Journal of Universal Computer Science, vol. 13, no. 1 (2007), 87-109
submitted: 29/8/05, accepted: 23/8/06, appeared: 28/1/07 © J.UCS

VCA: An Energy-Efficient Voting-Based Clustering Algorithm for Sensor Networks

Min Qin
(Computer Science Department, University of Southern California
Los Angeles, CA 90089, USA
mqin@usc.edu)

Roger Zimmermann
(Computer Science Department, University of Southern California
Los Angeles, CA 90089, USA
rzimmerm@imsc.usc.edu)

Abstract: Clustering provides an effective mechanism for energy-efficient data delivery in wireless sensor networks. To reduce communication cost, most clustering algorithms rely on a sensor's local properties in electing cluster heads. They often result in unsatisfactory cluster formations, which may cause the network to suffer from load imbalance or extra energy consumption. In this paper, we propose a novel *Voting-based Clustering Algorithm* (VCA) for energy-efficient data dissemination in wireless sensor networks. This new approach lets sensors vote for their neighbors to elect suitable cluster heads. VCA is completely distributed, location-unaware and independent of network size and topology. It combines load balancing, energy and topology information together by using very simple voting mechanisms. Simulation results show that VCA can reduce the number of clusters by 5-25% and prolong the lifetime of a sensor network by 10-30% over that of existing energy-efficient clustering protocols.

Keywords: Sensor network, Clustering, Cluster head, Voting, Energy-efficient, Data aggregation
Categories: C.2.2, C.2.4

1 Introduction

Energy consumption is a critical issue in sensor networks. Even with the latest energy-efficient circuit design, limited battery power hinders the deployment of wireless sensor networks. To reduce energy consumption, energy-efficient data dissemination techniques are needed. According to [Ratnasamy, 01], there are three major data dissemination methods: data-centric storage, local storage and external storage. For data-centric or local storage, data are kept within the network and queries are forwarded to nodes that store the requested data. In this paper we study the external storage problem, in which data need to be forwarded to a single sink outside the network. The sink location is often determined before all sensors are placed and a mobile sink is uncommon with current technology.

Clustering [Amis, 00, Bandyopadhyay, 03] is a useful approach to reduce energy dissipation in sensor networks. It is often coupled with data fusion [Hall, 92] to extend sensor lifetime. Each cluster elects one node as the cluster head. In Figure 1, for example, sensors are divided into 4 clusters. Nodes marked in solid circle serve as the heads for clusters. Data collected from sensors are sent to the cluster head first, and then forwarded to the sink. Compared to the sink, a cluster head is closer to sensors within the same cluster. Cluster heads can fuse data from sensors to minimize the amount of data to be sent to the sink. This can greatly reduce the energy consumption of sensor networks. When the network size increases, clusters can be organized hierarchically to further reduce energy consumption.

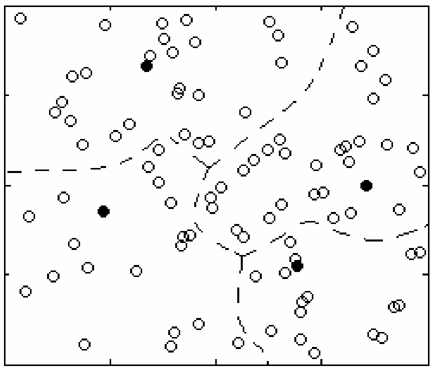


Figure 1: A sensor network with 4 clusters.

A lot of clustering algorithms [Choi, 04, Kalpakis, 02] require location information in order to work. Sensors need to know their position acquired through equipment like GPS (Global Positioning System), which is not available in many environments. Energy efficiency and load balancing are other considerations of clustering algorithms. Because most clustering protocols are based on local properties, clusters generated by these protocols may be undesirable. For example, cluster sizes are not well balanced or cluster heads with low energy may have lots of nodes subscribing to them. In this paper, we introduce a **voting-based clustering algorithm (VCA)** for energy-efficient data dissemination in wireless sensor networks. This new algorithm is capable of combining load balancing, topology and energy information together. Furthermore, VCA does not make any assumptions about sensor location and network topology. It is fully distributed and energy-efficient.

To evaluate the energy-efficiency of different clustering protocols, many studies use the network lifetime as criteria. The lifetime of a sensor network is often defined as the duration from the deployment of the network to the time when the first or the last sensor runs out of energy. We also calculate the network lifetime when comparing VCA with other clustering approaches. Simulation results show that our algorithm can generate a longer network lifetime than many traditional energy-efficient clustering approaches.

The paper is organized as follows. In Section 2, the related work is introduced. We present the communication model and challenges of distributed clustering algorithms in Section 3. Details of our VCA are introduced in Section 4. Section 5 presents the performance of our VCA and simulation results are reported in Section 6. Finally, we summarize the contributions and provide suggestions for further research in Section 7.

2 Related Work

Many clustering techniques for sensornets have been proposed in the past several years. Most of them can be classified into the following two categories: centralized or distributed. PEGASIS [Lindsey, 02], for example, is a centralized clustering algorithm. It uses a linear programming model to generate the optimal cluster formation for extending the lifetime of a sensor network. However, this requires sensors to have a global knowledge of the network. Spreading and collecting all sensors' information across a large network is often costly and impractical. Therefore, distributed clustering protocols are more desirable for large networks. Examples of distributed clustering algorithms include LEACH [Heinzelman, 00], HEED [Younis, 04], DCA [Basagni, 99] and WCA [Chatterjee, 02]. Because distributed clustering algorithms only have partial information of a network, clusters generated by these algorithms may not be optimal.

LEACH [Heinzelman, 00] uses a simple probability-based mechanism in electing cluster heads. Each sensor has the same probability of becoming a cluster head after the network is deployed. Sensors that do not elect themselves as cluster heads choose the closest head to join. The original LEACH protocol assumes that any two nodes can communicate with each other directly, which is unrealistic for large networks since it might cause too much energy to transmit data between two remote nodes. HEED [Younis, 04] extends LEACH by incorporating communication range limits and cost information. In HEED, the clustering process is divided into a number of iterations. The initial probability for a sensor to become a cluster head is defined as $\min(C_{prob} \times \frac{e_{residual}}{e_{max}}, p_{min})$, where $e_{residual}$ and e_{max} are the residual and the maximum energy of the node. In the next iteration, sensors that are not covered by any cluster head double their probability of becoming a cluster head. This procedure continues until all sensors are covered by at least one head. In the final stage, sensors join cluster heads that have the lowest cost within their range. The cost can be either the node degree or the residual energy of a cluster head. Both HEED and LEACH can finish their executions within a constant number of iterations. To balance the energy consumption of all sensors, both protocols require re-clustering after a period of time (called round), which causes extra energy consumption.

The Max-min clustering algorithm [Amis, 00] can generate *d*-hop clusters in O(*d*) time. Initially all sensors broadcast their node IDs to their neighbors. In the first *d* iterations, each sensor propagates the highest node ID they have heard to their neighbors. A node's ID cannot be propagated to nodes that are more than *d* hops away from it. In the next *d* iterations, sensors begin to propagate the minimum node ID ever heard. Each sensor then subscribes to the node whose ID was heard by it the last.

The generic clustering algorithms [Basagni, 97, Lin, 97] associate each sensor with some weight. The weight is calculated from a sensor's local properties, such as speed, node degree and residual energy. WCA and DCA are good examples of generic clustering. Initially, sensors broadcast their weight to their neighbors. Cluster heads are selected from those nodes that have the highest weight in their neighborhoods. However, defining a good weighing function for all sensors is difficult. A bad weighing function may cause a lot of ties during competition.

In [Choi, 04], the author introduced a two-phase clustering strategy. Initially, the network is partitioned into several clusters using traditional clustering protocols such as WCA. All sensors in the same cluster then send their location information to the cluster head and organize themselves into a chain. Data collected from all sensors are forwarded and fused along the chain. To reduce energy consumption, the length of every link along the chain is minimized.

Most of the above clustering strategies are suited for stationary or quasi-stationary sensornets. For networks with moving sensors, an adaptive clustering technique by predicting future link availability was introduced in [McDonald, 99]. Because many distributed clustering approaches use sensors' local properties in forming clusters, the cluster formations generated by these protocols might not be satisfactory. Some of the clustering approaches also need location information of all sensors, which is often infeasible due to size and energy limitations.

3 Clustering in Sensor Networks

3.1 Network Model

Table 1 lists all the terms and definitions used in this study. All nodes in the network are quasi-stationary and location-unaware. To model energy dissipation, we assume a $1/d^n$ path loss so that n is dependant on d. In order to transmit k bits data over distance d, the power spent on the radio is $E_T(k,d) = k(E_{Tx} + E_{amp}d^n)$. To receive a message of length k bits, the energy spent is $E_R(k) = kE_{Rx}$. In most studies, the relationship between E_{Tx} and E_{Rx} is simplified as $E_{Tx} = E_{Rx}$.

Term	Definition	Units
E_{Tx}	Energy dissipation rate to run the radio transmitter	J/bit
E_{amp}	Energy dissipation rate at the transmit amplifier	J/bit/m^n
E_{Rx}	Energy dissipated to run the receiver circuitry	J/bit
d_0	Threshold distance for E_{amp}	
N	Total number of nodes	
v_i	The ith sensor ($1 \le i \le N$)	
d_{ij}	Distance from sensor v_i to sensor v_j	m
R	Communication range	m
e_i	Residual energy of sensor v_i	J
$degree_i$	Node degree (number of neighbors) of sensor v_i	
S	An $[0,L]^2$ square area in which N sensors are dispersed	m^2
S_{CH}	The set of nodes that are cluster heads	
S_{WD}	The set of nodes that withdrew from voting	
S_{nbr}	The set of neighboring nodes	

Table 1: List of terms used and their definitions.

For all sensors in the network, we assume the following properties:

1) Sensors are homogeneous, they have the same communication range and energy consumption model.
2) All nodes are quasi-stationary.
3) Sensors are location unaware.
4) Due to location-unawareness and mobility, sensors use a fixed power level for broadcasting, which is dependant on the communication range.
5) Cluster heads can increase their transmission power so that they can communicate with the sink.

Many studies use network lifetime as a criteria in evaluating the energy-efficiency of different clustering protocols. The sensor network lifetime is divided into a number of rounds. Each round is comprised of a clustering phase T_{CP} and a network operation phase T_{NO} [Younis, 04]. During the clustering phase, a sensor either elects itself as a cluster head or joins a cluster. To avoid interference, each cluster head generates a TDMA schedule so that nodes can send data at different time slots. During the network operation phase, data generated at each sensor are sent to its cluster head according to the TDMA schedule. A cluster head fuses the data and sends them to the sink. To save the energy spent on clustering, the network operation phase may consist of multiple TDMA frames. Cluster heads can operate in two power levels so that they can directly communicate with the sink.

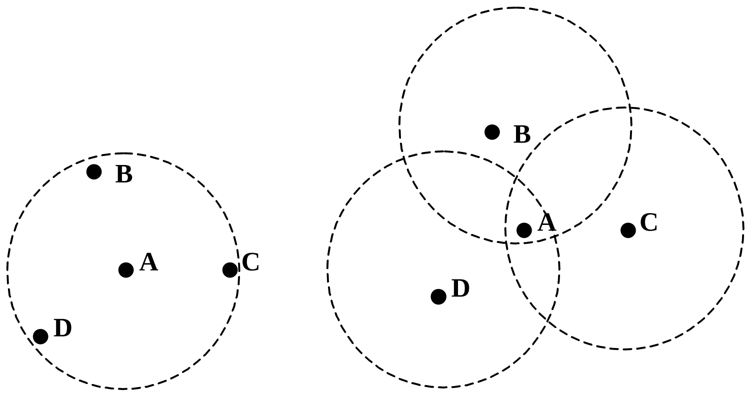
3.2 Problem Statement

Since a cluster head is responsible for fusing the data from sensors subscribing to it, only one data packet needs to be delivered to the sink out of a cluster. Thus the more clusters are present, the more messages need to be delivered to the sink. If the sink is far away from sensors, minimizing the number of clusters reduces overall energy consumption. However, finding the optimal cluster heads to cover a sensornet is NP-complete [Bollbas, 85]. Load balancing is another vital metric for distributed clustering algorithms. Sensors with high residual energy should have a higher opportunity of becoming cluster heads. Additionally, more sensors should subscribe to cluster heads with higher residual energy. However, many clustering algorithms try to balance the size of the clusters instead of the energy distribution.

When electing cluster heads, most distributed clustering approaches are based on local properties, such as node ID or residual energy. In LEACH and HEED, for example, sensors elect themselves as cluster heads based on a pre-determined probability. For generic clustering protocols, a sensor's weight is often calculated from its local properties. Due to the lack of information from neighbors, cluster formations generated by many distributed clustering algorithms are often unsatisfactory. Figure 2 shows a situation where undesirable results may be generated by most distributed clustering algorithms.

In Figure 2, sensors A, B, C and D all have 1J residual energy and the same communication range. Nodes B, C, and D are within the communication range of A. However, B, C and D cannot communicate with each other directly. As shown in Figure 2(a), if we group A, B, C and D together and select node A as the cluster head, data fusion can be performed at A and only one piece of message needs to be delivered to the sink. However, when using LEACH or HEED, all sensors have the same probability of becoming a cluster head. Suppose node C becomes a cluster head.

In that case, sensor A will subscribe to C and never elect itself as a cluster head. Since B and D are outside the range of C, they will form two individual clusters later, as shown in Figure 2(b). Thus three clusters are created instead of one. As a result, three messages are sent to the sink and no data fusion can be performed for B and D. This greatly increases the overall energy consumption of the network.



(a) A network of 4 nodes *(b) an undesirable cluster formation*

Figure 2: A clustering problem of traditional clustering approaches.

For weight-based clustering algorithms (such as WCA or DCA), the same problem occurs if the residual energy or the node ID is chosen as the weight for each sensor. Using the node degree as weight can solve the problem of Figure 2. However, for a large sensor network, many sensors may have the same degree. This causes a lot of ties when sensors are competing for cluster heads. Because cluster heads are among the high-degree nodes, they consume more energy and die quickly. After those high-degree nodes die, the remaining sensors are more likely to be separated apart. Additionally, sensors with extremely low energy may become cluster heads as long as they have high degrees. Using a complicated weighing function makes the meaning of the weight ambiguous and it is also hard to find suitable parameters. In order to solve the above problems of traditional clustering approaches, we introduce a voting based clustering algorithm in the next section.

4 Sensor Voting

Though a sensor has multiple neighbors, it may have different preferences of which neighbor is the most suitable cluster head for it. In order to represent such differences, we introduce a *voting-based clustering algorithm* (VCA). Similar to other distributed clustering algorithms, VCA does not guarantee an optimal cluster formation. However, it can improve cluster formations by incorporating information from neighboring nodes. VCA requires re-clustering after a period of time. This prevents the network from sticking to a few nodes as cluster heads and draining their energy quickly.

4.1 Vote Calculation

The heuristic behind sensor voting is that a sensor's *importance* should be reflected from all its neighbors (including itself) instead of from its local properties alone. Each sensor calculates the *suitability* that one of its neighbors (including itself) becomes its cluster head and casts a vote on that neighbor. At the same time, sensors collect votes from their neighbors and calculate the total vote received. The more votes a sensor accumulates, the more *important* it is in the whole network. During the clustering phase, sensors compete with each other based on the total votes each has received. Because sensors are location-unaware, topology and residual energy are the two primary factors in electing cluster heads. To be fair and to balance the workload, the following two rules are used to calculate the vote a sensor casts on its neighbors (including the sensor itself):

R1) The sum of the votes a node gives to all its neighbors (including itself) is 1.0.

R2) A neighbor with high residual energy should get more votes than a neighbor with low residual energy.

Rule R1 takes topology information into consideration. Since the total vote a sensor holds is 1.0, rule R1 has the following implications: If a sensor has more neighbors, each neighbor receives a smaller vote from that sensor. For a high-degree node, all its neighbors are candidate cluster heads for it. Therefore, an individual neighbor becomes less important to a high-degree node. On the contrary, sensors with more neighbors tend to collect higher votes since there are more voters for them. Thus cluster heads are likely to be those high-degree nodes. Electing high-degree nodes as cluster heads can save energy dissipation by increasing the number of messages to be fused together.

Rule R2 tries to balance the workload among all sensors. Because cluster heads consume more energy than other sensors, they should be selected from nodes with high residual energy. Based on rules R1 and R2, the following strategy is used to divide a sensor's vote among its neighbors (including itself). For a sensor v_i, the vote it casts on another sensor v_j is:

$$v(v_i, v_j) = \begin{cases} \dfrac{e_j}{\sum_{d_{ik} \le R} e_k} & , d_{ij} \le R \\ 0 & , d_{ij} > R \end{cases} \tag{1}$$

The total vote of sensor v_i is the sum of the votes from all its neighbors.

$$vote(v_i) = \sum_{d_{ij} \le R} v(v_j, v_i) \tag{2}$$

In Figure 2, for example, we have $v(A,B) = v(A,C) = v(A,D) = v(A,A) = 0.25$ and $v(C,A) = v(B,A) = v(D,A) = 0.5$. Thus $vote(A) = 1.75$ and $vote(B) = vote(C) = vote(D) = 0.75$. Sensor A becomes a cluster head since it holds the highest vote. After A becomes a cluster head, B, C and D will subscribe to it and only one cluster is created.

Observation 1: If the network is fully connected (all sensors can communicate with each other), the sensor with the highest residual energy becomes the cluster head.

When the network is fully connected, all sensors cast the same vote of $\frac{\boldsymbol{e_i}}{\sum \boldsymbol{e_i}}$ on sensor v_i. As a result, the higher residual energy a sensor has the higher vote it receives. Since high energy nodes are more likely to become cluster heads, VCA can balance the energy distribution across a sensor network by consuming the power of high energy nodes first.

Theorem 1: Suppose N sensors are randomly dispersed and each sensor's residual energy is uniformly distributed between 0 and 1J. For a sensor v_i with residual energy e and node degree d, the expected vote that it casts on itself is given by:

$$E\{v(v_i,v_i) \mid e_i = e, degree_i = d\} = \int_0^d \frac{e \sum_{k=0}^{d} (-1)^k \binom{d}{k} (u-k)^{d-1} \operatorname{sgn}(u-k)}{2(u+e)(d-1)!} du \quad (3)$$

Proof: Since a sensor's energy is independent of others', we can use d independent variables $X_1, X_2 \ldots X_d$ to denote the energy distribution of the d neighbors of sensor v_i. Each X_i is a uniform variate on the interval [0,1]. The distribution for the sum of d uniform variates [Renyi, 70] is given by:

$$P_{X_1+\ldots+X_d}(u) = \frac{1}{2(d-1)!} \sum_{k=0}^{d} (-1)^k \binom{d}{k} (u-k)^{d-1} \operatorname{sgn}(u-k)$$

According to Equation (1), the vote sensor v_i casts on itself is $v(v_i,v_i) = \frac{e}{e+u}$, where u is the sum of the energy of v_i's neighbors. Therefore the expected vote v_i casts on itself is given by

$$E\{v(v_i,v_i) \mid e_i = e, degree_i = d\} = \int_0^d \frac{e}{e+u} P_{X_1+\ldots+X_n}(u) du$$

$$= \int_0^d \frac{e \sum_{k=0}^{d} (-1)^k \binom{d}{k} (u-k)^{d-1} \operatorname{sgn}(u-k)}{2(u+e)(d-1)!} du$$

▮

According to rule R1, the sum of the vote v_i casts on itself and its neighbors is 1. By Theorem 1, the expected vote sensor v_i casts on all its neighbors is $1 - \int_0^d \frac{e \sum_{k=0}^{d} (-1)^k \binom{d}{k} (u-k)^{d-1} \operatorname{sgn}(u-k)}{2(u+e)(d-1)!} du$. Because each sensor is randomly dispersed over the field and has the same energy distribution, the expected vote each of v_i's neighbors receives from v_i should be equal.

Corollary 1: If all sensors are randomly dispersed with their residual energy uniformly distributed between 0 and 1J, the expected vote that a sensor v_i with residual energy e and node degree d casts to one of its neighbors v_j is:

$$E\{v(v_i,v_j)\,|\,e_i = e, degree_i = d\}$$
$$= \frac{1}{d}(1 - \int_0^d \frac{e\sum_{k=0}^{d}(-1)^k \binom{d}{k}(u-k)^{d-1}\operatorname{sgn}(u-k)}{2(u+e)(d-1)!}du)$$

Suppose all sensors are randomly dispersed over an S=[0, $L]^2$ (L>>R) square area. Then, for a given sensor, the probability that another sensor lies within its clusters radius is $(\pi R^2 / L^2)$. Thus we have:

Corollary 2: If N sensors are randomly dispersed over a [0, $L]^2$ (L>>R) square area and their residual energy is uniformly distributed between 0 and 1J, the expected vote that a sensor v_i with residual energy e gives to itself is:

$$\sum_{d=0}^{N-1}\left[\binom{N-1}{d}(\frac{\pi R^2}{L^2})^d(1-\frac{\pi R^2}{L^2})^{N-1-d}E\{v(v_i,v_i)\,|\,e_i = e, \mathrm{deg}ree_i = d\}\right]$$

Theorem 1 and Corollaries 1 and 2 provide useful theoretical analysis for future research. Also, they are helpful for vote calculation when part of the network information is unavailable. For example, if a sensor missed the opportunity to listen to its neighbors' energy information before the voting phase, it can use Theorem 1 and Corollary 1 to approximate the vote it should cast on itself and its neighbors.

4.2 Load Balancing

To balance the workload among cluster heads, we implemented two different strategies when a sensor is within the communication range of multiple cluster heads. The first strategy is for a node to join the cluster head with the minimum node degree. This strategy tries to balance the size of each cluster. However, a high-energy cluster head may have few neighbors subscribing to it while a low-energy head may have lots of sensors subscribing to it. Therefore, this strategy may not be fair for all the cluster heads.

Note that the energy consumption rate of a cluster head depends on the number of neighbors that subscribe to it. To balance the energy distribution, more sensors should subscribe to a high-energy cluster head. We define a fitness function to represent such properties of a cluster head. The fitness of a sensor v_i is defined as:

$$fitness(v_i) = \frac{e_i}{degree_i} \tag{4}$$

If the residual energy is the same, cluster heads with many neighbors have a smaller fitness than those with few neighbors. Our second load balancing strategy is to choose the cluster head with the highest fitness value within a sensor's communication range. This strategy balances the energy consumption among cluster heads.

According to Equation (4), sensors tend to subscribe to low-degree cluster heads. On the other hand, high-degree nodes are more likely to become cluster heads since more sensors vote for them. These two arguments do not conflict with each other because cluster head subscription is determined after cluster heads are elected. After electing high-degree nodes as cluster heads, sensors tend to subscribe to the cluster head that has the lowest degree in its vicinity. Choosing high-degree nodes as cluster heads can reduce the number of clusters in the network while subscribing to low-degree cluster heads can balance the energy distribution over different clusters.

4.3 VCA: Voting-based Clustering Algorithm

To calculate the vote of each neighbor, a sensor needs to know the residual energy of all its neighbors. We assume that sensor energy changes slowly over time. Since sensors are quasi-stationary, they need to send heartbeat messages regularly to update all their neighbors. Residual energy information can be attached to these heartbeat messages to reduce communication overhead.

We assume that all sensors are synchronized. After receiving a residual energy message, a sensor needs to update its vote to all neighbors. Only one message is needed to broadcast votes to all neighbors. Note that a small residual energy change of a neighbor does not change the votes significantly. A sensor does not need to re-broadcast its vote if the difference between its new vote and previous vote is within a certain threshold. In our experiment, we set this threshold to 5%. The vote information can also be attached to the heartbeat message to lower the communication cost. In Figure 3, t_1, t_2, t_3 are the timeout periods for a sensor to enter the next stage of clustering. We set them to 200 milliseconds in our experiment.

Figure 3 shows the procedure of our VCA algorithm. A sensor is called *covered* if there is one cluster head within its communication range. Similarly, a sensor is called *uncovered* if there is no cluster head within its neighborhood. During the initial stage, each sensor broadcasts the total vote it has received from its neighbors. Note that we use fitness as the load balancing strategy in Figure 3. To use the node degree for load balancing, lines 24-25 should be changed accordingly to find the cluster head with the minimum node degree.

The clustering stage is composed of a series of iterations. In each iteration, if a sensor finds that its vote is the highest among all uncovered neighbors, it elects itself as a cluster head. Fitness, node degree and identifier are used to break ties if there are any. The remaining sensors in that cluster head's neighborhood withdraw themselves from the election and will never become a cluster head in the following iterations. This guarantees that no two cluster heads are adjacent to each other.

Observation 2: No two cluster heads are within each other's range after VCA completes.

When a sensor becomes a cluster head, it sends out a CH (cluster head) advertisement to all its neighbors. The remaining sensors wait for t_2 seconds to check whether there are any cluster heads within their range. If a sensor finds that it was covered by a cluster head, it withdraws from the election by broadcasting a WITHDRAW message. Sensors withdraw from the election may still listen to CH advertisements to find out other cluster heads in their vicinity. All sensors will wait for t_3 seconds to collect these WITHDRAW messages. After each iteration, uncovered sensors may form a similar topology as shown in Figure 2. To isolate them from those that have already been covered, an uncovered sensor re-calculates its total vote by ignoring the votes from its neighbors that have sent the WITHDRAW messages. If there is any change in its total vote, it needs to re-broadcast the new total vote.

Sensors that are still uncovered collect these vote update messages and use the new vote in competing with each other.

```
Procedure Init
begin
1. $S_{nbr} \leftarrow \{v \mid v$ is within my communication range$\}$
2. $S_{CH} \leftarrow \phi$, $S_{WD} \leftarrow \phi$, $S_{uncovered} \leftarrow S_{nbr}$, $currentvote \leftarrow \sum_{v_j \in S_{nbr}} v(v_j, nodeID)$
3. broadcast currentvote and fitness to $S_{nbr}$
end

procedure VCA
begin
1. while ($cluster\_head = \phi$) do
2.     if ($S_{CH} = \phi$) then
3.         $newvote \leftarrow \sum_{v_j \in S_{uncovered}} v(v_j, nodeID)$
4.         if $newvote \neq currentvote$ then
5.             $currentvote = newvote$
6.             broadcast vote_update(nodeID, newvote)
7.         end if
8.         collect v_updates from neighbors in $t_1$ sec
9.         if $currentvote = max\{vote(v) \mid v \in S_{uncovered}\}$ and I win tie-breaks then
10.            $CH(nodeID) \leftarrow$ true, $Cluster\_head \leftarrow nodeID$
11.            send a CH advertisement and return
12.        end if
13.    end if
14.    collect incoming CH advertisements in $t_2$ sec
15.    $S_{CH} \leftarrow S_{CH} \cup \{v \mid$ CH overheard from $v\}$
16.    if $S_{CH} \neq \phi$ then
17.        broadcast a WITHDRAW packet
18.    end if
19.    collect incoming WITHDRAW in $t_3$ sec
20.    $S_{WD} \leftarrow S_{WD} \cup \{v \mid$ WITHDRAW heard from $v\}$, $S_{uncovered} \leftarrow \{v \mid v \in S_{nbr}$ and $v \notin S_{CH}$
       and $v \notin S_{WD}\}$
21.    if $S_{CH} \neq \phi$ then
22.        $head \leftarrow highest\_fitness(S_{CH})$;
23.        if $fitness(head) \geq max\{fitness(u) \mid u \in S_{uncovered}\}$ then
24.            $cluster\_head \leftarrow head$, join_cluster(cluster_head); return;
25.        end if
26.    end if
27. end while
end
```

Figure 3: Pseudo-code of VCA.

As indicated in the pseudo-code, a sensor finishes its clustering process in either of the following two states:

1. It becomes a cluster head.
2. It is covered by a cluster head and all neighbors that have higher fitness or smaller node degree (depending on load balancing strategy) than that cluster head withdraw from the election.

Situation 2 implies that a sensor always joins the cluster head that has the highest fitness (or minimum node degree, depending on the load balancing strategy in Section 4.2) within its communication range. Combined with our discussion in Section 4.1, VCA can integrate load balancing, energy and topology information together by using simple voting mechanisms.

A special case occurs when sensors are aligned into a chain. As shown in Figure 4, sensors v_1, v_2, ... , v_n form a chain with increasing node identifiers. Suppose they all have the same amount of residual energy. Consequently, v_3 ... v_{n-2} all receive a total vote of 1.0 and they have the same fitness values and node degree. To break the tie in their total vote, each of them needs to wait until a neighbor that has a smaller identifier or higher vote becomes a cluster head. Because v_2 and v_{n-1} receive a total vote of 1.17, they are elected as cluster heads in the first iteration. After v_2 and v_{n-1} become cluster heads, v_1, v_3, v_{n-2} and v_n withdraw themselves from the voting process. In the next iteration, v_4 updates its vote by ignoring the vote from v_3. As a result, node v_5 elects itself as a cluster head because it has a higher vote than v_4 and its node identifier is smaller than that of v_6. The above process continues and the resulting cluster heads are v_2, v_5, v_8...v_{n-1}. A total of n/3 iterations are needed to cluster all sensors. It can be easily shown that our algorithm can generate the minimum number of clusters in this case.

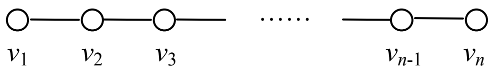


Figure 4: Clustering sensors that are aligned into a chain

***Theorem 2*:** Assume sensors are dispersed in an area S=[0, L]^2, the time complexity of VCA is $O(\min\{N, \frac{L^2}{R^2}\})$.

Proof In each iteration, at least one sensor finds that its total vote is the largest among all uncovered sensors. So at most N iterations are needed before all sensors decide their roles. According to *Observation 2*, no two cluster heads are within each other's range. So the maximum number of cluster heads in the network is bounded by $O(L^2 / R^2)$ and the time complexity of VCA is bounded by $O(\min\{N, L^2 / R^2\})$. ▌

Theorem 3: The message complexity of VCA is $O(N)$.

Proof Each sensor sends out one message during the initialization stage and when it becomes/joins a cluster head. During the clustering procedure, the only situation in which a sensor needs to broadcast a message is when its total vote changes. This happens when some of its neighbors become covered in the previous iteration. In Figure 5, for example, sensor C becomes a cluster head and sensor A is still not covered by any cluster head. If sensor B is within the communication range of C, sensor A needs to re-calculate its vote by ignoring the vote from B.

Since sensor A is not covered by C, we have 2R>d(A,C)>R. Thus when a sensor re-broadcast its vote, there must be a cluster head within the shaded region in Figure 5. According to *Observation* 2, no two cluster heads are within each other's range. Thus the number of cluster heads within the shaded area in Figure 5 cannot exceed a fixed upper bound *C*. Thus the number of messages each sensor send is O(1) and O(*N*) is the total message complexity. ▌

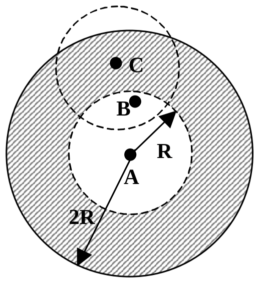


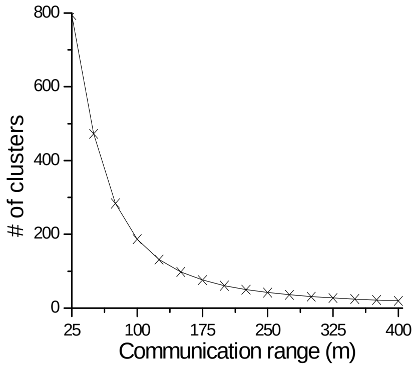
Figure 5: Non-neighboring cluster heads around a sensor.

After forming the clusters, cluster heads can communicate with each other to form multi-hop clusters. Accoding to [Younis, 04], as long as the inter-cluster transmission range R_t for cluster heads satisfies $R_t \geq 6\sqrt{2}c$, where $c^2N = aL^2 \ln L$ for some a>0, any two cluster heads in neighboring areas can communicate with each other. Details of how to derive *a* and *c* can be found in [Younis, 04].

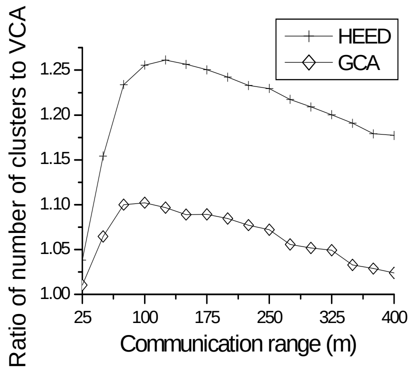
Theorem 4: VCA can form multi-hop clusters if the proper inter-cluster transmission range is set. We omit the proof since it was given in [Younis, 04].

5 Analytical Performance Evaluation

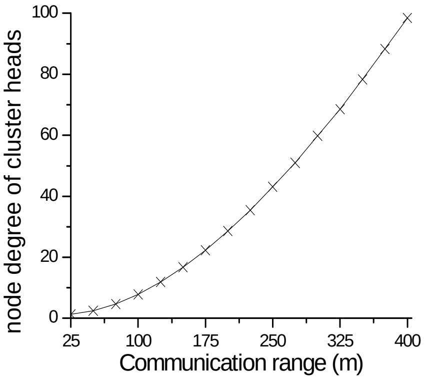
To evaluate the performance of VCA, we compared it with HEED and a generic weight-based clustering algorithm (GCA). Unlike VCA, HEED and GCA rely on local properties in clustering sensors. GCA functions similarly to VCA except for the voting stage. It associates each sensor with some weight. In each iteration, if a sensor finds that its weight is higher than those of its uncovered neighbors, it elects itself as a cluster head. Because sensors are location unaware, we use residual energy as weight in GCA. For HEED, C_{prob} and p_{min} are initialized to 0.05 and 0.0005 respectively and the node degree is chosen as the cost. A sensor joins the cluster head with the minimum node degree if there are multiple heads around it.



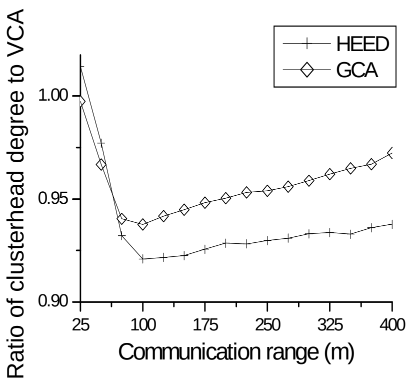
(a) average number of clusters generated by VCA



(b) ratio of number of clusters by HEED and GCA to that of by VCA



(c) average clusterhead degree of VCA



(b) ratio of average clusterhead degree of HEED and GCA to that of VCA

Figure 6: Comparison of number of clusters and clusterhead degree as a function of the radius.

We conducted a total of 500 independent experiments. For each experiment, 1000 sensors were randomly dispersed into an area of 2000×2000m^2. Each sensor was given a random energy between 0 and 1J. Because changing all nodes' communication range has a similar effect as changing the node density, our experiments shows the results with different communication range settings for all the sensors. Similar results can be obtained by changing the number of nodes in the network.

Figures 6(a, b) show the average number of clusters generated by VCA compared to those generated by HEED and GCA. Both HEED and GCA generate more clusters than our voting-based clustering algorithm. When the communication range is small, a large number of clusters is required to cover all the sensors. Most of the cluster heads have few neighbors around it, as shown in Figure 6(c). When the communication range increases, the number of clusters required to cover all sensors drops and the average node degree of cluster heads increases.

According to our discussion in Section 4.1, nodes with more neighbors tend to receive a higher vote. Therefore, it is very likely that cluster heads have higher degree in VCA. Because high-degree nodes play an important role in connecting all sensors, selecting them as cluster heads can boost the effect of data fusion in saving energy. In Figures 6(c, d), the average degree of cluster heads is higher in VCA than those in the other two clustering algorithms. As a result, VCA can reduce the number of clusters. As shown in Figures 6(a, b), HEED generates 20-25% more clusters than VCA. This may cause extra energy consumption since each cluster head needs to communicate with the remote sink.

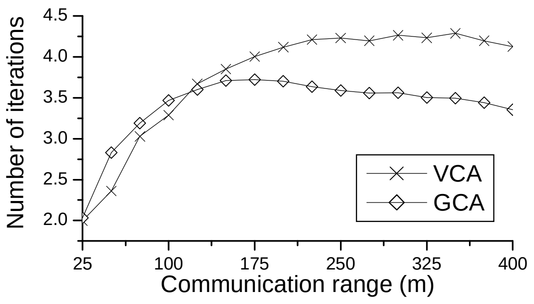
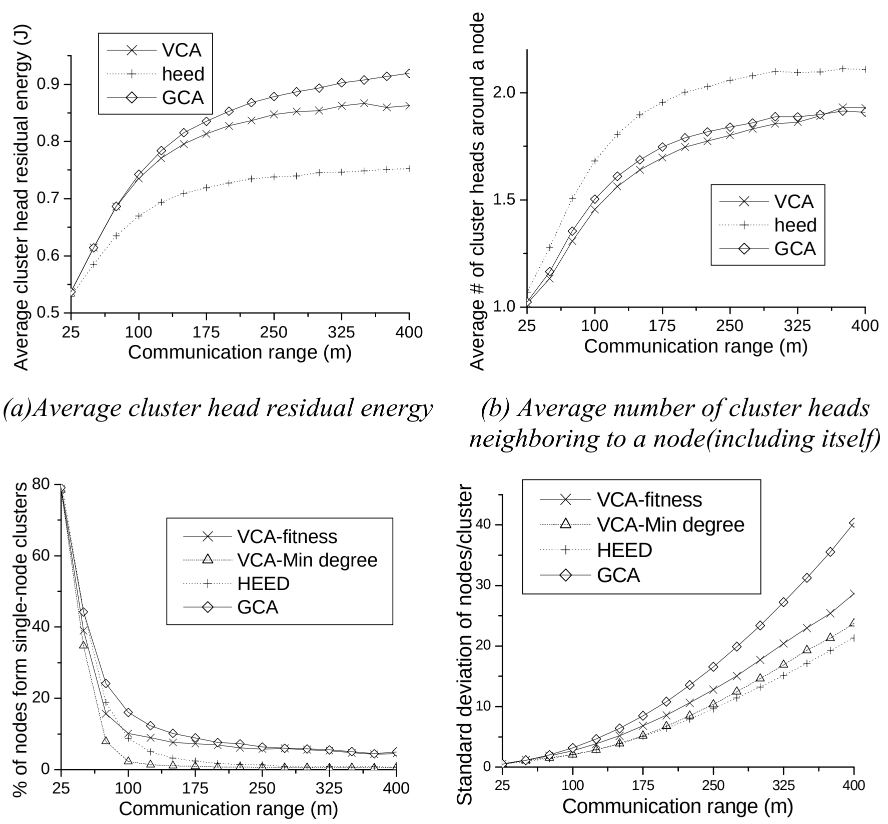


Figure 7: Average number of iterations performed.

Figure 7 shows the average number of iterations needed for the clustering procedure. HEED uses a fixed number of iterations depending on the initial probability that a sensor becomes a cluster head. Therefore, we only compare VCA with GCA since they work similarly. In VCA, sensors that are not covered after each iteration need to recalculate their vote. This may cause minor oscillations of choosing which node is a cluster head, especially when the network is denser. Since increasing the communication range has a similar effect as making the network denser, as shown in Figure 7, the difference of the number of iterations between VCA and GCA increases as the communication range increases. However, both algorithms need 2 to 5 iterations to finish, which is very efficient.

Figure 8(a) shows the average residual energy of cluster heads. Cluster heads consume more energy than other sensors and therefore they should have higher residual energy. GCA has the highest average residual energy for cluster heads. Since we use residual energy as weight in GCA, cluster heads are always chosen from sensors with high residual energy. VCA can achieve comparable average head energy to GCA. This is due to the fact that nodes with higher residual energy are more likely to get higher votes. For HEED, sensors with low residual energy can still become cluster heads with some probability. As a result, HEED has the lowest average residual energy of cluster heads among all three clustering algorithms.



(a)Average cluster head residual energy *(b) Average number of cluster heads neighboring to a node(including itself)*

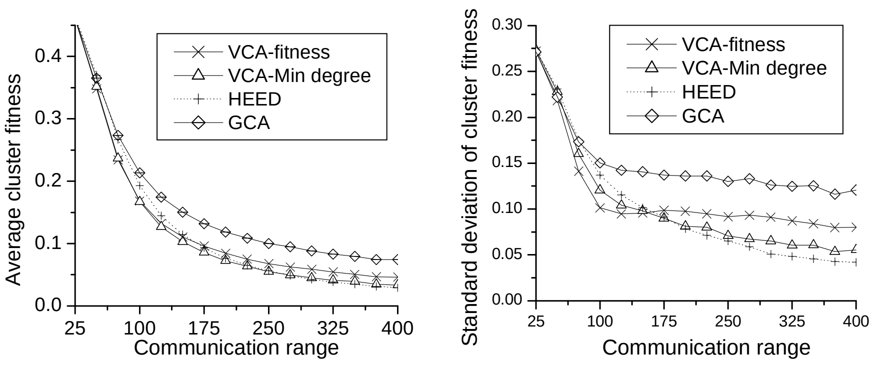
(c) Percentage of sensors in single-node clusters *(d) Standard deviation of number of nodes per cluster*

Figure 8: Average cluster properties.

For each node, we compared the average number of cluster heads neighboring to it in Figure 8(b). If there is more than one cluster head around a sensor, the sensor may need to negotiate with all of them to avoid transmission interference. Therefore, it may cause extra energy consumption for that sensor. Because VCA can reduce the number of cluster heads in the network, most nodes are covered by only one cluster head. On the other hand, HEED tends to disperse cluster heads randomly. As a result, it is possible that a node is covered by more than one cluster head.

Figure 8(c) shows the percentage of sensors in single-node clusters. Single-node clusters are not a desirable feature in sensor networks. For VCA, we tried two load balancing strategies as mentioned in Section 4.2. When using the maximum *fitness* for load balancing, the percentage of sensors in single-node clusters is higher than when using the minimum node degree. This is due to the fact that using fitness does not try to balance the size of the clusters. Instead, it tries to balance the energy distribution in a sensor network. As shown in Figure 8(c), when using the minimum node degree for load balancing, VCA has the lowest single-node percentage among all 3 algorithms.

Figure 8(d) shows the standard deviation of the number of nodes per cluster. Similarly to the single-node situation, VCA-min degree results in a smaller standard deviation than VCA-fitness since it tries to balance the size of clusters. For HEED, each sensor has some probability of becoming a cluster head, and the cluster heads are distributed randomly over the network. This makes HEED achieve a respectable balance of cluster sizes. Because GCA has no balance control over the cluster size, it has the highest standard deviation among all three algorithms.



(c) Average cluster fitness *(d) Standard deviation of cluster fitness*

Figure 9: Average cluster fitness.

Residual cluster head energy may not be proportional to the size of a cluster. For example, a high-energy cluster head may have few neighbors subscribing to it while a low-energy head may have lots of sensors subscribing to it. Therefore, using cluster size and residual energy separately cannot accurately reflect the fairness of energy distribution over different clusters. To solve this problem, we introduce a new metric called *actual fitness* for clusters. The actual fitness of a cluster C is defined as:

$$actualfitness(C) = \frac{residual\ energy\ of\ the\ cluster\ head\ of\ C}{number\ of\ nodes\ in\ C} \qquad (5)$$

Unlike the fitness of a sensor, the actual fitness of a cluster reflects the relationship between the cluster head energy and the cluster size, rather than the cluster head's degree. Figure 9(a) shows the average cluster fitness resulting from different clustering algorithms. In Figure 9(a), GCA has the highest actual fitness. This is because in Figure 8(a), GCA has the highest average cluster head residual energy. However, a high actual fitness value does not mean that the energy distribution is fair. Figure 9(b) shows the standard deviation of actual cluster fitness. As indicated by Figure 9(b), there is a big difference between the actual fitness of different clusters generated by GCA. Therefore, energy is not fairly distributed over all clusters. HEED has the smallest standard deviation of cluster fitness. This is due to the fact that the average cluster head energy of HEED is much lower than that of VCA and GCA. VCA-fitness tries to increase the actual fitness of all clusters. However, since cluster sizes are not well balanced, the standard deviation of cluster fitness of VCA-fitness is slightly higher than that of VCA-min degree.

6 Simulation Results

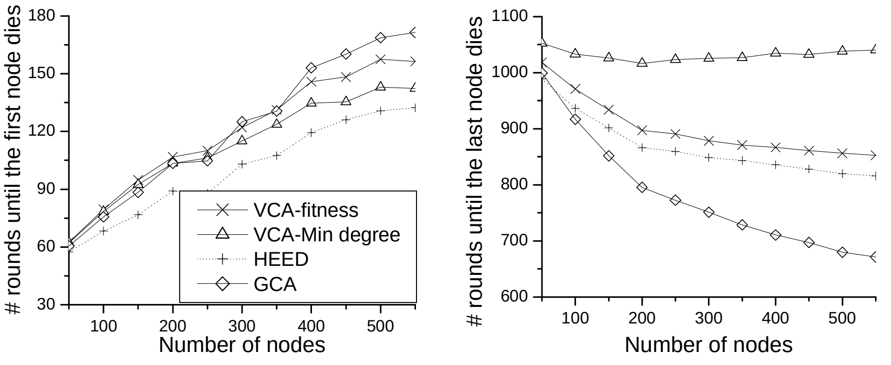
Table 2 lists all the simulation settings. Because clustering consumes energy, it should not be executed frequently. To reduce clustering costs, the network operation phase is set to 10 TDMA frames each round. We use a one-hop cluster setting in our simulation. Data gathered from sensors are sent to the cluster heads, which fuse data and send them to the sink. Because sensors are location unaware, we set their transmission range identical to the communication range. A cluster head is capable of increasing its transmission power so that it can communicate with the sink. We conducted 100 independent simulations and calculated the average lifetime of the network. Our radio model is similar to that of [Younis, 04], in which $E_{Tx}= E_{Rx}$ =50 nJ/bit, E_{amp}=100 pJ/bit/m^2 (n=2) when $d<d_0$ or 0.0013pJ/bit/m^4 (n=4) when $d>d_0$, where d_0=75m is the threshold distance. For each simulation setting, we conducted 100 independent simulations and calculated the average lifetime of the network.

Parameter	**Value**
S	[0, 150]^2
Sink location	(75,200)
Communication range	20 m
Data packet	250 byte
Clustering packet	30 byte
WITHDRAW packet	10 byte
Network operation phase	10 TDMA frames
Energy for data fusion	5nJ/bit/signal
Initial energy	2J
Threshold distance(d_0)	75m
E_{Tx}	50nJ/bit
E_{amp}	10 pJ/bit/m^2 or 0.0013pJ/bit/m^4

Table 2: Simulation settings.

To analyze the performance of VCA, we compared it with HEED and GCA. HEED is a state-of-the-art energy-efficient protocol for extending the network lifetime. Similar to VCA, HEED can generate well-distributed cluster heads and achieve load balancing to a certain degree. The original GCA algorithm does not require re-clustering after a period of time. As a result, cluster heads will quickly deplete their energy. To improve the performance of GCA, we let sensors re-elect their cluster heads as they do in HEED and VCA. Cluster heads are chosen from the sensors that have the maximum residual energy in its vicinity.

Figure 10(a) shows the number of rounds until the first sensor depletes its energy. Because VCA generates fewer clusters, it can save a portion of the energy used to communicate between cluster heads and the sink. As shown in Figure 10(a), VCA-fitness works better than VCA-min degree, especially when the network becomes denser. This is due to the fact that VCA-fitness tries to balance the energy over the network instead of the cluster size. By using VCA-min degree, a cluster head with low residual energy may have a large number of sensors subscribing to it. This shortens the lifetime of a particular set of sensors. When the network becomes denser, this situation is more likely to happen.



(a) when the first node dies *(b) when the last node dies*

Figure 10: Network lifetime as a function of number of nodes.

As shown in Figure 10(a), VCA can generate a longer lifetime than HEED in most configurations. However, the lifetime of VCA-min degree is close to that of HEED when the number of nodes increases to 550. Additionally, GCA outperforms VCA when node density increases. When the network becomes denser, each sensor receives more vote updates and WITHDRAW messages from its neighbors during the clustering phase. Sending and receiving these messages consumes energy. However, HEED and GCA are not severely affected by network density. For HEED, only cluster heads broadcast messages during the clustering phase. For GCA, each node only needs to broadcast its weight once. As a result, VCA works most effectively when the network is sparse.

Figure 10(b) shows the number of rounds when the last sensor depletes its energy. VCA works much better than HEED and GCA in this case. However, unlike in Figure 10(a), the last sensor dies much later in VCA-min degree than in VCA-fitness. Because VCA-fitness tries to balance the energy of all nodes, as indicated in Figure 11, all sensors are likely to die at a similar time. On the other hand, in VCA-min degree, some sensors may have high residual energy and die much later than others. This results in a much longer time until the last sensor in the network dies. As shown in Figures 10(a, b), VCA can outperform HEED by 5-30% in terms of network lifetime. In GCA, nodes with the highest residual energy are always chosen to be cluster heads. Because sensors recluster themselves after each round and cluster heads consumes more energy, these cluster heads will soon be replaced by others. As a result, sensors will rorate their role as cluster heads in GCA. As shown in Figure 11, sensors tend to die at a similar time. Therefore, the last node dies much earlier in GCA.

When fixing the number of sensors to 500, Figure 11 shows the average number of sensors alive as a function of time when continuously running these clustering protocols. VCA works much better than HEED and GCA. Most sensors in VCA die later than those in HEED in GCA. Because VCA-fitness tries to balance the energy of all sensors, there is not a big gap between two consecutive deaths of sensors. On the other hand, for VCA-min degree, 20% of the nodes die later than other nodes as shown in Figure 11.

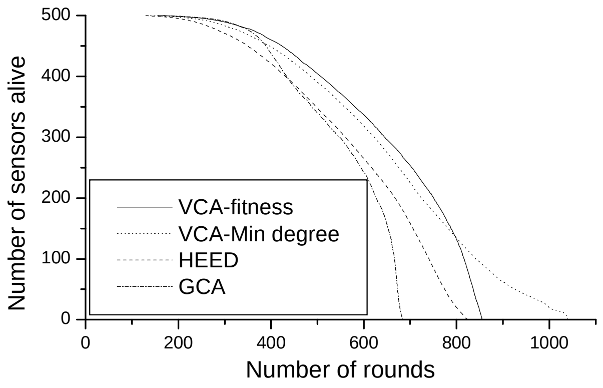
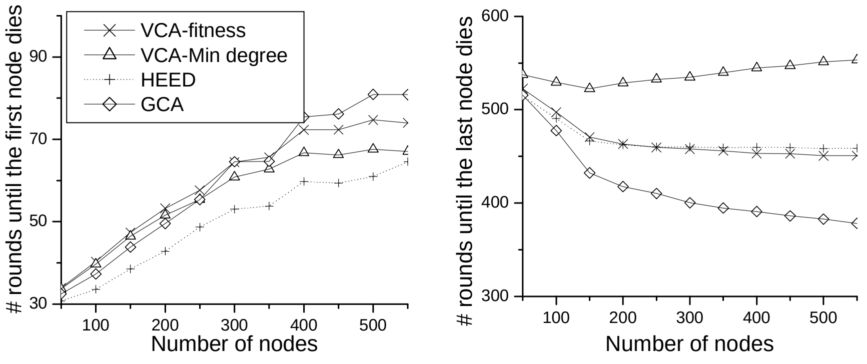


Figure 11: Number of sensors alive when running the protocol continuously for a 500-node network.

Figure 12 shows the network lifetime when the network operation phase is set to 20 TDMA frames. Compared to the settings in Table 2, each cluster head sends more messages to the sink during each round. Consequently, cluster heads consume about twice the amount of energy in each round. As shown in Figure 12, the network lifetime is almost half of that of Figure 10. Because only one clustering is performed during each round, the amount of energy spent on clustering is largely reduced when the network operation phase is prolonged.

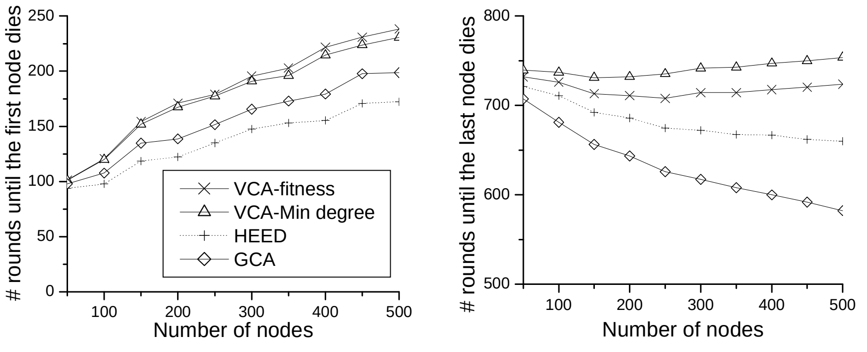
As shown in Figure 12, changing the network operation time yields similar curves as those shown in Figure 10. However, unlike in Figure 10(b), the last sensor in Figure 12(b) dies a little later in HEED than in VCA-fitness. When the network operation time increases, the energy requirement for a sensor to become a cluster head also rises. Sensors with low residual energy can no longer become a cluster head since their energy cannot last for a round. VCA-fitness tries to balance the energy of all sensors. As a result, when the energy requirement rises, almost all sensors cannot become cluster heads at the same time. However, for HEED, some high energy nodes may not be chosen as cluster heads during the initial rounds and can survive for a long period of time. As a result, the last sensor in HEED may die a little later than that in VCA-fitness as the network operation time increases.



(a) when the first node dies *(b) when the last node dies*

Figure 12: Network lifetime when the network operation time is increased to 20 TDMA frames

To study the performance of VCA when nodes are mobile, we conducted a number of simulations with the same settings shown in Table 2. We choose the random waypoint mobility model[Camp, 02] to model node movement. For simplicity, we assume that all sensors do not move during the network operation phase. As a result, node movement does not interfere with the fusion and transmission of data packet in each round. For each node, we set its maximum speed to 3m/round. We conducted 100 independant simulations for each node density setting and calculated the average network lifetime. Figure 13 shows the simulation results.



(a) when the first node dies *(b) when the last node dies*

Figure 13: Network lifetime when sensors are mobile

Compared to Figure 10, the first sensor dies much later while the last sensor dies much earlier when sensors become mobile. Also, energy distribution can be easily balanced due to node mobility. For example, a high energy sensor can move to an area where low energy sensors concentrate. As a result, node mobility can greatly balance the energy consumption among all sensors. Since VCA can balance the energy distribution across a sensornet to some extent, it can benefit greatly from node mobility. As shown in Figure 13, VCA performs much better than HEED and GCA when nodes become mobile. It can out perform HEED by 8-40% in terms of the lifetime when the first node dies, or 2-9% in terms of the lifetime when the last node dies.

7 Conclusions

We introduced a voting-based clustering algorithm for maximizing the lifetime of quasi-stationary sensor networks. The algorithm is energy-efficient, location unaware, and fully distributed. Additionally, we introduced two load balancing strategies, one that balances the size of each cluster while the other balances the energy distribution over the network. Simulation results show that our algorithm can reduce the number of clusters by 5-20% and prolong the lifetime of a sensor network by 5-30% over HEED.

Because of its limited functionality, a cluster head cannot have too many sensors subscribed to it. We plan to investigate a degree limit on cluster heads, where the number of sensors subscribing to a cluster head cannot exceed a certain limit. Additionally, our current simulation concentrates on the one-hop cluster performance. It will be extended to energy-efficient data dissemination in multi-hop clusters.

Acknowledgements

This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), CMS-0219463 (ITR), IIS-0534761, and equipment gifts from the Intel Corporation, Hewlett-Packard, Sun Microsystems and Raptor Networks Technology.

References

[Amis, 00] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh, "Max-Min D-Cluster Formation in wireless Ad Hoc Networks," in Proc. IEEE INFOCOM, March 2000.

[Bandyopadhyay, 03] S. Bandyopadhyay and E. J. Coyle, "An Energy Efficient Hierarchinal Clustering Algorithm for Wireless Sensor Networks", in Proc. IEEE INFOCOM, San Fransisco, April, 2003.

[Basagni, 99] S. Basagni, "Distributed Clustering for Ad Hoc Networks", in Proc. International Symposium on Parallel Architectures, Algorithms and Networks, June 1999, pp. 310-315.

[Basagni, 97] S. Basagni, I. Chlamtac and A. Farago, "A generalized clustering algorithm for peer-to-peer networks", in Proc. Workshop on Algorithmic Aspects of Communication (satellite workshop of ICALP), July 1997.

[Bollbas, 85] B. Bollbas, Random Graphs, Academic Press, 1985.

[Camp, 02] T. Camp, J. Boleng and V. Davies, “A survey of mobility models for ad hoc network research”. Wireless Communications F4 Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, 2(5), 2002, pp. 483-502.

[Chatterjee, 02] M. Chatterjee, S. Das and D. Turgut, “WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks”, Journal of Cluster Computing, Special issue on Mobile Ad hoc Networking, No. 5, 2002, pp. 193-204.

[Choi, 04] W. Choi, P. Shah and S. K. Das, “A Framework for Energy-Saving Data Gathering Using Two-Phase Clustering in Wireless Sensor Networks”, in Proc. Mobile and Ubiquitous Systems, Boston, MA, August 2004.

[Hall, 92] D. Hall, “Mathematical Techniques in Multisensor Data fusion,” Artech House, Boston, MA, 1992.

[Heinzelman, 00] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-Efficient Communication Protocols for Wireless Microsensor Networks.” In Proc. Hawaiian Int'l Conf. on Systems Science, January 2000.

[Kalpakis, 02] K. Kalpakis, K. Dasgupta, and Parag Namjoshi, “Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks,” In Proc. of the 2002 IEEE International Conference on Networking (ICN'02), Atlanta, August, 2002, pp. 685-696.

[Lin, 97] C. R. Lin and M.Gerla, **“**Adaptive Clustering for Mobile Wireless Networks”, in IEEE Journal of Select Areas Communication, September, 1997.

[Lindsey, 02] S. Lindsey and C. S. Raghavendra, “PEGASIS: Power Efficient GAtshering in Sensor Information Systems,” 2002 IEEE Aerospace Conference, March 2002, pp. 1-6.

[McDonald, 99] 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, Aug. 1999, pp. 1466-1487.

[Ratnasamy, 01] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, L. Yin S. Shenker, and F. Yu. “Data-centric storage in sensornets”. In ACM First Workshop on Hot Topics in Networks, 2001.

[Renyi, 70] A. Renyi. “Probability Theory”. North-Holland, Amsterdam, 1970.

[Younis, 04] O. Younis and S. Fahmy, “Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach”, in Proc. IEEE INFOCOM 2004, Hong Kong, China, March, 2004.