratio under various node speeds, and the larger the transmission range, the lower the cluster ratio (as expected). Such observations can also be made in Fig. 4(b), 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. Similar observations have also been made for HEED (figures not shown).



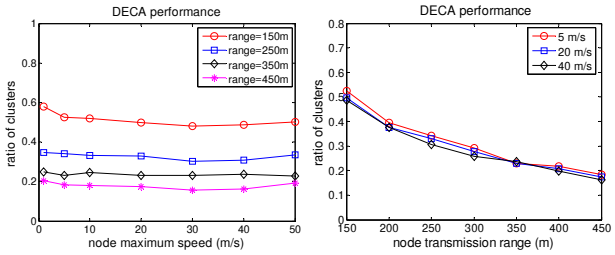
(a) (b)

Figure 2. Ratio of clusters (a), and single-node clusters (b).



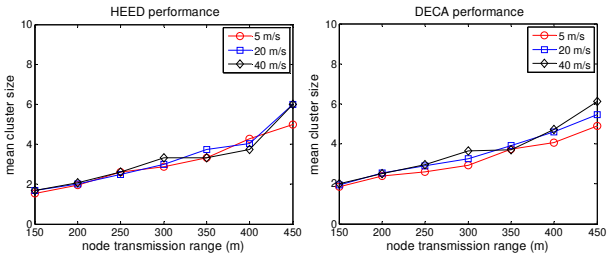
(a) (b)

Figure 3. Number messages (a), and residual clusterhead energy (b).



(a) (b)

Figure 4. Cluster ratio under different (a) speeds, and (b) ranges.



(a) (b)

Figure 5. Cluster ratio under different (a) speeds, and (b) ranges.

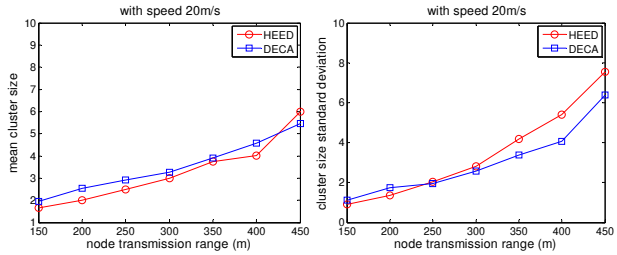
Next, we look at average cluster sizes. Fig. 5 shows that for both HEED and DECA, the mean cluster sizes obtained with different node speeds track each other quite closely. Similar results are also obtained with respect to the standard deviation of cluster sizes for both HEED and DECA (not shown here). These results reveal that both DECA and HEED are resilient against node mobility.

To observe the clustering performance, Fig. 6 shows that DECA essentially incurs similar or a bit better performance compared with HEED with respect to both the mean and the standard deviation of cluster sizes. Further, it is obvious from Fig. 6 that as the transmission range increases, both the mean and the standard deviation increase. The latter is typically not desirable since more uniform clusters are generally preferred to achieve better scalability. Hence, care must be taken to prevent large cluster size variations.

Though Fig. 5 and Fig. 6 may suggest that DECA only performs similarly as HEED, we will show below that, in fact, DECA outperforms HEED significantly for every transmission range used in the simulations. Given the mobility resilience, we pick 10 m/s node speed as a representative scenario. Fig. 7 illustrates the simulation results obtained by comparing DECA and HEED.

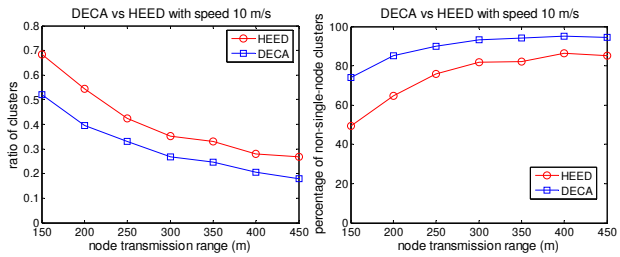
It is obvious in Fig. 7 that DECA consistently incurs smaller ratio of clusters and higher percentage of non-single-node clusters for every transmission range. In addition, it is interesting to observe that the performance gain of DECA over HEED decreases as the transmission range increases. This is particularly evident in Fig. 7(b).

Such phenomenon is somewhat similar to another finding in terms of the per node protocol message. Fig. 8 shows that as the transmission range increases, while DECA incurs only one message per node, the number of messages sent per node in HEED consistently decreases.



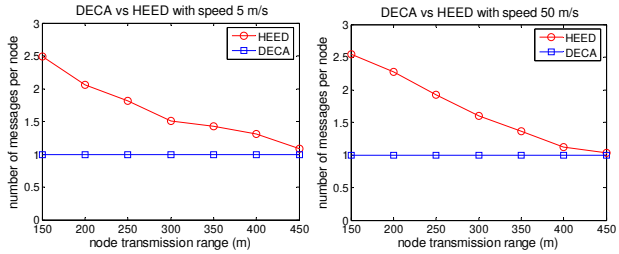
(a) (b)

Figure 6. Cluster size comparisons: (a) mean, (b) standard deviation.



(a) (b)

Figure 7. Ratio of clusters (a), percent of non-single-node clusters (b).



(a) (b)

Figure 8. Number of messages with node speed (a) 5m/s and (b) 50m/s.

This observation is in accordance with how HEED works. In HEED's main processing phase [6], a node sends a clusterhead announcement message when its probability of becoming a clusterhead, CH_{prob}, reaches 1. Further, when the set of tentative cluster heads S_{CH} is empty and $CH_{prob} < 1$, a node can send an announcement message randomly based on its CH_{prob}. Hence, when the transmission range is small, there are fewer neighbors (or no neighbors at all) for each node compared to the case with a larger transmission range. As a result, the tentative clusterhead set S_{CH} perceived by each node is more likely to be empty, and each node then needs to transmit a message according to its CH_{prob}. This situation in turn leads to more messages being sent (and *wasted*) during HEED iterations. As the transmission range gets larger, the set S_{CH} at each node tends to be non-empty, and thus some number of random message transmissions is saved. Thus, the performance gains of DECA over HEED in terms of number of messages sent per node will decrease with larger transmission range.

From Fig. 7 and Fig. 8, one may conclude that larger transmission range is more preferable for better clustering performance. However, it is generally undesirable to extend a node's transmission range in multi-hop wireless networks due to energy and interference issues. To tackle this trade-off, we propose the energy-conservative approach: *select the smallest transmission range that brings about the largest performance improvement.*

For example, the ratio of clusters in Fig. 4(a) drops from about 0.55 to about 0.35 with range increases from 150m to 250m. The ratio drops only to about 0.25 if the range increases further to 350m. As a result, it might not be worthwhile to increase the range over around 250m. This insight can be further validated by Fig. 4(b), 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. Fig. 7(b) also indicates that beyond 250m transmission range, the performance of DECA become "flat" and its gain over HEED gets smaller. This energy-conservative approach is not only of simulation interests, practical deployment of the DECA algorithm should also follow such insights.

V. DISCUSSIONS

Node mobility is of great importance in multi-hop ad hoc networks. One motivating example is related to battlefield surveillance where communication nodes can move and organize among themselves 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.

Our simulation results reveal that both DECA and HEED perform quite consistently under different maximum node speed. This is not coincident: a node in both DECA and HEED will stop trying to claim itself as the potential clusterhead after some initial period (delayed announcement in DECA 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 DECA and HEED are resilient to node mobility.

On the other hand, DECA outperforms HEED in an all-round manner with respect to the common clustering performance measures. In particular, DECA only incurs one protocol message per node, which directly implies better energy efficiency and less wireless interference. This is especially important in wireless sensor networks where sensor nodes are severely resource-constrained. Since recharging is typically not possible, energy-efficient sensor network protocols are required for energy conservation and prolonging network lifetime. With less protocol message overhead, DECA can result in more economic energy consumption than HEED.

In addition, HEED may possess another undesirable feature in its protocol operation. Over time, the energy of each node fades. The decrease of residual energy leads to a uniformly smaller probability of announcement 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 could incur even more messages in the next round of clustering. DECA, on the contrary, does not possess this "positive feedback" even with energy fading, since each node only sends *one* message during the operation.

It can be observed that the dispersed delay timers in DECA assume global synchronization among nodes. While this might not be a problem for some military ad hoc network applications, synchronization can be rather tricky in less-equipped sensor networks. However, we have shown in [20] that DECA is in fact quite resilient to synchronization errors. Further, Ref. [20] also shows that our clustering protocol is fairly robust against wireless packet losses.

The DECA clustering scheme provides a useful service that can be leveraged by many different applications to achieve scalability. For instance, in secure group communication scenarios, the large size of the serving group, combined with the dynamic nature of group changes, pose a significant challenge on the scalability and efficiency on key management research. In a previous work [22], the scalability problem was solved by partitioning the mobile devices into *subgroups* and further organizing the subgroups into hierarchies. Key management and actual data transmissions follow the hierarchy. DECA algorithm was utilized to organize mobile devices into subgroups that result in better scalability and efficiency. Other examples include sensor network applications that require efficient data aggregation and prolonged network lifetime, e.g. environmental monitoring.

VI. CONCLUSIONS

In this paper we present a distributed and efficient clustering algorithm that works with resilience to node mobility and at the same time leads to energy efficiency. The algorithm terminates fast, has low computational complexity, and generates non-overlapping clusters with good clustering performance. Our approach is generally applicable to most multi-hop wireless networks.

Our future work includes more extensive simulations on large-scale wireless networks with elaborate power models, extension to k-hop clustering, and integration with various wireless network applications spanning from efficient sensor network data fusion to cooperative intrusion detection in ad hoc networks.

ACKNOWLEDGMENT

The authors gratefully acknowledge the partial support received from the US Air Force Research Laboratory under the contract NO. FA8750-05-C-0161.

REFERENCES

[1] 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, Special Issue on Wireless Ad Hoc Networks*, vol. 17, no. 8, pp. 1466–1487, August 1999.

[2] S. Basagni, "Distributed clustering for ad hoc networks," in *Proceedings 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 *Proceedings of the IEEE International Conference on Communications, ICC 2001*, Helsinki, pp. 1539–1543.

[4] P. Krishna, N.N. Vaidya, M. Chatterjee and D.K. Pradhan, "A cluster-based approach for routing in dynamic networks", *ACM SIGCOMM Computer Communication Review*, vol. 49, pp. 49–64, 1997.

[5] 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, 1997.

[6] 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

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

[8] D. J. Baker, A. Ephremides, and J. A. Flynn, "The design and simulation of a mobile radio network with distributed control," *IEEE Journal on Selected Areas in Communications*, vol.2, no. 1, pp. 226–237, January 1984.

[9] A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 56–73, January 1987.

[10] R. Sivakumar, B. Das, and V. Bharghavan, "The clade vertebrata: Spines and routing in ad hoc networks," in *Proceedings of the IEEE Symposium on Computer Communications (ISCC'98)*, Athens, Greece, 1998.

[11] 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, special Issue on Mobile Computing.

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

[13] 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, January 2002.

[14] J. Wu and H. Li, "On calculating connected dominating sets for efficient routing in ad hoc wireless networks," *Telecommunication Systems, Special Issue on Mobile Computing and Wireless Networks*, vol. 18, no. 1/3, pp. 13–36, September 2001.

[15] P. Sinha, R. Sivakumar, and V. Bharghavan, "Enhancing ad hoc routing with dynamic virtual infrastructures," in *Proceedings of IEEE Infocom 2001*, vol. 3, Anchorage, AK, April 22-26 2001, pp. 1763–1762.

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

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

[18] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy efficient communication protocol for wireless micro sensor networks," in *Proceedings of the 3rd Annual Hawaii International Conference on System Sciences, HICSS 2000*, Maui, HA, January 4–7 2000, pp. 3005–3014

[19] J. H. Li, M. Yu, and R. Levy, "Distributed Efficient Clustering Approach for Ad Hoc and Sensor Networks" in *Proc. of the First International Conference on Mobile Ad hoc and Sensor Networks*, Wuhan, China, 2005.

[20] J. H. Li, M. Yu, R. Levy, and A. Teittinen, "A Mobility-Resistant Efficient Clustering Approach for Ad Hoc and Sensor Networks", in *Mobile Computing and Communications Review*, 2006, in press.

[21] B. N. Clark, C. J. Colburn, and D. S. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, pp. 65–167, 1990.

[22] J. H. Li, M. Yu, R. Levy, and B. Bhattacharjee, "A Scalable Key Management and Clustering Scheme for Ad Hoc Networks", in *Proc. InfoScale*, Hong Kong, 2006

Miao Yu obtained her B.S and M.S. degrees in the field of engineering mechanics from the Tsinghua University, Beijing, China, in 1996 and 1998 respectively, and her Ph.D. degree in mechanical engineering from the University of Maryland at College Park, USA in 2002.

She has been an assistant professor in the Department of Mechanical Engineering in the University of Maryland at College Park since January 2005. Her research interests include smart sensors, sensor systems for military, civil, mechanical, electrical, biochemical, and environmental applications, collaborative sensor signal processing, and sensor networks.

Dr. Yu is a member of the OSA, SPIE, ASME, ASEE, and SEM. Her awards include the *Invention of the Year* Award from the University of Maryland in 2002.

Jason H. Li received his B.E. and M.S. degrees in Electrical Engineering from the Tsinghua University, Beijing, China in 1993 and 1996 respectively, and his Ph.D. degree in Electrical and Computer Engineering, from the University of Maryland at College Park, USA in 2002 majoring in computer communication networks.

He is currently a senior research scientist at Intelligent Automation, Inc., Rockville, MD, USA. Prior to that, he was a senior research scientist in Hughes Network Systems. His research interests lie in the general area of computer communication networks, including network protocols, network security, network management and control, and distributed software agents.

Dr. Li is a member of the IEEE.

Renato Levy received his BSEE degree in electrical engineering from the Federal University of Rio de Janeiro, Brazil in 1986, his MBA degree from Institute for Business and market economy, Brazil in 1992, and his D.Sc. degree in computer science from the George Washington University, USA in 2004.

He is currently a principal scientist at Intelligent Automation, Inc., Rockville, MD, USA. His research interests include distributed systems, embedded systems, modeling and simulation, software engineering, and wireless networks.

Dr. Levy is a member of the IEEE and ACM.