

Distributed Energy-Efficient Hierarchical Clustering for Wireless Sensor Networks

Ping Ding, JoAnne Holliday, Aslihan Celik
{pding, jholliday, acelik}@scu.edu
Santa Clara University

Abstract: Since nodes in a sensor network have limited energy, prolonging the network lifetime and improving scalability become important. In this paper, we propose a distributed weight-based energy-efficient hierarchical clustering protocol (DWEHC). Each node first locates its neighbors (in its enclosure region), then calculates its weight which is based on its residual energy and distance to its neighbors. The largest weight node in a neighborhood may become a clusterhead. Neighboring nodes will then join the clusterhead hierarchy. The clustering process terminates in $O(1)$ iterations, and does not depend on network topology or size. Simulations show that DWEHC clusters have good performance characteristics.

1 Introduction

Sensor nodes are relatively inexpensive and low-power. They have less mobility and are more densely deployed than mobile ad-hoc networks. Since sensor nodes are always left unattended in sometimes hostile environments, it is difficult or impossible to re-charge them. Therefore, energy use is a key issue in designing sensor networks.

Energy consumption in a sensor network can be due to either *useful* or *wasteful* work. Useful energy consumption results from transmitting/receiving data, querying requests, and forwarding data. Wasteful energy consumption is due to collisions and resulting retransmissions, idle listening to the channel, and overhead of each packet header (even when the data packet is short). Energy consumption reduces network lifetime, which is defined as the time elapsed until the first node (or a certain percentage of nodes [1]) use up their energy.

To reduce energy consumption, clustering techniques have been suggested [3-20]. These techniques organize the nodes into clusters where some nodes work as clusterheads and collect the data from other nodes in the clusters. Then, the heads can consolidate the data and send it to the data center as a single packet, thus reducing the overhead from data packet headers. Clustering has advantages for: 1) reducing useful energy consumption by improving bandwidth utilization (i.e., reducing collisions caused by contention for the channel); 2) reducing wasteful energy consumption by reducing overhead.

In a clustered network, the communication is divided into intra and inter cluster communication. The intra-cluster communication is from the nodes inside a cluster to the head. The inter-cluster communication is from the heads to the data center (sink node). The energy efficiency of a clustered sensor network depends on the selection of the heads. Heinzelman et al. [3] propose a low-energy adaptive clustering hierarchy (LEACH), which generates clusters based on the size of the sensor network. However, this approach needs a priori knowledge of the network topology. Younis and Fahmy [4] propose a Hybrid Energy-Efficient Distributed clustering (HEED), which creates distributed clusters without the size and density of the sensor network being known. However, the cluster topology fails to achieve minimum energy consumption in intra-cluster communication. Also, as we show in Section 5, the clusters generated by HEED are not well balanced.

In this paper, our goal is to achieve better cluster size balance and obtain clusters such that each has the minimum energy topology. We propose a distributed weight-based

energy-efficient hierarchical clustering protocol (DWEHC). DWEHC makes no assumptions on the size and the density of the network. Every node implements DWEHC individually. DWEHC ends after seven iterations that are implemented in a distributed manner. Once DWEHC is over, the resulting clusters have a hierarchical structure. Each node in the network is either a clusterhead, or a child (first level, second level, etc). The number of levels depends on the cluster range and the minimum energy path to the head. Within a cluster, TDMA (Time Division Multiple Access) is used. Each node responds to their nearest parent's polling with their data, then that parent is polled by the next parent until the data gets to the clusterhead. For inter-cluster communication, the heads contend for the channel using IEEE 802.11 to send data to the data center.

The paper is organized as follows: We review the literature in Section 2, and propose DWEHC in Section 3. In Section 4, we provide an analysis of correctness and energy-efficiency, complexity and scalability of DWEHC. In Section 5, we present our performance studies. Section 6 concludes the paper.

2 Related Work

Many clustering algorithms have been proposed [3-20]. Most of these algorithms are heuristic in nature and their aim is to generate the minimum number of clusters. In the Linked Cluster Algorithm [6], a node becomes the clusterhead if it has the highest id among all nodes within two hops. In updated LCA [7], those nodes with the smallest id become cluster head. All the other nodes which are 1-hop to the heads become children of the heads. In [8], those nodes with highest degree among their 1-hop neighbors become cluster heads. In [10], the authors propose two load balancing heuristics for mobile ad hoc networks, where one is similar to LCA and the other is degree-based algorithm. The Weighted Clustering Algorithm (WCA) [11] elects clusterheads based on the number of surrounding nodes, transmission power, battery-life and mobility rate of the node. WCA also restricts the number of nodes in a cluster so that the performance of the MAC protocol is not degraded. The Distributed Clustering Algorithm (DCA) uses weights to elect clusterheads [12]. These weights are based on the application and the highest weight node among its one hop neighbors is elected as the clusterhead. All of the above algorithms generate 1-hop clusters, require synchronized clocks and have a complexity of $O(n)$, where n is the number of sensor nodes. This makes them suitable only for networks with a small number of nodes.

The Max-Min d -cluster algorithm [5] generates d -hop clusters with a complexity of $O(d)$, which achieves better performance than the LCA without clock synchronization. In [13], the authors aim at maximizing the lifetime of a sensor network by determining optimal cluster size and assignment of clusterheads. They assume that both the number and the location of the clusterheads are known, which is generally not possible in all scenarios. McDonald et al. [14] propose a distributed clustering algorithm for mobile ad hoc networks that ensures that the probability of mutual reachability between any two nodes in a cluster is bounded over time. In [15], the authors generate a 2-level hierarchical telecommunication network in which the nodes at each level are distributed according to two independent homogeneous Poisson point processes and a node is connected to the closest node lying on the another level. Baccelli et al. extend previous study to hierarchical telecommunication networks with more than two levels in [16]. They use point processes and stochastic geometry to determine the average cost of connecting nodes for assigning them to multiple levels in the networks.

Heinzelman et al.[3] propose a low-energy adaptive clustering hierarchy (LEACH) for microsensor networks. LEACH uses probability to elect clusterheads. The remaining nodes join the clusterhead that requires minimum communication energy, thus forming a 1-hop cluster. LEACH also calculates the optimal number of clusterheads that minimizes the energy used in the 1-level network. Bandyopadhyay et al.[17] propose a hierarchical clustering algorithm to minimize the energy used in the network. They generate a hierarchical structure, which is up to 5-levels in intra-cluster communication. They assume all nodes transmit at the same power levels and hence have the same radio ranges. Based on the size of the network, they calculate the optimal number of clusterheads in a network and the optimal number of hops from the nodes to the clusterheads.

All the previous protocols require either knowledge of the network density or homogeneity of node dispersion in the field. Younis and Fahmy.[4] propose Hybrid Energy Efficient Distributed clustering (HEED). HEED does not make any assumptions about the network, such as, density and size. Every node runs HEED individually. At the end of the process, each node either becomes a clusterhead or a child of a clusterhead. Residual energy of a node is the first parameter in the election of a clusterhead, and the proximity to its neighbors or node degree is the second. HEED generates a 1-level hierarchical clustering structure for intra-cluster communication. In DWEHC, we do not make any assumptions about the network similar to HEED. DWEHC creates a multi-level structure for intra-cluster communication, which uses the minimum energy topologies.

3 A Distributed, Weighted, Energy-Efficient Hierarchical Clustering Algorithm

3.1 Network Model

We assume that a sensor network can be composed of thousands of nodes. Also:

- 1) Nodes are dispersed in a 2-dimensional space and cannot be recharged after deployment.
- 2) Nodes are quasi-stationary.
- 3) Nodes transmit at the same fixed power levels, which is dependent on the transmission distance.
- 4) Nodes base decisions on local information.
- 5) Nodes are location-aware, which can be defined using GPS, signal strength or direction.
- 6) The energy consumption among nodes is not uniform.

We do not make any assumptions about:

- 1) the size and density of the network;
- 2) the distribution of the nodes;
- 3) the distribution of energy consumption among nodes;
- 4) the probability of a node becoming a clusterhead;
- 5) the synchronization of the network.

We believe this model and these assumptions are appropriate for many real networks. In a sensor network, sensor nodes collect their local information and send them to the data center. Frequently, the information is location-dependent, so the nodes know their own position via GPS or by other means. On the other hand, density is not uniform or known.

3.2 Clustering Structure

After running DWEHC, a clustered network has the features:

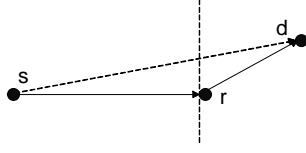


Figure 1 Relaying through node r

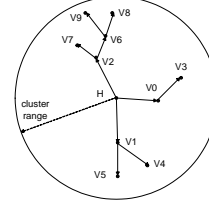


Figure 2 Levels in DWEHC

- 1) A node is either a clusterhead or a child in the cluster. The level of the node depends on the cluster range and the minimum energy path to the head.
- 2) Heads are well distributed over the sensor field.
- 3) Each cluster has a minimum energy topology.
- 4) A parent node has a limited number of children.

3.3 Related Concepts

We will introduce some concepts used in DWEHC. As shown by Li and Wan [2], given a set of wireless nodes, and a directed weighted transmission graph, there exists the minimum power topology. This topology is the smallest subgraph of the transmission graph and contains the shortest paths between all pairs of nodes. Li and Wan [2] propose a distributed protocol to construct an enclosure graph which is an approximation of the minimum power topology for the *entire network*. In DWEHC, we generate a minimum power topology for each *cluster*, which is an enclosure graph, using their protocol. As shown in [2], enclosure graph is a planar graph and the average number of edges incident to a node is at most 6.

1. Relay

In this paper, we assume that all mobile devices have similar antenna heights as [2], so we will only concentrate on path loss that is distance-dependent. With this assumption, the power required by distance r is r^α , which is called the transmitter power of a node s . For example, in Figure 1, the node s tries to send a packet to d . Node s can send the packet directly with transmission power $\|sd\|^\alpha + c$, where α is equal to 2 or 4 and c is a constant. Or, s sends the packet through r with transmission power $\|sr\|^\alpha + c + \|rd\|^\alpha + c$ which is called relay. If $\|sd\|^\alpha + c > \|sr\|^\alpha + c + \|rd\|^\alpha + c$, then relaying through node r consumes less transmission energy than directly transmitting from s to d .

2. Relay Region [2]

Given sender node s and relay node r , the nodes in the relay region can be reached with the least energy by relaying through r . And, the region $R_{\alpha,c}(s,r)$, is called the relay region of s with respect to r .

$$R_{\alpha,c}(s,r) = \{x | \text{such that } \|sx\|^\alpha > \|sr\|^\alpha + \|rx\|^\alpha + c\} \quad (1)$$

3. Enclosure Region [2]

Enclosure region of s with respect to a node r , $E_{\alpha,c}(s,r)$, is the complement of $R_{\alpha,c}(s,r)$. The enclosure region $E_{\alpha,c}(s)$ of a node s is defined in Equation 2, where $T(s)$ is the set of nodes lying inside the transmission range of node s .

Notations:

$N(s)$: the neighbors of node s ;

$T(s)$: the nodes inside cluster range of node s ;

Algorithm:

1. $N(s) = \phi$; $Q = T(s)$

2. while ($Q \neq \phi$) {

2.a. Let $v \in Q$ and be the nearest node to s

2.b. $N(s) = N(s) \cup \{v\}$

2.c. Eliminate all nodes x from Q , such that

$$\|sv\|^\alpha + \|vx\|^\alpha + c < \|sx\|^\alpha \}$$

Figure 3 Finding Neighbors

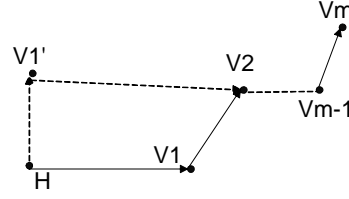


Figure 5 HV1V2 is a minimum energy path

$$E_{\alpha,c}(s) = \bigcap_{r \in T(s)} E_{\alpha,c}(s,r) \quad (2)$$

4. Neighbors [2]

Equation 3 defines the set of neighbors $N_{\alpha,c}(s)$ of a node s . These are the nodes which do not need relaying when s transmits to them,

$$N_{\alpha,c}(s) = \{u | \text{such that } u \in T(s), u \in E_{\alpha,c}(s)\} \quad (3)$$

5. Cluster range (cluster radius), R

This parameter specifies the radius of a cluster, i.e., the farthest a node inside a cluster can be from the clusterhead. The cluster radius is a system parameter and is fixed for the entire network.

6. Weight used in clusterhead election

The weight will be the only locally calculated parameter used in clusterhead election, and is represented by *my_weight* in DWEHC. We define it in Equation 4, where R is the cluster range and d is the distance from node s to neighboring node u , $E_{residual}(s)$ is the residual energy in node s , and $E_{initial}(s)$ is the initial energy in node s , which is the same for all nodes.

$$W_{weight}(s) = \left(\sum_{u \in N_{\alpha,c}(s)} \frac{(R-d)}{6R} \right) \times \frac{E_{residual}(s)}{E_{initial}(s)} \quad (4)$$

Our primary goal is to improve energy efficiency and prolong the lifetime of the network. Since a clusterhead needs to forward all data from the nodes inside its cluster to the data center, the clusterhead will consume much more energy than its child nodes. Therefore, the residual energy is a key measurement in electing clusterheads. On the other hand, the more nodes inside a cluster, the more energy the clusterhead will consume. We build the clusters by considering the neighboring nodes, of which there are at most 6 [2]. Each parent node has a limited number of child nodes. Additionally, intra-cluster communication will consume less energy if the cluster is a minimum energy topology. Therefore, the distances from a node to all its neighbors inside its enclosure region becomes another key measurement in electing clusterheads.

7. Levels in a cluster

Each cluster is multi-level. There is no optimal number of levels. This is because we make no assumptions about the size and topology of the network. The number of levels in a cluster depends on the cluster range and the minimum energy path to the clusterhead, which is

represented by my_level in DWEHC. Figure 2 shows a multi-level cluster generated by DWEHC, where H is the clusterhead, first level children are v_0 , v_1 , and v_2 , second level children are v_3 , v_4 , v_5 , v_6 and v_7 , and the third level children are v_8 and v_9 . A parent node and its child nodes are neighbors. For example, H 's neighbors are v_0 , v_1 , and v_2 .

8. my_range and my_dis

In a cluster, each child node should be inside the cluster range of its clusterhead. my_range is the distance (in Figure 2) from v_3 to H and calculated $\sqrt{(x_H - x_{v_3})^2 + (y_H - y_{v_3})^2}$, where (x_H, y_H) and (x_{v_3}, y_{v_3}) are the (x, y) coordinates for H and v_3 respectively. my_dis denotes the minimum energy path to the clusterhead. In Figure 2, my_dis of v_3 is calculated by $(x_H - x_{v_0})^2 + (y_H - y_{v_0})^2 + (x_{v_0} - x_{v_3})^2 + (y_{v_0} - y_{v_3})^2$.

3.4 The DWEHC Algorithm

Before the communications may start, the clusters need to be generated. It takes $T_{generating}$ to generate the clusters. Then, two types of communication may occur in a clustered network: intra-cluster and inter-cluster. The time committed for these is $T_{cluster}$. $T_{cluster}$ should be much longer than $T_{generating}$ to guarantee good performance. To prevent a clusterhead from dying due to energy loss, the DWEHC algorithm, runs periodically, every $T_{cluster} + T_{generating}$.

During $T_{generating}$, each node runs DWEHC to generate the clusters. During initialization, a node broadcasts its (x, y) coordinates and then uses the algorithm in Figure 3 to find its neighbors[2]. The complexity of a node u finding its neighbors is $O(\min(d_{G_t}(u)d_{G_e}(u), d_{G_t}(u)\log d_{G_t}(u)))$, where $d_{G_t}(u)$ is the degree of node u in its cluster range and $d_{G_e}(u)$ is the degree of node u in its enclosure region [2]. The DWEHC algorithm is Figure 4.

After running the algorithm, each node will have $N(s)$, the set of neighbors inside its enclosure region, and these neighbors will be its first level children if it becomes a clusterhead. All the other nodes inside its cluster range will be reached through at least one relay using one of these neighbors. All nodes set their my_level to -1 in the beginning, which indicates that they have not joined any cluster yet. Next, we explain how a node either becomes a clusterhead or a child in a cluster.

Clusterhead: A node that has the largest weight of all its neighbors will become a temporary clusterhead ($temp_head=1$). A node can become a real clusterhead only if a given percentage of its neighbors elect it as their temporary clusterhead. In the first iteration ($i=0$), this percentage is 100%. In subsequent iterations ($i++$), it is decreased to $(6-i)/6$. When a node becomes a real clusterhead, it sets $my_level=0$. Those nodes with $my_level = -1$ could still become clusterheads during the following iterations. Those who become real clusterheads broadcast the information including their my_level , x ($head_x$), and y ($head_y$) coordinates.

Non Clusterhead: A node will become a child node in the following three cases. The first case occurs when a node's my_level is equal to -1. The node receives a broadcast message from its neighbors, which includes my_level of the neighbor and the x and y coordinates of a clusterhead ($head_x$ and $head_y$). If the distance from the clusterhead to the node is less than or equal to the cluster range, then the node chooses the clusterhead as its cluster-

Notations:

my_id, *my_x*, *my_y*: the id, x and y coordinates of a node;
my_level, *my_weight*: the level and the weight of a node; *my_range*: the distance to the head;
my_dis: the minimum energy distance to the head;
my_temp_head: 1 (if a node is a temporary head); 0 (o.w.)
my_head_num: the temporary head id in the neighborhood;
my_per: percentage of neighbors choosing the node as their temporary head;
temp_head_id: the id of a temporary head;
head_x, *head_y*: the x and y coordinate of a head
my_dir_parent: the id of my directly parent node;
MAX: the maximum number of neighbors (6);

Algorithm:**1. Initialization**

- 1.1. broadcast x and y coordinates;
- 1.2. collect broadcasts inside cluster range;
- 1.3. find neighbors inside own enclosure region using Fig 3;
- 1.4. calculate *my_weight* and broadcast it;

2. Cluster Generation

FOR (*i*=0; *i*<MAX; *i*++) {

2.1. IF *my_level* = -1 {

2.1.a. IF *my_weight* is largest among my neighbors then {

my_temp_head = 1;
my_head_num = *my_id*; }ELSE {
my_temp_head = 0; }

my_head_num = *temp_head_id*; } //use neighbor's with the largest weight

2.1.b. broadcast *my_head_num*;

2.1.c. IF ((*my_per* $\geq \frac{MAX-i}{MAX}$ AND *my_temp_head* = 1) OR

(no neighbors) OR (all neighbor's *my_level* > -1)) {

my_level = *my_dis*=0; *head_x* = *my_x*; *head_y* = *my_y*; //become a real cluster head
broadcast *my_level*, *my_dis*,*head_x*, *head_y*; } //end 2.1.c

} //end 2.1

2.2. IF(*my_level* <>0) {

2.2.a. receives broadcast message from a neighbor;

2.2.b. $my_range = \sqrt{(head_x - my_x)^2 + (head_y - my_y)^2}$;

my_dis_new = neighbor's *my_dis* + (distance to the neighbor)²

2.2.c. IF ((*my_dis_new* < *my_dis* AND *my_level* > 0) OR (*my_range* <= Cluster Range AND *my_level* = -1)) {

my_level = neighbor's *my_level* + 1; *my_dis* = *my_dis_new*; *my_dir_parent* = neighbor's id;
head_x = neighbor's *head_x*; *head_y* = neighbor's *head_y*;
broadcast *my_level*, *my_dis*,*head_x*, *head_y*; } //end 2.2

} //end 2

3. Finalization

Repeat cluster generation one more time.

Figure 4 DWEHC

head and sets its *my_level* to *my_level* from the broadcast message plus one, and its *my_dis* to *my_dis* from the broadcast message plus its distance to the neighbor. The second case is when a node's *my_level* is not equal to -1. This indicates that it has already chosen its clusterhead. If the neighbor who sends the broadcast has a different clusterhead, and the distance from the node to the neighbor's clusterhead is inside the cluster range, and the new calculated *my_dis* is less than the previous *my_dis* from the node to its current clusterhead;

then the node will choose the neighbor's clusterhead as its clusterhead and its *my_level* will be changed to current neighbor's level plus one. The third case occurs when a node's *my_level* is not equal to -1. The node receives a broadcast from its neighbor. If the neighbor has the same clusterhead as the node, and the *my_dis* from the node to this neighbor is less than the previous *my_dis*, then the node will choose the neighbor as its parent and reset its *my_level* and *my_dis* as in the second case.

Iteration: The cluster generating process runs at most seven times (including finalization) since each node has at most six neighbors[2]. After running DWEHC, a node either becomes a clusterhead or becomes a child in a cluster, and its level in the cluster is represented by *my_level*.

4 Analysis of DWEHC

4.1 Intra-cluster Communication

Intra-cluster communication is contentionless using TDMA. Each parent node polls its direct children and forwards the data to its parent node until the data reaches the clusterhead. The parent node may combine several data packets from its children together with its own data into one packet.

Correctness and Energy Efficiency: DWEHC is completely distributed on the whole network. Each node is either a clusterhead or a child node in a cluster. Each cluster contains the minimum-power topology, which is locally optimal. Lemmas 1-4 prove these statements.

Lemma 1: After running DWEHC, a node is either a clusterhead or a child in a cluster.

Proof. Assume a node *A* is neither a clusterhead nor a child after running DWEHC. Node *A* can only have two conditions before running DWEHC: 1) *A* does not have neighbors; 2) *A* has neighbors.

In the first case, since *A* does not have any neighbor, it will become a clusterhead in step 2.1.c of cluster generation, which contradicts the assumption. In the second case, since *A* is neither a clusterhead nor a child, this indicates that none of the *A*'s neighbors have become clusterheads. If one of them were a clusterhead, *A* would have to be a first level child. If *A* and some of its neighbors do not join any clusters, then, one of them will become a clusterhead, which will satisfy one of the conditions to become a clusterhead in 2.1.c of DWEHC. *A* will then become a child node. Therefore, all *A*'s neighbors should join other clusters before finalization of DWEHC. *A* is the only one which does not join any clusters at most after DWEHC's six iterations of cluster generation. In the finalization of DWEHC, *A* will become a clusterhead, which contradicts the assumption.

Lemma 2: A node is covered by only one clusterhead.

Proof. A node *A* will set its *my_level* to only one value based on the distance to the clusterhead (inside the clusterhead's cluster range) and its *my_dis* variable. Therefore, *A* will only belong to one clusterhead.

Lemma 3: DWEHC distributes the clusterheads well, i.e., when two nodes are within each other's cluster range, the probability of both of them becoming clusterheads is very small.

Proof. Omitted because of space limitations.

Lemma 4: DWEHC generates each cluster with the minimum energy topology.

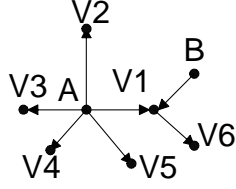


Figure 6 Node A and its neighbors from V1 to V5

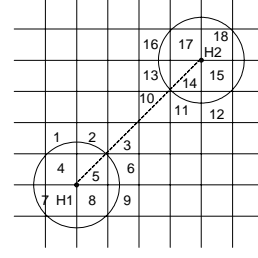


Figure 7 The longest distance between two clusterheads

Proof. Let Figure 5 show a cluster generated by DWEHC. Let H be the clusterhead. V_1 is the child of H , V_2 is the child of V_1 , and V_m is the child of V_{m-1} . Path HV_1 should be on the minimum energy path from H to V_1 since V_1 is a neighbor of H (using the definition of *neighbor* from Section 3.3). Similarly, the path V_1V_2 should be on the minimum energy path from V_1 to V_2 (V_2 is a neighbor of V_1).

Let us assume that HV_1V_2 is not a minimum energy path, which indicates that there should exist another path $HV_1'V_2$, where v_1' is a neighbor of H , V_2 is a neighbor of v_1' and $\|HV_1'\|^\alpha + \|v_1'V_2\|^\alpha < \|HV_1\|^\alpha + \|V_1V_2\|^\alpha$. If such a node v_1' existed, then V_2 would have chosen v_1' to be its parent in step 2.2.c of DWEHC cluster generation. Therefore, v_1' does not exist. This contradicts the assumption, thus, HV_1V_2 is a minimum energy path. The same argument can be made to prove that $HV_1V_2\dots V_{m-1}V_m$ is the minimum energy path. Therefore, DWEHC provides the minimum energy topology inside each cluster.

Complexity: DWEHC generates clusters in at most seven iterations. Each node sends only $O(1)$ broadcast messages. Lemmas 5-7 prove these statements.

Lemma 5. The complexity of broadcast message exchange is $O(1)$ for each node.

Proof. In $T_{generating}$, during initialization, each node broadcasts two messages: 1) a broadcast message of its coordinates; 2) a broadcast message including its weight. During cluster generation, each node will announce information in two broadcast messages: 1) *my_head_num*; and 2) *my_level*, *my_dis*, *head_x*, and *head_y* variables after joining a cluster. A node which becomes a clusterhead only needs to send two messages. As for the non clusterhead nodes, the minimum number of broadcast messages exchanged is two and the maximum is twelve (changing its *my_level* in each iteration). During finalization, a node with *my_level*=-1 broadcasts the same variables as in cluster generation. Therefore, the minimum number of exchanged broadcast messages is four, and the maximum is fourteen. Therefore, the complexity of broadcast message exchange is $O(1)$ for each node.

Lemma 6. The complexity of a node becoming a clusterhead is $O(1)$.

Proof. We will show that the complexity of a node becoming a clusterhead is $O(1)$. Note that, only a temporary clusterhead will become a clusterhead. Suppose A is a temporary clusterhead and nodes from V_1 to V_5 are its neighbors as shown in Figure 6. We will discuss two cases: A has neighbors which choose it as their temporary clusterhead (we call

this set neighbor1) and neighbors that do not choose it (we call this set neighbor2). If at least one node is in neighbor1 and remains there, then A becomes a clusterhead by the sixth iteration. Since some neighbors in neighbor2 may join other clusters, the percentage of remaining nodes choosing A increases and A will become a clusterhead even sooner.

In the second case, A only has neighbors in neighbor2. In this case, all the nodes in neighbor2 may choose other nodes as their temporary clusterhead. For example, V_I may choose B to be its temporary clusterhead. B could be a temporary head or not. If B is already a temporary head, then B could become a head for the same reason as A becoming head in the previous case. If B is not a temporary head and B does not join any clusters, there are two possibilities. 1) B will become a temporary head with neighbor V_I . If B becomes a head, then V_I joins it or not. 2) B never becomes a temporary head with V_I or not. Then there must exist another temporary head which will become a head and B joins that cluster. The same is valid for the other nodes. Therefore, if A does not join any clusters, then A itself will become a head by the time DWEHC is finalized.

Scalability: Each parent node has a limited number of child nodes. This is important in terms of scalability. DWEHC achieves good load balance per node, prolonging the life time of a clusterhead.

4.2 Inter-Cluster Communication

Inter-cluster communication is contention based. The clusterheads poll their first level children, include their own data and transmit to the data center. The clusterheads consolidate several data packets into one data packet thus reducing overhead. Next, we prove that DWEHC provides end-to-end connectivity in the network.

To find out the conditions under which the clusters generated by DWEHC are *asymptotically almost surely* (a.a.s) connected, let us consider a sufficiently large network $R = [0, L]^2$, which is divided into square cells, with side $R_c/\sqrt{2}$, where R_c is the cluster range. This is because all the nodes within the same cluster should be reachable from each other and the highest two nodes should be at the end of the diagonal. We will show DWEHC ensures connectivity a.a.s when the clusterheads transmitting range, R_t , is greater than or equal to $4R_c$.

Lemma 7: Assume that n nodes, each with transmitting range R_c , are distributed uniformly, independently, and randomly in $R = [0, L]^2$ where R is a 2 dimensional plane, and assume that the area is divided into square cells of size $(R_c/\sqrt{2}) \times (R_c/\sqrt{2})$. If $R_c^2 N = kL^2(\ln L)$, for some $k > 0$, then each cell contains at least one node a.a.s.

Proof. Our proof is similar to [1], so, we omit it.

Lemma 8: When Lemma 7 holds, two clusterheads, H_1 and H_2 , can communicate if $R_t \geq 4R_c$, where R_t is the minimum transmitting range.

Proof. Our proof is similar to that in [18]. In Figure 7, we show a boundary case, where H_1 and H_2 are heads. DWEHC generates clusters with the largest radius, R_c (R_c is the cluster range). With lemma 3 holding, there is no overlap between the clusters generated by DWEHC. With lemma 7 holding, there is at least one node in each square cell size $R_c/\sqrt{2}$. To cover cell 3, H_1 can be any position of cell 5. The farthest position for covering

cell 3 is the H_1 position shown in Figure 7. To cover cell 10, H_2 can be in any position in cell 14, and the H_2 position shown is the farthest position. Thus, the distance between H_1 and H_2 is the farthest distance between any two clusterheads. The distance between clusterheads H_1 and H_2 is $4R_c$, which is the minimum transmission range for H_1 to reach H_2 .

Lemma 9: DWEHC generates multi-hop clusters a.a.s.

Proof. Omitted because of space limitations.

5 Simulation

In this section, we will evaluate the performance of DWEHC via simulation. We ran simulations with 300 and 1000 sensor nodes, which are randomly dispersed into a field with dimensions 1000 by 1000 meters and 2000 by 2000 meters. Simulations with 300 nodes are run for cluster ranges of 75, 100, 150, 200, 250, 300 meters. Those for 1000 nodes are run for cluster ranges of 100, 150, 200, 250, 300, 350, 400 meters. In each simulation, we randomly initialize the nodes' residual energy and generate the topology. Each result represents the average of 20 simulation runs with the same parameters.

Wireless transmission laws dictate that power attenuation be proportional to the square of the covered distance (assuming fixed transmission power). If the distances are small (up to hundreds of meters), then the power attenuation can be assumed to be linear [19]. Other factors may also affect the received power, such as combined noise or physical obstacles. For simplicity, we ignore all these factors in our simulations[4], therefore, we consider the distance between two nodes as the only requirement to decide the transmission power.

In the simulations, we will compare DWEHC with HEED-AMRP[4]. HEED-AMRP (average minimum reachability power) considers the average minimum power levels required by the nodes within the cluster range of a node as the second parameter besides using residual energy as the first parameter to elect clusterheads. Both DWEHC and HEED-AMRP consider minimum power levels in the protocol design with no assumption about the size and density of the network. The simulation results will be discussed in two parts. In the first part, we compare DWEHC with HEED-AMRP with respect to cluster characteristics. In the second part, we will compare their energy usage and network lifetime.

5.1 Cluster Characteristics

In this section, we will compare DWEHC with HEED-AMRP in number of iterations to terminate, and the features of clusterheads, such as the number of clusterheads, the number of single node clusterheads, the maximum number of nodes in a cluster, and the distribution of nodes in clusters. The clustering code is written in C and all nodes run DWEHC synchronously. (Synchronization improves performance but is not necessary for correctness.)

a) Iterations to Terminate

In HEED-AMRP, we initialize both CH_{prob} and C_{prob} to be 0.05 (same as the simulations in [4]), where CH_{prob} is used to decide the probability to be a clusterhead, and C_{prob} is used to initialize the residual energy. In each iteration, CH_{prob} is multiplied by two. A node terminates when CH_{prob} reaches 1. So, HEED-AMRP takes six iterations to terminate. In DWEHC, we use weight as the parameter to elect clusterheads (see details about weight calculation in section 3.3). Theoretically, DWEHC will take at most seven iterations (see

Table 1: # of clusterheads (300)

Cluster range	75	100	150	200	250	300
DWEHC	57	30	19	16	12	8
HEED-AMRP	81	54	29	18	13	11

Table 2: # of single node cluster (300)

Cluster range	75	100	150	200	250	300
DWEHC	7	3	0	0	0	0
HEED-AMRP	14	8	4	1	0	0

Table 3: Max # of nodes in clusters (300)

Cluster range	75	100	150	200	250	300
DWEHC	8	12	23	28	39	53
HEED-AMRP	11	15	24	40	43	78

Table 4: # of clusters with size 75 (300)

Number of nodes	<=5	<=10	>10
DWEHC	42	15	0
HEED-AMRP	65	15	1

proof in lemma 4). In fact, all simulations took at most three iterations to terminate. So, we can see DWEHC will use less time, $T_{generating}$ in generating clusters .

b) Clusterhead Characteristics

1) With 300 nodes

In this section, all the simulations are done on a sensor network with 300 nodes. Table 1 shows the number of clusters for various cluster ranges. We see that HEED-AMRP generates significantly more clusters than DWEHC when the cluster range varies from 75 to 150. Since more clusters means more interference between the clusterheads and more overhead, HEED-AMRP will consume more energy than DWEHC. From 200 to 300, HEED-AMRP produces slightly more clusterheads. The performance also depends on the number of nodes inside a cluster, which we will show in Table 4 and 5.

Table 2 shows the number of single node clusters. Single node clusters are not desirable. When the cluster range varies from 75 to 100m, HEED-AMRP generates more than twice as many single node clusters as DWEHC. Starting at 150, DWEHC does not generate single node clusters, but HEED-AMRP has 4 and 1 when the cluster range is 150 and 200, respectively. Single node clusters disrupt the load balance. So, we expect the clusters generated by DWEHC to have better load balance than the clusters generated by HEED-AMRP.

Table 3 shows the maximum number of nodes in a cluster. The maximum number of nodes in a cluster in DWEHC is less than that of HEED-AMRP. This is because each parent node has a limited number of child nodes and a node could join a cluster only when at least one of its neighbors are inside the cluster. Therefore, DWEHC can achieve load balance. HEED-AMRP may cause many nodes to join a cluster which causes the clusterhead to deplete its energy quickly. For example, using DWEHC, the maximum number of nodes is 53 when the cluster range is 300. However, HEED-AMRP generates a 78-node cluster at this cluster range. Table 4 shows the number of clusters (distribution of nodes) with different cluster size (the cluster range is 75m). DWEHC has 42 clusters which have 5 or less, and 15 clusters which have 10 or fewer nodes. However, HEED-AMRP generates more clusters (65) which have 5 or less nodes. Note that, if child nodes send data to the clusterhead frequently, the intra-cluster communication will consist of many short data packets. Thus, HEED-AMRP will cause more overhead.

Table 5 shows the number of clusters with different cluster size (the cluster range is 300m). For example, when the number of nodes is less than or equal to 10 in a cluster, DWEHC has two such clusters and HEED-AMRP has 3 such clusters. HEED-AMRP also

Table 5: # of clusters with size 300 (300)

# of nodes	10	15	20	25	30	35	40	45	50	55	=78
DWEHC	2	2	0	1	1	0	0	1	0	1	0
HEED-AMRP	3	2	0	1	1	0	1	0	2	0	1

Table 6: # of clusterheads (1000)

Cluster range	100	150	200	250	300	350	400
DWEHC	125	71	60	58	50	45	22
HEED-AMRP	203	111	66	43	32	27	19

Table 7: # of single node clusters (1000)

Cluster range	100	150	200	250	300	350	400
DWEHC	10	2	0	0	0	0	0
HEED-AMRP	22	10	2	0	1	0	0

Table 8:Max # of nodes in clusters (1000)

Cluster range	100	150	200	250	300	350	400
DWEHC	12	20	30	42	42	49	71
HEED-AMRP	15	30	48	85	97	97	100

Table 9:# of clusters with size 100 (1000)

Number of nodes	<=5	<=10	<=15
DWEHC	74	46	5
HEED-AMRP	121	72	10

Table 10: # of clusters with size 400 (1000)

Number of nodes	<=40	<=60	<=80	<=100
DWEHC	9	9	4	0
HEED-AMRP	8	5	4	2

generates a cluster with 78 nodes but DWEHC generates one cluster with the maximum number of nodes to 53 (see Table 3). From the previous tables, we can see DWEHC realizes better load balance.

2) With 1000 nodes

In this section, all the simulations are done on a sensor network with 1000 nodes. Table 6 shows the number of clusterheads when cluster range is between 100 and 400m. DWEHC generates fewer clusters than HEED-AMRP when the cluster range varies from 100 to 200. This is because DWEHC realizes better clusterhead distribution than HEED-AMRP does and has fewer single node clusters. Starting at cluster range 250, HEED-AMRP generates fewer clusters than DWEHC. The reason is each parent node in DWEHC has a limited number of nodes (see lemma 7) in a cluster which is determined by the cluster range and the my_dis (see details in section 3.3) variable of a node. In HEED-AMRP, all nodes inside a clusterhead's range can join the cluster without the limited number of nodes. Therefore, DWEHC has better load balance than HEED-AMRP. Table 7 shows the number of single node clusters when the cluster range varies from 100 to 400. HEED-AMRP generates many more single node clusters than DWEHC. Table 8 shows the maximum number of nodes in clusters. Similar to the 300-node network, HEED-AMRP always generates more nodes in a cluster, which will make the clusterhead fail sooner. Table 9 shows the number of clusters with different cluster sizes (the cluster range is 100). For example, DWEHC generates 74 clusters with the number of nodes less than or equal to 5, however, HEED-AMRP generates 121 such clusters, where 22 of those clusters are single node clusters (as shown in Table 7). Therefore, DWEHC achieves better load balance than HEED-AMRP. Table 10 shows the number of clusters with different cluster size (cluster range 400). Here, DWEHC achieves better load balance than HEED-AMRP. Since HEED-AMRP has two clusters which have 100 nodes, the clusterheads will consume more energy in transmissions than other clusters which have fewer nodes.

5.2 Simulation of Clustering Applications

The simulation results are implemented by using ns[20]. Table 11 shows the parameters used in simulations. We have implemented two sets of simulations, one with 300 nodes in one square kilometer, and the other with 1000 nodes in a four square kilometer area. The sink node (i.e., the data center) is placed in the middle.

The following parameters are the same as those in [4]. In the simple radio model that we use, energy expenditure is due to: 1) digital electronics, E_{elec} , (actuation, sensing, signal emission/reception), and 2) communication, E_{amp} . In our model, E_{amp} varies according to the distance d between a sender and a receiver where $E_{amp} = \epsilon_{fs}$ assuming a free space model when $d < d_0$, while $E_{amp} = \epsilon_{mp}$, assuming a multipath model when $d \geq d_0$, where d_0 is a constant distance (i.e., Threshold) that depends on the environment. To transmit n_b bits for a distance d , the radio expends $n_b(E_{elec} + E_{amp} \times d^n)$ J, where $n=2$ for $d < d_0$, and $n=4$ for $d \geq d_0$. To receive n_b bits at the receiver, the radio expends $n_b \times E_{elec}$ J. This energy model assumes a continuous function for energy consumption.

Table 11: Simulation Parameters

Parameter	Simulation1	Simulation2
Number of Nodes	300	1000
Network grid	(0,0), (1000,1000)	(0,0), (2000,2000)
Sink (data center)	(500, 500)	(1000, 1000)
Threshold distance	75m	75m
Cluster range	75,100, ..., 300	100,150, ..., 400
E_{elec}	50 nJ/bit	50 nJ/bit
ϵ_{fs}	10pJ/bit/ m^2	10pJ/bit/ m^2
ϵ_{mp}	0.0013 pJ/bit/ m^4	0.0013 pJ/bit/ m^4
E_{fusion}	5nJ/bit/signal	5nJ/bit/signal
Data packet size	100 bytes	100 bytes
Broadcast packet	25 bytes	25 bytes
Packet header size	25 bytes	25 bytes
Round ($T_{cluster}$)	300 TDMA frames	1000 TDMA frames
Initial energy	10 J/battery	10 J/battery

In DWEHC, a data packet size is 100 bytes. A parent non clusterhead node polls its children. The children respond to the polling with a data packet of 100 bytes. The parent node responds to the polling from its parent node with a data packet and the size can be up to 800 bytes. The clusterheads send data packets to the data center with the data packet size up to 800 bytes also. Clusterheads in both HEED-AMRP and DWEHC send 800 byte-long data packets. In our simulations, a node always responds to a polling. The clusters are regenerated every 300 TDMA frames ($T_{cluster}=300$) in the 300-node network. In the 1000-node network, the clusters are regenerated every 1000 TDMA frames ($T_{cluster}=1000$). In a real application, $T_{cluster}$ should be high enough to avoid re-clustering too often.

Next, we show the energy consumption in intra-cluster, inter-cluster communications in a $T_{cluster}$ and the number of rounds until the first node dies. A node is considered dead when 99% of its energy is used up.

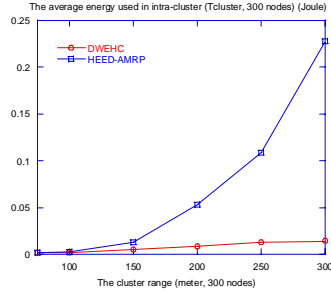


Figure 8 Average energy used for intra-cluster (300)

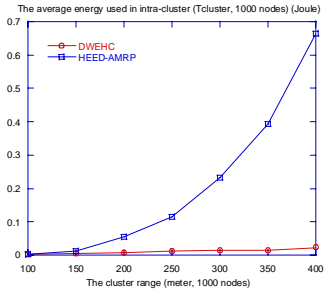


Figure 9 Average energy used for intra-cluster (1000)

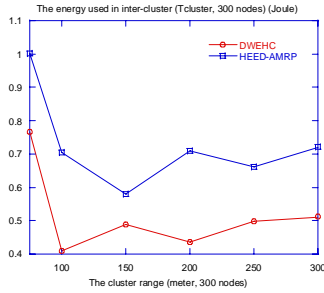


Figure 10 Energy used for inter-cluster (300)

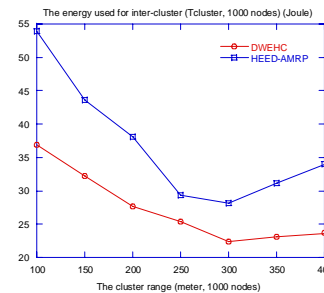


Figure 11 Energy used for inter-cluster (1000)

a) Intra-Cluster Communication

Figure 8 shows the average energy spent per node for intra-cluster communication in the network with 300 nodes in time $T_{cluster}$. HEED-AMRP consumes more energy than DWEHC, and especially as the cluster range is increased. Even at small cluster range, DWEHC saves energy. Since in HEED-AMRP the nodes communicate directly with the clusterhead, when cluster range is increased, so will the distance from the senders to the clusterheads. This will cause the senders to consume more energy. In DWEHC, senders relay messages through their parent, achieving optimal energy consumption within a cluster. For example, DWEHC consumes 82.2% and 65.7% of the energy consumed by HEED-AMRP when the cluster range is 75 and 100 respectively.

Figure 9 shows the average energy spent per node in intra-cluster communication with 1000 nodes in time $T_{cluster}$. The results are similar to the 300-node network. HEED-AMRP consumes significantly more energy than DWEHC as the cluster range is increased. For example, DWEHC consumes 68.4% and 39.1% of the energy consumed by HEED-AMRP when the cluster range is 100 and 150 respectively.

b) Inter-Cluster Communication

Figure 10 shows the energy consumed in inter-cluster communication in time $T_{cluster}$ (simulation1) when the total number of nodes is 300. The x_axis is the cluster range and y_axis is the energy consumed for inter-cluster communication in time $T_{cluster}$. HEED-

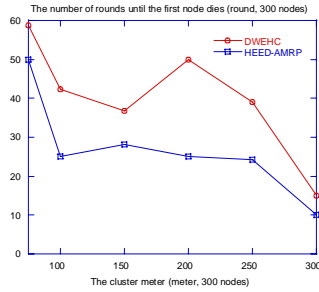


Figure 12 # of rounds until the first node dies (300)

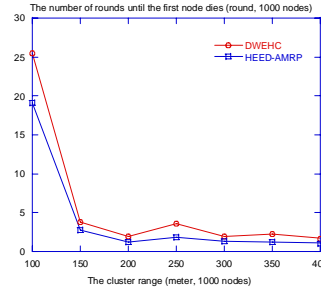


Figure 13 # of rounds until the first node dies(1000)

AMRP consumes more energy than DWEHC does. This is because DWEHC achieves better distribution than HEED-AMRP. As shown in Table 1 and Table 2, HEED-AMRP generates more clusterheads especially single node clusterheads than DWEHC. More clusterheads will cause more contention between clusterheads during transmissions from clusterheads to data center, and more overhead because fewer data packets can be consolidated by the clusterheads. When the cluster range is 75, both DWEHC and HEED-AMRP reach the highest in their energy consumption. This is because they both have the maximum number of clusterheads. With the increase of cluster range, the number of clusterheads decreases. The interferences caused by the contentions between clusterheads will decrease. But, the energy consumption is also related to the distances from the clusterheads to the data center. So, in DWEHC, energy consumed with cluster range of 200 is larger than the energy consumed with cluster range of 150. The same condition occurs at HEED-AMRP. Figure 11 shows the energy consumed in inter-cluster communication in time $T_{cluster}$ (simulation2) when the total number of nodes is 1000. The results are similar to the previous one. DWEHC consumes less energy in inter-cluster communication. Both of them consume the maximum energy when cluster range is 100 since both of them generate the largest number of clusterheads at cluster range 100.

c) The number of rounds until the first node dies

Figure 12 shows the number of rounds of $T_{cluster}$ until the first node dies when the total number of nodes is 300 in simulation1. The x_axis is the cluster range and y_axis is number of rounds until the first node dies. DWEHC lasts much longer than HEED-AMRP, except it is close to HEED-AMRP when the cluster range is 75 and 300 where it is still 20.51% and 28.25% longer respectively. Although both DWEHC and HEED-AMRP consume the most energy at cluster range of 75 (see Figure 10), both last the longest at this cluster range. This is because they contain many single node clusters, which do not have much data to send. Figure 13 shows the number of rounds of $T_{cluster}$ until the first node dies when the total number of nodes is 1000 in simulation2. The results are similar to the 300 nodes network. DWEHC lasts much longer than HEED-AMRP does. They are closest when the cluster range is 100, where DWEHC lasts 23.4% longer than HEED-AMRP.

6 Conclusions

In this paper, we proposed a distributed, weight-based Energy-Efficient Hierarchical Clustering algorithm (DWEHC). DWEHC operates with only realistic assumptions. The algo-

rithm constructs multilevel clusters and the nodes in each cluster reach the clusterhead by relaying through other nodes. DWEHC is well-distributed, and runs in $O(I)$ time which is a major advantage in a power-constrained sensor network. Our simulations demonstrated that DWEHC generates well balanced clusters. Both intra-cluster and inter-cluster energy consumption is greatly improved over clusters generated by the HEED-AMRP algorithm.

References:

- [1] D. M. Blough and P. Santi, "Investigating Upper Bounds on Network Lifetime Extension for Cell-Based Energy Conservation Techniques in Stationary Ad Hoc Networks", in Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), 2002.
- [2] X.-Y. Li, and P.-J. Wan, "Constructing Minimum Energy Mobile Wireless Networks", in Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), Rome, Italy, July 2001.
- [3] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", in Proceedings of IEEE HICSS, Jan 2000.
- [4] S. Younis, S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach", in Proceedings of IEEE INFOCOM, March, Hong Kong, China, 2004
- [5] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh, "Max-Min D-clusterFormation in Wireless Ad Hoc Networks", in Proceedings of IEEE INFOCOM, March 2000.
- [6] D. J. Baker and A. Ephremides, "The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm", IEEE Transactions on Communications, Vol. 29, No. 11, pp. 1694-1701, Nov. 1981.
- [7] A. Ephremides, J. E. Wieselthier and D. J. Baker, "A Design concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling", Proceedings of IEEE, vol. 75, No. 1, pp. 56-73, 1987.
- [8] A. K. Parekh, "Selecting Routers in Ad-Hoc Wireless Networks", in Proceedings of ITS, 1994.
- [9] C. R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks", Journal on Selected Areas in Communication, Vol. 15, pp. 1265-1275, Sep. 1997.
- [10] A. D. Amis, and R. Prakash, "Load-Balancing Clusters in Wireless Ad Hoc Networks", in Proceedings of ASSET 2000, Richardson, Texas, Mar. 2000.
- [11] M. Chatterjee, S. K. Das, and D. Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks", Cluster Computing, pp. 193-204, 2002
- [12] S. Basagni, "Distributed Clustering for Ad Hoc Networks", in Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks, pp. 310-315, June. 1999.
- [13] C. F. Chiasserini, I. Chlamtac, P. Monti and A. Nucci, "Energy Efficient design of Wireless Ad Hoc Networks", in Proceedings of European Wireless, Feb. 2002.
- [14] 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, Aug. 1999.
- [15] S. G. Foss and S. A. Zuyev, "On a Voronoi Aggregative Process Related to a Bivariate Poisson Process", Advances in Applied Probability, vol. 28, no. 4, pp. 965-981, 1996.
- [16] F. Baccelli and S. Zuyev, "Poisson Voronoi Spanning Trees with Applications to the optimization of Communication Networks", Operations Research, vol. 34, no. 1, pp. 619-631, 1999.
- [17] S. Bandyopadhyay and E. Coyle, "An Energy-Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks", in Proceedings of IEEE INFOCOM, April. 2003.
- [18] F. Ye, G. Zhong, S. Lu, and L. Zhang, "PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks", in International Conference on Distributed Computing Systems (ICDCS), 2003.
- [19] W. C. Y. Lee, Mobile Cellular Telecommunications. McGraw Hill, 1995.
- [20] The CMU Monarch Project. The CMU Monarch Project's Wireless and Mobility Extensions to NS.