A Distributed Efficient Clustering Approach for Ad Hoc and Sensor Networks

Jason H. Li[1], Miao Yu[2], and Renato Levy[1]

[1] Intelligent Automation, Inc., Rockville, MD 20855, USA*
{jli, rlevy}@i-a-i.com
[2] Department of Mechanical Engineering, University of Maryland, College Park, MD 20742, USA
mmyu@glue.umd.edu

Abstract. This paper presents a Distributed, Efficient Clustering Approach (DECA) for ad hoc and sensor networks. DECA 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 DECA is energy-efficient and resilient against node mobility.

1 Introduction

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 [9].

A closely related area of ad hoc networks is *wireless sensor networks (WSNs)* [1], which comprise of a higher number of nodes (in the thousands and more) scattered over some region. Sensor nodes are typically less mobile, and more densely deployed than mobile ad hoc networks (MANETs). The sensor nodes gather data from the environment and can perform various kinds of activities—such as collaborative processing of the sensor data, and performing some synchronized actions based on the gathered sensor data. Sensor nodes are usually heavily resource-constrained (especially on power), irreplaceable, and become unusable

* This work was supported by the Air Force Research Laboratory, grant FA8750-05-C-0161.

X. Jia, J. Wu, and Y. He (Eds.): MSN 2005, LNCS 3794, pp. 937–949, 2005.

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.

Given the potentially large number of mobile devices, scalability becomes a critical issue. To scale down networks with a large number of nodes, clustering protocols have been investigated for ad hoc and sensor networks in the literature [7][8][11]. 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 topology; none of the proposed schemes, even equipped with some local maintenance schemes, is satisfactorily resistant to node mobility beyond rare and trivial node movement. One of the purposes of this work is to propose a clustering protocol that is resilient against mild to moderate mobility where each node can potentially move.

In the hybrid energy-efficient distributed clustering approach (HEED) proposed for ad hoc sensor networks [12], clusterhead selection is primarily based on the residual energy of each node. The clustering process entails a number of rounds of iterations; each iteration exploiting 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. While HEED is one of the most recognized energy-efficient clustering protocols, we argue that its clustering performance can be further enhanced. In this work, we will present a distributed, energy-efficient clustering approach (DECA) that outperforms HEED in terms of energy-efficiency and possesses the advantages of better clustering efficiency and resilience against node mobility.

Our contributions are as follows. DECA aims to prolong network lifetime by efficiently organizing nodes into clusters, with the clusters resistant to node mobility. The protocol terminates without rounds of iterations as required by HEED, which makes DECA a less complex and more efficient algorithm. Further, DECA's efforts of minimizing control overhead render even smaller overhead than HEED, which implies better energy-efficiency in sensor networks.

The remainder of this paper is organized as follows. Section 2 describes the network model and the clustering problem that we address in this work. Section 3 presents the DECA protocol with correctness and complexity analysis. Performance evaluation is presented in Section 4, followed by the descriptions on relevant work in Section 5. We conclude the paper in Section 6.

2 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. 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 [5]. Neighboring nodes share the same wireless media, and each message is transmitted by a local broadcast. Nodes within the 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 network hardware in use. 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, $T_O >> T_C$. 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. 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. One such example is related to fast and efficient command and control in military applications, where nodes can frequently move.

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). 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 and overall network longevity.

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.

3 DECA Clustering Algorithm

The DECA algorithm structure is somewhat similar to that presented by Lin and Gerla [8] in that each node broadcasts its decision as the clusterhead in the neighborhood based on some local information and score function. In [8] 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 [12] uti-

lizes 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 announcement 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 [8] that the network topology does not change during the algorithm execution, and therefore it is valid for each node to wait until it overhears every higher-score 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. v then has to wait endlessly according to the stopping rule.

Similar assumption exists in HEED. Under node mobility, HEED will not halt though, since each node will terminate according to its probability-doubling procedure. However, we observe that the rounds of iterations are not necessary and can potentially harm the clustering performance due to the possibly excessive number of transmitted announcements.

We emphasize the important insights on distributed clustering: *those nodes with better scores should announce themselves earlier than those with worse scores.* In this work, 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 does not need to overhear every neighbor in order to stop; and, 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 saves on message transmissions lead to better energy efficiency.

3.1 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 the score function at each node as score $= w_1E + w_2C + w_3I$, where E stands for node residual energy, C stands for node connectivity, I stands for node identifier, and weights follow $\sum_{\mathrm{i}=1}^{3} w_{\mathrm{i}} = 1$. We put higher weight on node residual energy in our simulations. 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 in practice, like the DIFS parameter in IEEE 802.11. In our simulation, we choose $D_{\max} = 10$ms and the protocol works well. After the clustering starts, the procedure will terminate after time T_{stop}, which is another key parameter whose selection needs to take node computation capability and mobility into consideration. In the simulation, we choose $T_{\mathrm{stop}} = 1$s.

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 for some delay time.

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

```
II. RECEIVING-CLUSTERING-MESSAGE(id, cid, score)
 1  if (id == cid)
 2      then if (myCid == UNKNOWN)
 3               then if (score > myScore)
 4                        then myCid = cid;
 5                             cancelDelayAnnouncement ();
 6                             broadcastCluster (myId, myCid,score);
 7           elseif (score > myScore)
 8               then if (myId == myCid)
 9                        then needConversion = true;
10                        else
11                             convertToNewCluster ();
```

```
III. FINALIZE-CLUSTERING-ALGORITHM()
1  if (needConversion)
2      then if (!amIHeadforAnyOtherNode ())
3               then convertToNewCluster ();
4  if (myCid == UNKNOWN)
5      then myCid = cid;
6           broadcastCluster (myId, myCid, score);
```

On receiving such clustering messages, a node needs to check whether the node ID and cluster ID embedded in the received message are the same; same node ID 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 score, the node can simply join the advertised cluster and cancel its delayed announcement.

If the receiving node currently belongs to some other cluster, and the received score is better than its own score, two cases are considered. First, if the current node belongs to a cluster with itself as the head, receiving a better scored message 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 9 in Phase II) and defer it to finalizing phase, where it checks to make sure that no other nodes are affiliated with this node in the cluster as the head, before the switching can occur. But if the current node receiving a better-scored message is not itself a clusterhead, as an ordinary node, it can immediately convert to the new cluster, and this is the second case (line 11 in Phase II). It is critical to note that the switch process mandates that a node needs to leave a cluster first before joining a new cluster. 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.

3.2 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 1. *Eventually DECA terminates.*

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

Note that in order for DECA to outperform related protocols presented in [8] and [12] under node mobility, it is critical to design the key parameters $D_{\max}$ and T_{stop} appropriately taking node computation and mobility patterns into considerations. With carefully designed parameters, node needs not to wait (possibly in vain as in [8]) to transmit or terminate, nor need it to go through rounds of probabilistic announcement. In HEED, every iteration takes time t_{c}, which should be long enough to receive messages from any neighbor within the transmission range. We can choose $D_{\max}$ to be roughly comparable to (probably slightly larger than) t_{c} and DECA can generally terminate faster than HEED.

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

Proof. Suppose a node does not determine its cluster when entering Phase III. Then 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 a cluster to another one. The first two methods do not make a node join more than one clusters, and the switch procedure checks for consistency and mandates that a non-responsible node (a node not serving as 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 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 are formed executing line 4 to line 6 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 strictly determined using the score function.

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

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. □

To show that the algorithm is energy-efficient, we prove that the communication and time complexity is low.

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

Proof. In broadcastCluster method, a Boolean variable iAlreadySent (not shown in 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 either get executed or cancelled at line 5 in Phase II. Note that the cancellation is immediately followed by a transmission so each node will eventually transmit. □

Theorem 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 4, each node only sends one message and therefore there are only $|V|$ messages in the system. Thus the time complexity is $O(|V|)$. □

4 Performance Evaluation

We evaluate the DECA protocol using an in-house simulation tool called agent-based ad-hoc network simulator (NetSim). In our simulations, random graphs are generated so that nodes are randomly dispersed in a 1000m × 1000m region and each node's transmission range is bound to 250m. We investigate the clustering performance under different node mobility patterns, and the node speed ranges from 0 to 50m/s. For each speed, each node takes the same maximum speed and a large number of random graphs get generated. Simulations are run and results are averaged over these random graphs.

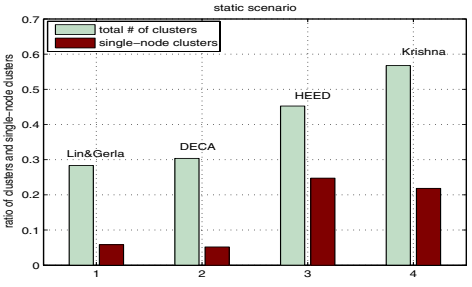

Fig. 1. Ratio of number of clusters. Static scenario.

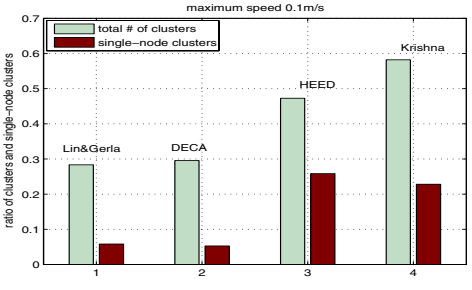

Fig. 2. Ratio of number of clusters. Maximum speed 0.1m/s.

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). While many other protocols generate lots of single-node clusters as node mobility gets more aggressive, our algorithm shows much better resilience. 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 choose [8] proposed by Lin & Gerla (LIN) as a representative for those general clustering protocols, and choose Krishna's algorithm (KRISHNA) [7] to represent dominating-set based clustering protocols. For energy-aware protocols, we choose HEED [12] to compare with DECA. From Fig. 1 (static scenario) and Fig. 2 (0.1m/s max. speed) it is easy to observe that KRISHNA has the worst clustering performance with the highest cluster-to-nodes ratio, while DECA and LIN possess the best performance. HEED performs in between.

Fig. 3, which combines Fig. 1 and Fig. 2, shows that all four protocols perform consistently under (very) mild node mobility. In fact, with maximum node speed set as 0.1m/s, both LIN and DECA perform exactly the same as their static scenarios, while HEED and KRISHNA degrade only to a noticeable extent.

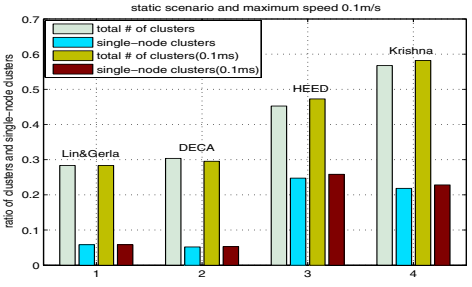

Fig. 3. Ratio of number of clusters. Put together.

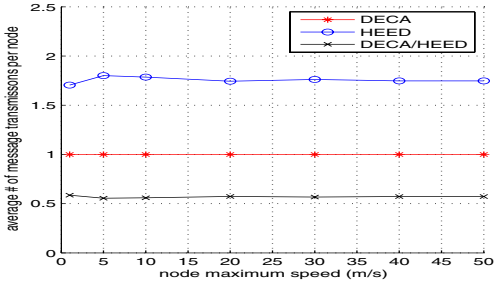

Fig. 4. Average number of transmissions per node and DECA/HEED ratio

During our simulations, as we increase the maximum node speed, both LIN and KRISHNA fail to generate clusters. 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 do not receive a message from each of its neighbors. Node mobility can make the holding node wait for ever. 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 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.

Fig. 4 shows that for DECA, the number of protocol messages for clustering remains one per node, regardless of node speed, as proven in Theorem 4. For HEED, the number of protocol messages is roughly 1.8 for every node speed, and a node running DECA transmits about 56% number of messages as that in HEED (shown as DECA/HEED in Fig. 4). The fact 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 [12]. Reducing the number of transmissions is of great importance, especially in sensor networks, since it would render better energy efficiency and fewer packet colli-

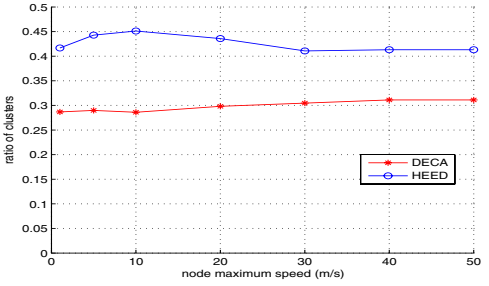

Fig. 5. Ratio of clusters to total number of nodes in network

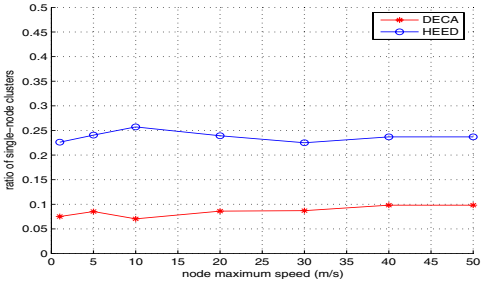

Fig. 6. Ratio of single-node clusters to total number of nodes in network

sions (e.g. CSMA/CA type MAC in IEEE 802.11). Fig. 5 and Fig. 6 illustrate the ratio of number of clusters and single node clusters to the total number of nodes in network. In both cases, DECA outperforms HEED.

Note that both DECA and HEED perform quite *consistently* under different maximum node speed and this is not coincident: a node in both DECA and HEED will stop trying to claiming itself as the potential clusterhead after some initial period (delayed announcement in DECA and rounds of iterations in HEED) and enters the finalizing phase. As a result, the local information gathered, which serves as the base for clustering, is essentially what can be gathered

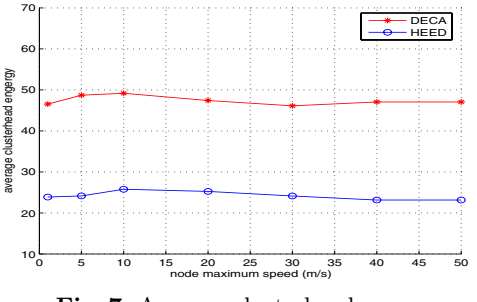

Fig. 7. Average clusterhead energy

within the somewhat invariant initial period which leads to consistent behaviors under different node mobility.

Further, we compare DECA and HEED with respect to the (normalized) average clusterhead energy in Fig. 7. Again both DECA and HEED perform quite consistently and DECA outperforms HEED with about **twice** the average clusterhead residual energy. This is in accordance with Fig. 5 where DECA consistently incurs fewer message transmissions than HEED. In sensor networks, sending fewer messages by each node in DECA while achieving the intended goal usually means energy-efficiency and longer node lifetime.

In addition, HEED may possess another undesirable feature in its protocol operation. Over time, each node's energy fades leading to a smaller probability of transmission in HEED for each node, which implies more rounds of iterations. As a result, more announcements could be sent and more energy could be consumed, which could lead to more messages sent and more energy consumed in the next round of clustering! In future work we will analyze HEED and execute more extensive simulations to see if such amplifying-effects really exist. DECA, on the contrary, does not posses this potential drawback even with energy fading, since each node only sends one message during the operation.

5 Related Work

Das and Sivakumar et al. [10] identified 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. The main drawback of this algorithm is that it still needs a non-constant number of rounds to determine a connected dominating set [11].

In [11] the authors proposed an 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. This algorithm is simple, which greatly eases its implementation, with low communication and computation cost; but it tends to create small clusters.

Similar to [8], Basagni [3] 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 [4] further generalized the scheme by allowing each clusterhead to have at most k neighboring clusterhead and described an algorithm for finding a maximal weighted independent set in wireless networks.

One of the first protocols that use clustering for network longevity is the Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol [6]. 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 gets evenly distributed among the sensors in the network. In addition, when possible, data are compressed at the clusterhead to reduce the number of transmissions. A limitation of this scheme is that it requires all current clusterheads to be able to transmit directly to the sink.

6 Conclusion and Future Work

In this paper we present a distributed, efficient clustering algorithm that works with resilience to node mobility 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.

It can be observed that in DECA the dispersed delay timers for clusterhead announcement assume the existence of a global synchronization system. While this might not be a problem for many (military) ad hoc network applications, for sensor networks synchronization can become trickier. It could be an interesting research to study time synchronization protocols combined with clustering protocol in sensor networks, with an effort to provide the maximum degree of functionality and flexibility with minimum energy consumption. Further, it could be interesting to observe how much improvement DECA can still maintain over HEED as transmission range varies.

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, March 2002.
2. 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. SAC-2, no. 1, pp. 226-237, January 1984.
3. S. Basagni, "Distributed clustering for ad hoc networks," in Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)
4. 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, ICC 2001, June 11-14 2001, pp. 1539-1543.
5. B. N. Clark, C. J. Colburn, and D. S. Johnson, "Unit disk graphs," Discrete Mathematics, vol. 86, pp. 165-167, 1990.
6. W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy efficient communication protocol for wireless microsensor networks," in Proceedings of the3rd Annual Hawaii International Conference on System Sciences, HICSS 2000,January 4-7 2000, pp. 3005-3014
7. 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.
8. 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.
9. 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.

10. 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.
11. 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.
12. O. Younis, S. Fahmy, "HEED: A Hybrid, Energy-Efficient,Distributed Clustering Approach for Ad Hoc Sensor Networks", IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 3, NO. 4, OCTOBER-DECEMBER 2004