EURASIP Journal on Wireless Communications and Networking 2005:5, 686–697

Automatic Decentralized Clustering for Wireless Sensor Networks

Chih-Yu Wen
Department of Electrical and Computer Engineering, University of Wisconsin-Madison, 1415 Engineering Drive, WI 53706-1691, USA
Email: wen@cae.wisc.edu

William A. Sethares
Department of Electrical and Computer Engineering, University of Wisconsin-Madison, 1415 Engineering Drive, WI 53706-1691, USA
Email: sethares@ece.wisc.edu

Received 6 June 2004; Revised 28 March 2005

We propose a decentralized algorithm for organizing an ad hoc sensor network into clusters. Each sensor uses a random waiting timer and local criteria to determine whether to form a new cluster or to join a current cluster. The algorithm operates without a centralized controller, it operates asynchronously, and does not require that the location of the sensors be known a priori. Simplified models are used to estimate the number of clusters formed, and the energy requirements of the algorithm are investigated. The performance of the algorithm is described analytically and via simulation.

Keywords and phrases: wireless sensor networks, clustering algorithm, random waiting timer.

1. INTRODUCTION

Unlike wireless cellular systems with a robust infrastructure, sensors in an ad hoc network may be deployed without infrastructure, which requires them to be able to self-organize. Such sensor networks are self-configuring distributed systems and, for reliability, should also operate without centralized control. In addition, because of hardware restrictions such as limited power, direct transmission may not be established across the complete network. In order to share information between sensors which cannot communicate directly, communication may occur via intermediaries in a multihop fashion. Scalability and the need to conserve energy lead to the idea of organizing the sensors hierarchically, which can be accomplished by gathering collections of sensors into clusters. Clustering sensors are advantageous because they

(i) conserve limited energy resources and improve energy efficiency,
(ii) aggregate information from individual sensors and abstract the characteristics of network topology,
(iii) provide scalability and robustness for the network.

This paper proposes a decentralized algorithm for organizing an ad hoc sensor network into clusters. Each sensor operates independently, monitoring communication among others. Those sensors which have many neighbors that are not already part of a cluster are likely candidates for creating a new cluster by declaring themselves to be a new "clusterhead." The clustering algorithm via waiting timer (CAWT) provides a protocol whereby this can be achieved and the process continues until all sensors are part of a cluster. Because of the difficulty of the analysis, simplified models are used to study and abstract its performance. A simple formula for estimating the number of clusters that will be formed in an ad hoc network is derived based on the analysis, and the results are compared to the behavior of the algorithm in a number of settings.

2. LITERATURE REVIEW

Several clustering algorithms have been proposed in recent years [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 22]. Many of the algorithms are heuristics intended to minimize the number of clusters. Some of the algorithms organize the sensors into clusters while minimizing the energy consumption needed to aggregate information and communicate the information to the base station. Perhaps the earliest of the clustering methods is the identifier-based heuristic called the

linked cluster algorithm (LCA) [5], which elects sensor to be a clusterhead if the sensor has the highest identification number among all sensors within one hop of its neighbors. The connectivity-based heuristic of [6, 8] selects the sensors with the maximum number of 1-hop neighbors (i.e., highest degree) to be clusterheads.

The weighted clustering algorithm (WCA) [9] considers the number of neighbors, transmission power, mobility, and battery usage in choosing clusters. It limits the number of sensors in a cluster so that clusterheads can handle the load without degradation in performance. These clustering methods rely on synchronous clocking for the exchange of information among sensors which typically limits these algorithms to smaller networks [10].

The Max-Min D-cluster algorithm [1] generates D-hop clusters with a complexity of $O(\mathrm{D})$ without time synchronization. It provides load balancing among clusterheads in the network. Simulation results suggest that this heuristic is superior to the LCA and connectivity-based solutions.

The low-energy adaptive clustering hierarchy (LEACH) of [11] utilizes randomized rotation of clusterheads to balance the energy load among the sensors and uses localized coordination to enable scalability and robustness for cluster set-up and operation. LEACH-C (centralized) [12] uses a centralized controller. The main drawbacks of this algorithm are nonautomatic clusterhead selection and the requirement that the position of all sensors must be known. LEACH's stochastic algorithm is extended in [13] with a deterministic clusterhead selection. Simulation results demonstrate that an increase of network lifetime can be achieved compared with the original LEACH protocol. In [14], the clustering is driven by minimizing the energy spent in wireless sensor networks. The authors adopt the energy model in [11] and use the subtractive clustering algorithm and fuzzy C-mean (FCM) algorithm to form clusters. Although the above algorithms carefully consider the energy required for clustering, they are not extensively analyzed (due to their complexity) and there is no way of estimating how many clusters will form in a given network.

The ad hoc network design algorithm (ANDA) [15] maximizes the network lifetime by determining the optimal cluster size and the optimal assignment of sensors to clusterheads but requires a priori knowledge of the number of clusterheads, number of sensors in the network, and the location of all sensors.

The distributed algorithm in [3] groups sensors into a hierarchy of clusters while minimizing the energy consumption in communicating information to the base station. They use the results provided in [18] to obtain optimal parameters of the algorithm and analyze the number of clusterheads at each level of clustering.

Most of these design approaches are deterministic protocols in which each sensor must maintain knowledge of the complete network [12, 15] or identify a subset of sensors with a clusterhead to partition the network into clusters in heuristic ways [1, 2, 4, 5, 6, 7, 8, 9, 22]. The algorithms proposed in [11, 12, 13, 14] focus on reducing the energy consumption without exploring the number of clusters generated by the protocols, though [1, 9] demonstrate the average number of clusterheads via simulations. For most of the algorithms, no analysis of the number of clusters is available.

The method of this paper is a randomized distributed algorithm in which each sensor uses a random waiting timer and local criteria to decide whether to be a clusterhead. The algorithm operates without a centralized controller, it operates asynchronously and does not require that the location of the sensors be known. Based on simplified models, an estimate of the number of clusterheads and a simple prediction formula are derived to approximate and describe the behavior of the proposed algorithm. To examine the energy usage of the algorithm, the result provided in [19] is used to investigate situations where the minimum transmission range ensures that the network have a strong connectivity. The performance of the algorithm is investigated both by simulation and analysis.

3. THE CLUSTERING ALGORITHM VIA WAITING TIMER

This section describes a randomized distributed algorithm that forms clusters automatically in an ad hoc network. The main assumptions are

(i) all sensors are homogeneous with the same transmission range,
(ii) the sensors are in fixed but unknown locations; the network topology does not change,
(iii) symmetric communication channel: all links between sensors are bidirectional,
(iv) there are no base stations to coordinate or supervise activities among sensors. Hence, the sensors must make all decisions without reference to a centralized controller.

Each active sensor broadcasts its presence via a "*Hello*" signal and listens for its neighbor's "*Hello*." The sensors that hear many neighbors are good candidates for initiating new clusters; those with few neighbors should choose to wait. By adjusting randomized waiting timers, the sensors can coordinate themselves into sensible clusters, which can then be used as a basis for further communication and data processing.

After deployment, each sensor sets a random waiting timer. If the timer expires, then the sensor declares itself to be a clusterhead, a focal point of a new cluster. However, events may intervene that cause a sensor to shorten or cancel its timer. For example, whenever the sensor detects a new neighbor, it shortens the timer. On the other hand, if a neighbor declares itself to be a clusterhead, the sensor cancels its own timer and joins the neighbor's new cluster.

Assume the initial value of the waiting time of sensor i, $WT_i^{(0)}$, is a sample from the distribution $C+\alpha\cdot U(0,1)$, where C and α are positive numbers, and $U(0,1)$ is a uniform distribution. In the clustering phase of the network, each sensor broadcasts a *Hello* message at a random time. This allows each sensor to estimate how many neighbors it has. A *Hello* message consists of (1) the sensor ID of the sending sensor, and (2) the cluster ID of the sending sensor. At the beginning, the cluster ID of each sensor is zero. Note that a sensor

(1) Each sensor initializes a random waiting timer with a value $WT_i^{(0)}$.
(2) Each sensor transmits the *Hello* message at random times:
draw a sample r from the distribution $\lambda \cdot WT_i^{(0)} \cdot U(0,1)$, where $0 < \lambda < 0.5$,
wait r time units and then transmit the *Hello*.
(3) Establish and update the neighbor identification:
if a sensor receives a message of assigning a cluster ID at time step k
(a) join the corresponding cluster,
(b) draw a sample r' from the distribution $WT_i^{(k)} \cdot U(0,1)$,
(c) wait r' time units and then send an updated *Hello* message with the new cluster ID,
(d) stop the waiting timer. (Stop!)
else
collect neighboring information.
end
(4) Decrease the random waiting time according to (1).
(5) Clusterhead check:
if $WT_i = 0$ and the neighboring sensors are not in another cluster
(a) broadcast itself to be a clusterhead,
(b) assign the neighboring sensors to cluster ID i. (Stop!)
elseif $WT_i = 0$ and some of the neighboring sensors are in other clusters
join any nearby cluster after τ seconds, where τ is greater than any possible waiting time. (Stop!)
else
go to step (3).
end

Algorithm 1: The CAWT: an algorithm for segmenting sensors into clusters.

ID is not needed to be unambiguously assigned to each sensor before applying the CAWT. The following are two possible ways for each sensor to determine its sensor ID: (1) each sensor can automatically know an ID number (like an IP address or an RFID tag), and (2) each sensor could pick a random number when it first turns on, which is a "random" ID assignment. If the range of numbers is large compared to the number of sensors, then it is unlikely that two sensors (within radio range) would pick the same number.

Sensors update their neighbor information (i.e., a counter specifying how many neighbors it has detected) and decrease the random waiting time based on each "new" *Hello* message received. This encourages those sensors with many neighbors to become clusterheads. The updating formula for the random waiting time of sensor i is

$$WT_i^{(k+1)} = \beta \cdot WT_i^{(k)}, \tag{1}$$

where $WT_i^{(k)}$ is the waiting time of sensor i at time step k and $0 < \beta < 1$.

If both of the following conditions apply, then sensor i declares itself a clusterhead:

(i) the random waiting timer expires, that is, $WT_i = 0$;
(ii) none of the neighboring sensors are already members of a cluster.

If sensor i satisfies the above conditions, it broadcasts a message proclaiming that it is beginning a new cluster; this also serves to notify its neighbors that they are assigned to join the new cluster with ID i. When a sensor joins the cluster, it sends an updated *Hello* message and stops its waiting timer. The complete procedure of the initialization phase is outlined in the CAWT of Algorithm 1.

After applying the CAWT, there are three different kinds of sensors: (1) the clusterheads, (2) sensors with an assigned cluster ID, and (3) sensors which are unassigned. These unassigned sensors may join the nearest cluster later depending on the neighboring information or the demand of specific applications, such as sensor location estimation problem. Thus, the topology of the ad hoc network is now represented by a hierarchical collection of clusters.

4. SIMPLIFIED METHODS OF CLUSTERING

Because of the complexity of the CAWT, it is difficult to evaluate the algorithm directly other than via simulation. Since the connectivity among sensors and the number of neighboring sensors play important roles in the CAWT, it is reasonable to investigate the performance from the perspective of these parameters. Therefore, we abstract the behavior of the algorithm using two simplified models which approximate the desired global behavior and serve to analyze its performance.

4.1. The neighboring density model

The first simplified model is the neighboring density model (NDM) which is detailed in Algorithm 2. The basic idea of NDM is to suppose that the probability of each sensor of being a clusterhead, p_i, is proportional to the number of the

(a) Assign a probability to sensor i, p_i, proportional to the number of the neighboring sensors, N_i. That is, $p_i \propto N_i / \sum_{i=1}^{n} N_i$.
(b) Let B_i be the set of neighboring sensors of sensor i.
I is the index set of clusterheads.
(c) $\mathbf{P}^{(k)}$, $\overline{\mathbf{P}}^{(k)}$, and $\widetilde{\mathbf{P}}^{(k)}$ are 1 by n vectors to store the probability distribution at time step k.
(d) Assign $k = 0$ and $\mathbf{P}^{(0)} = (p_1, p_2, \ldots, p_n)$.
while $\text{sum}(\mathbf{P}^{(k)}) > 0$
(1) Select a clusterhead
if $j = \arg\max_i \{p_i^{(k)}\}$
$j \in I$,
end
(2) Update the probability distribution
$\widetilde{p}_i^{(k)} = p_i^{(k)} \cdot 1_{\{i \notin B_j,\ B_i \cap B_j = \varnothing,\ j = \arg\max_i \{p_i^{(k)}\}\}}$,
$\widetilde{p_j}^{(k)} = 0$.
(3) Normalize the updated probability distribution.
if $\text{sum}(\widetilde{\mathbf{P}}^{(k)}) > 0$
$\overline{p_i}^{(k)} = \widetilde{p}_i^{(k)} / \text{sum}(\widetilde{\mathbf{P}}^{(k)})$.
else
$\overline{\mathbf{P}}^{(k)} = \widetilde{\mathbf{P}}^{(k)}$.
end
(4) Store the normalized probability distribution.
$\mathbf{P}^{(k)} = \overline{\mathbf{P}}^{(k)}$,
set $k = k + 1$.
end

Algorithm 2: The neighboring density model: a procedure for analyzing the CAWT.

neighboring sensors, N_i. That is,

$$p_i \propto \frac{N_i}{\sum_{i=1}^{n} N_i}. \tag{2}$$

If the sensor is not already chosen as a clusterhead and its neighboring sensors are not already in other clusters, then the sensor with the largest p_i is chosen to be a clusterhead and it assigns probability 0 to its neighbors. Thus, a sensor becomes a clusterhead if it has the highest neighboring density among all sensors which have not yet become cluster members. Moreover, if a sensor is not a member of a cluster and some of its neighbors have already become cluster members, this sensor should choose to wait and join the nearest cluster later. After normalizing the updated probability distribution of sensors, the procedure repeats until all sensors are members of a cluster. The rationale for this choice is that, if the random waiting time of each sensor is long enough (in the sense that each sensor is able to collect sufficient neighboring information), then the model is likely to closely approximate the behavior of the CAWT on any given ad hoc network. The close connection between the model and the algorithm is explored via simulation.

4.2. The averaged model

This subsection models the CAWT by a simplified averaging procedure. Assume that a single clusterhead and an average number of neighboring sensors $E^{(k)}[N_i]$ are removed during each iteration k. Assume that each sensor will be removed with probability $p_{rm}^{(k)} = r_k / m_k$, where r_k is the number of sensors to be removed and m_k is the number of sensors remaining at iteration k. Denote the collection of sensors at iteration k by V_k. Since a clusterhead and its neighboring sensors are removed at each iteration, the collection of sensors at the next iteration, V_{k+1}, is simply a new and smaller network. Theorem 1 can be applied to approximate the distribution of the number of clusterheads at iteration k by $\mathcal{N}(\mu_k, \sigma_k^2)$, where $\mu_k = \sum_{i=1}^{m_k} p_i^{(k)}$, $\sigma_k^2 = \sum_{i=1}^{m_k} p_i^{(k)}(1 - p_i^{(k)})$, m_k is the number of sensors in V_k, $p_i^{(k)}$ is the updated probability distribution of sensors at iteration k, $i \in I_k$, and I_k is the index set of sensors at iteration k. Once the procedure terminates, the number of iterations is an estimate of the number of clusterheads formed in the network. A statement of the averaged model I is given in Algorithm 3.

4.3. Analysis of the averaged model

This section analyzes the averaged model of Algorithm 3 and derives a simple expression for the expected number of clusterheads in a given network. Later sections show via simulation that this is also a reasonable estimate of the number of clusterheads given by the implementable CAWT of Algorithm 1.

4.3.1. The Lindeberg theorem

This section reviews the probability that is used when analyzing the performance of the model. Readers may see [20] for a complete discussion and proof of the theorem.

(a) Let $N_b^{(k)}$ be the sum of neighboring sensors at iteration k.
$N_b^{(k)} = \sum_{i=1}^{m_k} N_i^{(k)}$.
$i \in I_k$; I_k is the index set of sensors at iteration k.
(b) Let $E^{(k)}[N_i]$ be the average number of neighbors at iteration k.
(c) Assign the probability $p_i^{(k)}$ to sensor i, proportional to the number of neighboring sensors, $N_i^{(k)}$. That is, $p_i^{(k)} \propto N_i^{(k)}/N_b^{(k)}$.
(d) Assign $k = 0$, $m_0 = n$, $r_0 = 0$.
while $(m_k - r_k) > 0$
$r_k = \lceil E^{(k)}[N_i] \rceil^* + 1$,
$m_{k+1} = m_k - r_k$,
$k = k + 1$.
end
$^*\lceil \cdot \rceil$ is the ceiling function.

Algorithm 3: Averaged model I: procedure for analyzing the CAWT.

Suppose for each n that

$$\begin{aligned} &(X_{11}, X_{12}, \ldots, X_{1r_1}),\\ &(X_{21}, X_{22}, \ldots, X_{2r_2}),\\ &\qquad\vdots\\ &(X_{n1}, X_{n2}, \ldots, X_{nr_n}) \end{aligned} \tag{3}$$

are independent random vectors. The probability space may change with n. Put $S_n = X_{n1} + \cdots + X_{nr_n}$. In the network application, $r_n = n$, $X_{ni} = X_i, 0$, and (3) is called a *triangular array* of random variables. Let X_i take the values 1 and 0 with probability p_i and $q_i = 1 - p_i$. We may interpret X_i as an indicator that sensor i is chosen to be a clusterhead with probability p_i and S_n is the number of clusters in the network.

Denote $Y_i = X_i - p_i$. Hence,

$$\begin{aligned} S_n^Y \equiv \sum_{i=1}^{n} Y_i &= \sum_{i=1}^{n} X_i - \sum_{i=1}^{n} p_i = S_n - \sum_{i=1}^{n} p_i,\\ E[Y_i] &= E[X_i] - p_i = 0,\\ \sigma_{Y_i}^2 &= \sigma_{X_i}^2 = p_i(1 - p_i),\\ s_n^2 = \sum_{i=1}^{n} \sigma_{Y_i}^2 &= \sum_{i=1}^{n} \sigma_{X_i}^2 = \sum_{i=1}^{n} p_i(1 - p_i). \end{aligned} \tag{4}$$

For our case, the Lindeberg condition [20] reduces to

$$\lim_{n\to\infty} \sum_{i=1}^{n} \frac{1}{s_n^2} \int_{|Y_i| \geq \epsilon s_n} Y_i^2 dP \leq \lim_{n\to\infty} \sum_{i=1}^{n} \frac{1}{s_n^2} \int_{|Y_i| \geq \epsilon s_n} dP = 0, \tag{5}$$

which holds because all the random variables are bounded by 1 and $[|Y_i| \geq \epsilon s_n] \to 0$ as $n \to \infty$.

Theorem 1. *Suppose that Y_i is an independent sequence of random variables and satisfies $E[Y_i] = 0$, $\sigma_{Y_i}^2 = E[Y_i^2]$, $S_n^Y = \sum_{i=1}^{n} Y_i$, and $s_n^2 = \sum_{i=1}^{n} \sigma_{Y_i}^2$. If the Lindeberg condition* (5) *holds, then $S_n^Y/s_n \to \mathcal{N}(0, 1)$.*

By Theorem 1, the distribution of the number of clusters can be approximated by $\mathcal{N}(\sum_{i=1}^{n} p_i, s_n^2)$ since $E[S_n] = E[S_n^Y] + \sum_{i=1}^{n} p_i = \sum_{i=1}^{n} p_i$ and $\sum_{i=1}^{n} \sigma_{X_i}^2 = \sum_{i=1}^{n} \sigma_{Y_i}^2 = s_n^2$.

4.3.2. Special case

Assume that n sensors are deployed in a circle and the distance between each pair of neighboring sensors is equal. In addition, because of the radio range, assume that each sensor can detect two neighboring sensors. Hence each sensor may be chosen as a clusterhead with probability $p_i = 1/n$. As mentioned before, let X_i be the indicator that sensor i is chosen to be a clusterhead with probability p_i and let S_n be the number of clusterheads in the network. Based on these assumptions, the expectation and variance of S_n are

$$\begin{aligned} E[S_n] &= \sum_{k=1}^{n} kP_r(S_n = k) = np_i,\\ s_n^2 &= \sum_{i=1}^{n} \sigma_{X_i}^2 = np_i(1 - p_i). \end{aligned} \tag{6}$$

4.3.3. Analysis

This section shows that, with appropriate simplification, the averaged model (AM) can be used to make simple prediction of the behavior of the CAWT.

To obtain the mean and variance of the number of clusterheads of each iteration, the probability distribution of these random variables must be updated. However, it is not simple to calculate $p_i^{(k)}$ at each iteration since the process of selecting a clusterhead at each iteration is complex. The following simplified analysis restructures the connectivity of the network so that each sensor has the same average neighboring density at each iteration. Therefore, we have

$$E^{(k+1)}[N_i] = \frac{N_b^{(k)} - r_k \cdot E^{(k)}[N_i]}{m_{k+1}}. \tag{7}$$

This simplified averaged model is summarized in averaged model II in Algorithm 4.

(a) Let $N_b^{(k)}$ be the sum of neighboring sensors of sensors at iteration k.
$N_b^{(k)} = \sum_{i=1}^{m_k} N_i^{(k)}$.
$i \in I_k$; I_k is the index set of sensors at iteration k.

(b) Let $E^{(k)}[N_i]$ be the average number of neighbors at iteration k.
$E^{(0)}[N_i] = N_b^{(0)}/m_0$.

(c) Assign the probability $p_i^{(k)}$ to sensor i, proportional to the number of neighboring sensors, $N_i^{(k)}$. That is, $p_i^{(k)} \propto N_i^{(k)}/N_b^{(k)}$.

(d) Assign $k = 0$, $m_0 = n$, $r_0 = 0$.
while $(m_k - r_k) > 0$
$m_{k+1} = m_k - r_k$,
$E^{(k+1)}[N_i] = (N_b^{(k)} - r_k \cdot E^{(k)}[N_i])/m_{k+1}$,
$r_{k+1} = \lceil E^{(k+1)}[N_i] \rceil^* + 1$,
$k = k + 1$.
end
$^*\lceil \cdot \rceil$ is the ceiling function.

Algorithm 4: Averaged model II: procedure for analyzing the CAWT.

Thus, the distribution of the number of clusterheads can be approximated by $N(\mu_{\text{ch}}, \sigma_{\text{ch}}^2)$, where

$$\mu_{\text{ch}} = \sum_{k=1}^{N_{it}} \mu_k = \sum_{k=1}^{N_{it}} \sum_{i=1}^{m_k} p_i^{(k)},$$
$$\sigma_{\text{ch}}^2 = \sum_{k=1}^{N_{it}} \sigma_k^2 = \sum_{k=1}^{N_{it}} \sum_{i=1}^{m_k} p_i^{(k)}(1 - p_i^{(k)}), \tag{8}$$

where N_{it} is the number of iterations.

Moreover, suppose that the expectation of the number of neighboring sensors of each sensor in the network is used to approximate the number of neighboring sensors that will be removed at each iteration (i.e., the sensors which will eventually join the new cluster). Thus,

$$E^{(k)}[N_i] = E[N_i] = \frac{1}{n} \sum_{i=1}^{n} N_i, \quad \forall k. \tag{9}$$

Then

$$r_k = \lceil E[N_i] \rceil + 1, \tag{10}$$

and a simple formula for predicting the number of clusterheads is

$$N_{\text{ch}} = \frac{n}{\lceil E[N_i] \rceil + 1}. \tag{11}$$

The comparison of the performance of the CAWT and the simplified models will be illustrated in Section 6.

5. ANALYSIS OF ENERGY CONSUMPTION

This section considers the energy consumption of the CAWT assuming homogenous sensors. The total power requirements include both the power required to transmit messages and the power required to receive (or process) messages.

In the initialization phase, each sensor broadcasts a *Hello* message to its neighboring sensors. Therefore, the number of transmissions N_{T_x} is equal to the number of sensors in the network, n, and the number of receptions N_{R_x} is the sum of the neighboring sensors of each sensor. That is,

$$N_{T_x} = n, \qquad N_{R_x} = \sum_{j=1}^{n} N_j. \tag{12}$$

As a sensor, say sensor i, meets the conditions of being a clusterhead, it broadcasts this and assigns cluster ID i to its neighboring sensors. Its neighboring sensors then transmit a signal to their neighbors to update cluster ID information. During this clustering phase, $(1+N_i)$ transmissions and $(N_i + \sum_{j \in C_i} N_j)$ receptions are executed, where C_i is the index set of neighboring sensors of sensor i. This procedure is applied to all clusterheads and their cluster members. Now let $N_{T_x}^c$ and $N_{R_x}^c$ denote the number of transmissions and receptions for all clusters, respectively. Hence,

$$N_{T_x}^c = \sum_{i \in I} (1 + N_i),$$
$$N_{R_x}^c = \sum_{i \in I} \left(\sum_{j \in C_i} N_j + N_i \right), \tag{13}$$

where I is a index set of clusterheads. Therefore, the total number of transmissions N_T and the number of receptions

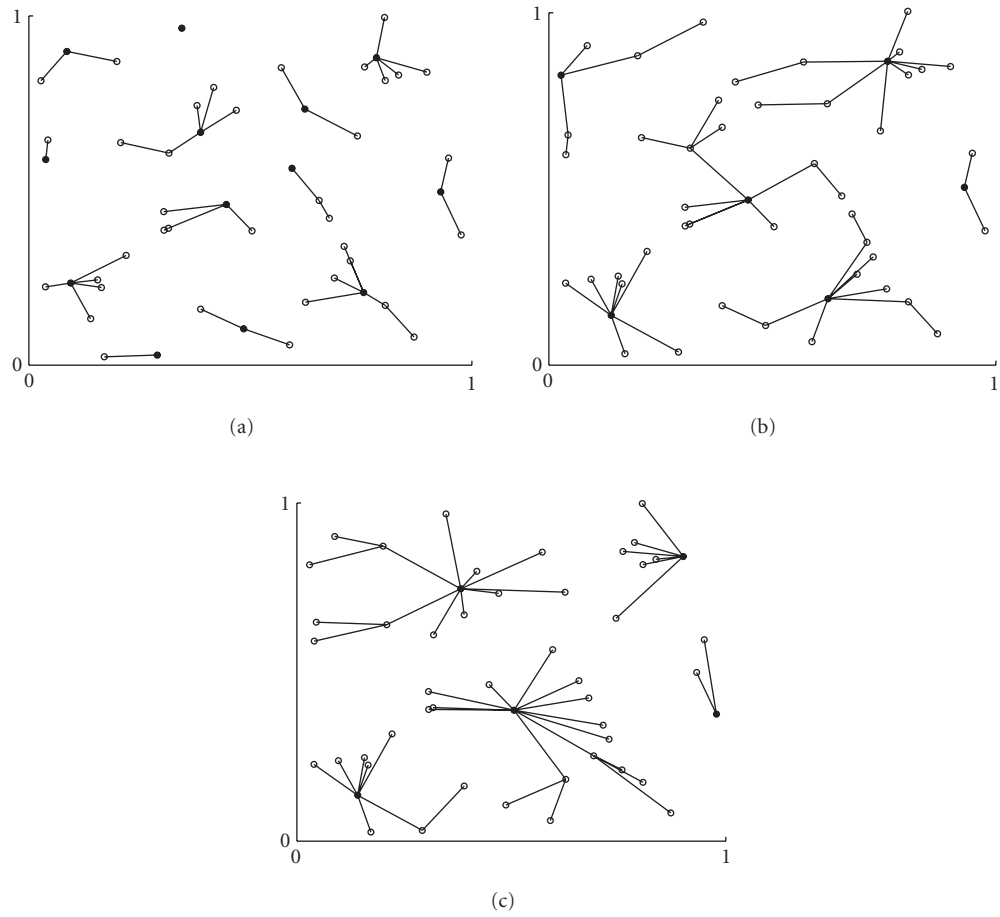

Figure 1: Clusters are formed in a random network of 50 sensors with (a) $R/l = 0.15$, (b) $R/l = 0.2$, and (c) $R/l = 0.25$.

N_R are

$$N_T = N_{T_x} + N_{T_x}^c = n + \sum_{i \in I} (1 + N_i),$$
$$N_R = N_{R_x} + N_{R_x}^c = \sum_{j=1}^{n} N_j + \sum_{i \in I} \left(\sum_{j \in C_i} N_j + N_i \right). \tag{14}$$

Suppose that the energy needed to transmit is E_T, which depends on the transmitting range R, and the energy needed to receive is E_R. From (14), the total energy consumption, E_{total}, for cluster formation in the wireless sensor network is

$$E_{\text{total}} = N_T \cdot E_T + N_R \cdot E_R. \tag{15}$$

Observe that the above analysis is suitable for any transmitting range. However, overly small transmission ranges may result in isolated clusters whereas overly large transmission ranges may result in a single cluster. Therefore, in order to optimize energy consumption and encourage linking between clusters, it is sensible to consider the minimum transmission power (or range R) which will result in a fully connected network. This *range assignment* problem is investigated in [19], which proposes lower boundson the magnitude of $R^d n$ (with respect to l), $R^d n \in O(l^d)$, and shows that $R^d n \approx l^d \ln(l)$ may be a good initial value for the search of optimized range assignment strategies to provide a high probability of connectivity. As usual, n is the number of sensors and l is the length of sides of a d-dimensional cube. The performance of the total energy consumption of the CAWT with different selections of R is examined via simulation.

6. SIMULATION RESULTS

The simulations of this section examine the performance of the CAWT and validate the simplified models for which analytical results have been derived.

Assume that n sensors are uniformly distributed over a square region in a two-dimensional space. Parameters for the random waiting timer, number of sensors, and ratio of transmitting range R to the side length l of the square, R/l, are investigated to provide a simulation-based study of the CAWT. Note that the entire experiments are conducted in a square region with side length $l = 1000$ unit length.

The first set of experiments examines the variation of the average number of clusterheads with respect to the ratio R/l. With random waiting time parameters $C = 100$, $\alpha = 10$, and

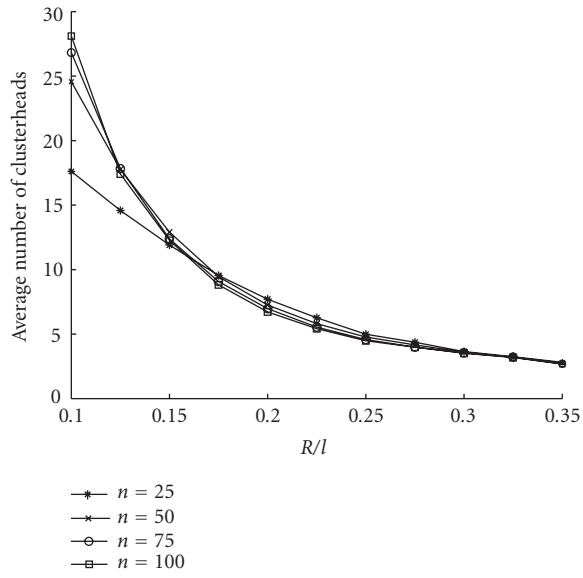

Figure 2: Average number of clusterheads as a function of the ratio R/l.

$\beta = 0.9$, Figure 1 depicts typical runs of the algorithm based on the same network topology but with different R/l ratios. The results show that each cluster is a collection of sensors which are up to 2 hops away from a clusterhead. Figure 2 shows the relationship between the average number of clusterheads and the R/l ratio with varying the number of sensors. The average number of clusterheads in each case is the sample mean of the results of 200 typical runs. Observe that the average number of clusterheads decreases as the ratio R/l increases (i.e., the transmission power increases). Since larger transmission power allows larger radio coverage, a clusterhead has more cluster members, which reduces the number of clusters in the network. Figure 2 also shows that when the transmission range is small, the network with a lower sensor density will have a larger percentage of isolated sensors which eventually become clusterheads in their own right. This is because the network is only weakly connected with these values. On the other hand, when the transmission power is large enough to ensure strong connectivity of the network, the average number of clusterheads stabilizes as the number of sensors increases.

The second set of experiments in Figure 3 evaluates the performance of the neighboring density model (NDM), which compares cluster formation when using the NDM and the CAWT. The outputs of the two methods are not identical due to the randomness of the waiting timer. Nonetheless, both these clustering structures are qualitatively similar given the same network settings, suggesting that the NDM provides a good approximation to the CAWT.

The third set of experiments compares the estimates of the number of clusterheads when applying the CAWT, the neighboring density model (NMD), the averaged model (AM), and the prediction formula. In each method, the results of 200 typical runs are merged. For the CAWT, the NDM, and the prediction formula cases, the estimates of the number of clusterheads are given by the sample mean and sample variance of the results of typical runs. For the AM case, the estimates of mean and variance of the number of clusterheads are generated in each typical run, which means the best estimate may not be obtained by averaging the typical runs. The covariance intersection (CI) method of [21] provides the best estimate given the information available. The CI algorithm takes a convex combination of mean and covariance estimates that are represented in information space. Since these typical runs are independent, the cross-correlations between these estimates are 0. Therefore, the general form is

$$\begin{aligned} P_{cc}^{-1} &= \omega_1 P_{a_1 a_1}^{-1} + \cdots + \omega_n P_{a_n a_n}^{-1}, \\ P_{cc}^{-1} c &= \omega_1 P_{a_1 a_1}^{-1} a_1 + \cdots + \omega_n P_{a_n a_n}^{-1} a_n, \end{aligned} \tag{16}$$

where $\sum_{i=1}^{n} \omega_i = 1$, $n > 2$, a_i is the estimate of the mean from available information, $P_{a_i a_i}$ is the estimate of the variance from available information, c is the new estimate of the mean, and P_{cc} is the new estimate of the variance. We choose to weight each typical run equally.

In order to compare the CAWT and the simplified models, Figures 4a and 4b show the standard deviation of the mean number of clusterheads. The plots vary the number of sensors n and the transmission power R/l. Also shown in Figures 4c and 4d are the confidence intervals for the mean number of clusterheads at a 90% confidence level. The graphs suggest that the NDM approximates the CAWT somewhat better than the AM. This is reasonable because the NDM retains global connectivity information while the AM uses only the average density information. Though the NDM outperforms AM, these results provide evidence that the AM provides a way to roughly predict the performance of the CAWT.

The fourth set of experiments considers the total energy consumption of the CAWT. Assume that the communication channel is error-free. Since each sensor does not need to retransmit any data, two transmissions are executed, one for broadcasting the existence and the other for assigning a cluster ID to its cluster members or updating the cluster ID information of its neighbors. Hence, the total number of transmissions is $2n$. Under these circumstances, sensor i will receive $2N_i$ messages. Then, the total number of receptions is $2\sum_{i=1}^{n} N_i$. Figures 5 and 6 show the average number of transmissions and receptions of random networks after applying the proposed algorithm. Figure 6 also shows that the number of receptions tends to increase as the ration R/l increases. This implies that energy consumption is higher for the network with larger transmission power. This can be attributed to the fact that larger transmission power allows sensors to detect more neighbors, which increases the number of receptions when assigning cluster ID or updating cluster ID information. Therefore, in order to minimize energy use and keep strong connectivity in the network, an appropriate selection of the transmission range R is essential. In [19], the authors

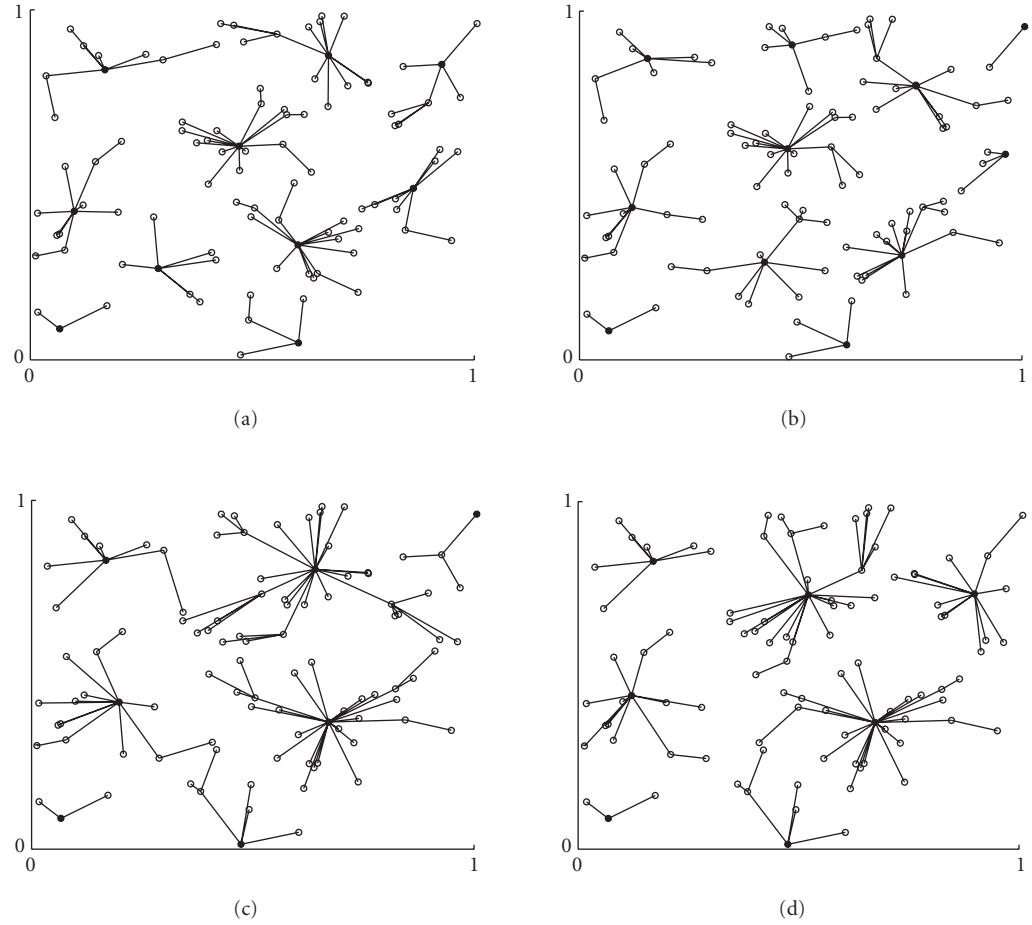

Figure 3: Cluster formation in a random network with 100 sensors and (a) the CAWT with $R/l = 0.15$, (b) the NDM algorithm with $R/l = 0.15$, (c) the CAWT with $R/l = 0.2$, and (d) the NDM algorithm with $R/l = 0.2$.

suggest that

$$R \approx l\sqrt[d]{\frac{\log l}{n}} \tag{17}$$

may be a good choice for the initial range assignment for sensors in the d-dimensional space. Hence, if $l = 1000\,\text{m}$ and $n = 100$, then $R \approx 173.21\,\text{m}$. This means that for $R/l \approx 0.173$, it may lead to a strongly connected network and energy conservation.

The final set of experiments compares the cluster formation when using the Max-Min D-cluster formation algorithm [1] and the new decentralized clustering algorithm with random waiting timer. The Max-Min heuristic generalizes the clustering heuristics so that a sensor is either a clusterhead or at most D hops away from a clusterhead. This heuristic has complexity of $O(\text{D})$ rounds which is better than most clustering algorithms in the literature (see [5, 6, 7, 8, 22]) with time complexity of $O(n)$, where n is the number of sensors in the network. In the proposed CAWT, each sensor initiates 2 rounds of local flooding to its 1-hop neighboring sensors, one for broadcasting sensor ID and the other for broadcasting cluster ID, to select clusterheads and form 2-hop clusters. Hence, the time complexity is $O(2)$ rounds. This implies that the CAWT and the Max-Min heuristic with $\text{D} = 2$ have the same time complexity $O(2)$. Thus the Max-Min heuristic with $\text{D} = 2$ provides a good way to benchmark the performance of the CAWT.

As shown in Figure 2 and by the figures in [1], load balancing may not be achieved without an appropriate transmission range since this may lead to either too large or too small cluster sizes. Hence, the cluster formation is examined with respect to the R/l ratio and network density suggested in (17) when using both the CAWT and the Max-Min heuristic. Figures 7 and 8 show that both the average number of the CAWT clusterheads and the Max-Min clusterheads increase approximately linearly with increased network density though the Max-Min heuristic has more clusterheads and slightly smaller cluster sizes than the CAWT. Figure 8 also demonstrates that a good selection of transmission range may lead to a minimal variation of the cluster size with increased network density. This

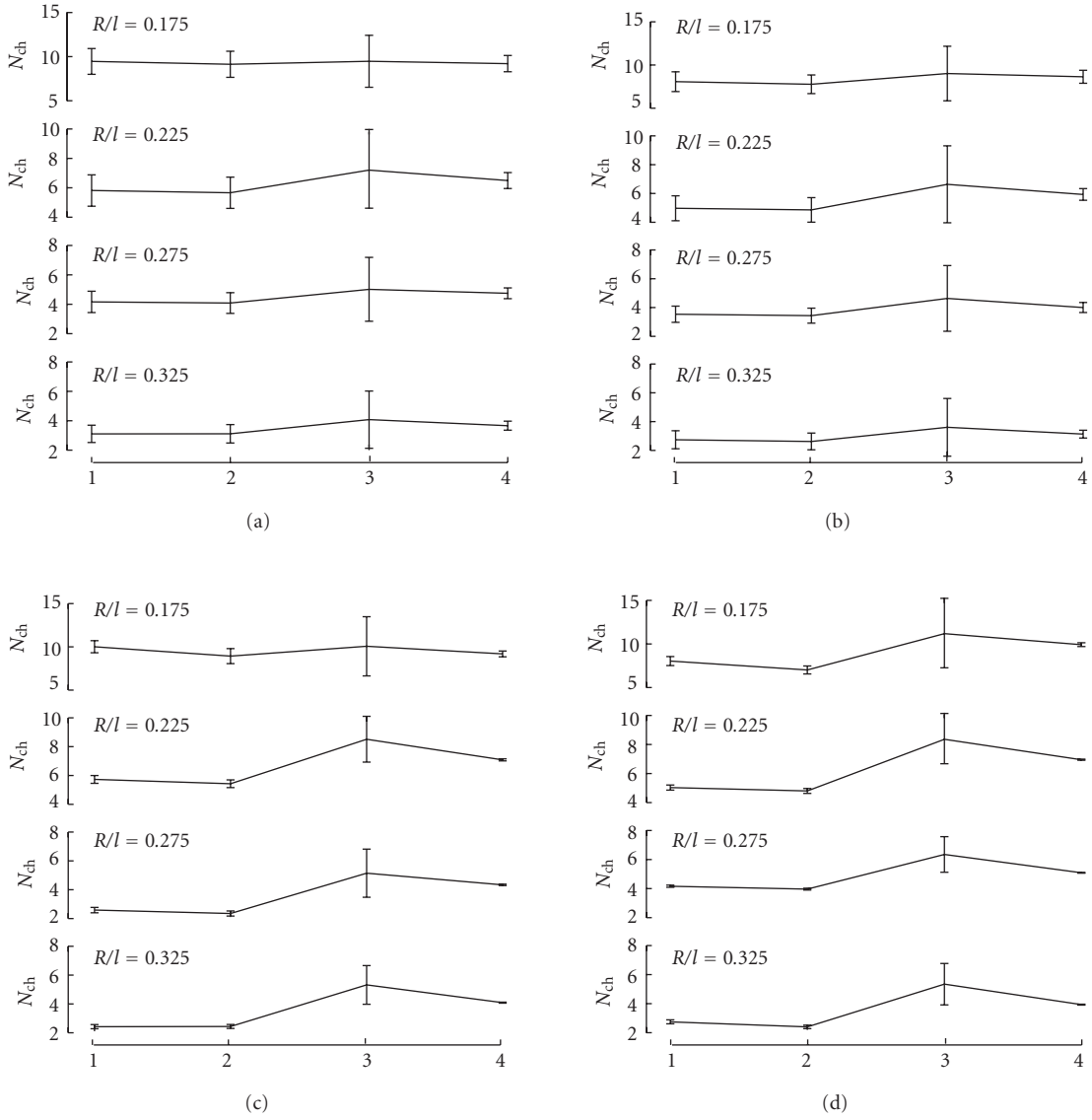

Figure 4: The number of clusterheads formed in a random network using (1) the CAWT, (2) NDM, (3) AM, and (4) the prediction formula, respectively, with varying R/l ratios. Parts (a) $n = 50$ and (b) $n = 100$ show the standard deviation over 200 runs. Parts (c) $n = 50$ and (d) $n = 100$ show the confidence intervals at the 90% level.

may help to achieve the load balance among the clusterheads.

The above set of experiments imply that the CAWT is competitive with the Max-Min heuristic in terms of time complexity and cluster formation. The authors in [1] show that the Max-Min heuristic may fail to provide a good cluster formation in some network configurations and more study is needed to determine appropriate times to trigger the Max-Min heuristic. In comparison, the CAWT may be reliably applied to any network topology and network density.

7. CONCLUSION

This paper has presented a randomized, decentralized algorithm for organizing the sensors of an ad hoc network into clusters. A random waiting timer and a neighbor-based criteria were used to form clusters automatically. Two simplified models are introduced for the purpose of understanding the performance of the CAWT. Simulation results indicated that the simplified models agree well with the behavior of the algorithm. Under the assumption of fixed transmission power and homogenous sensors, the energy requirements of the method were determined.

There are several ways this work may be generalized. For a fixed clusterhead selection scheme, a clusterhead with constrained energy may drain its battery quickly due to heavy utilization. In order to spread the energy usage over the network and achieve a better load balancing among clusterheads, reselection of the clusterheads may be a useful

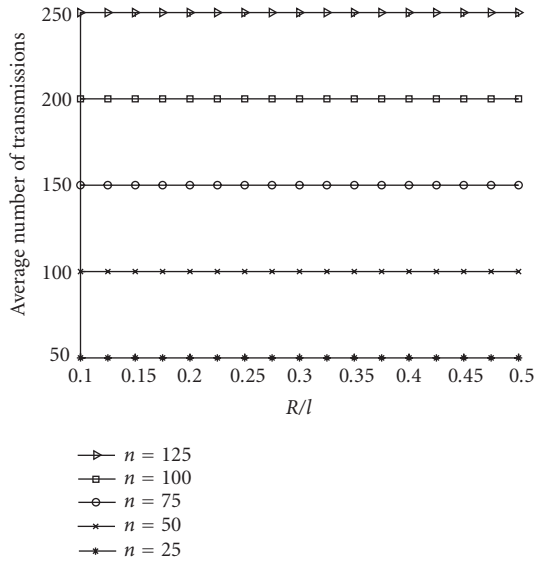

Figure 5: The number of transmissions in random networks as a function of the number of sensors and R/l ratio.

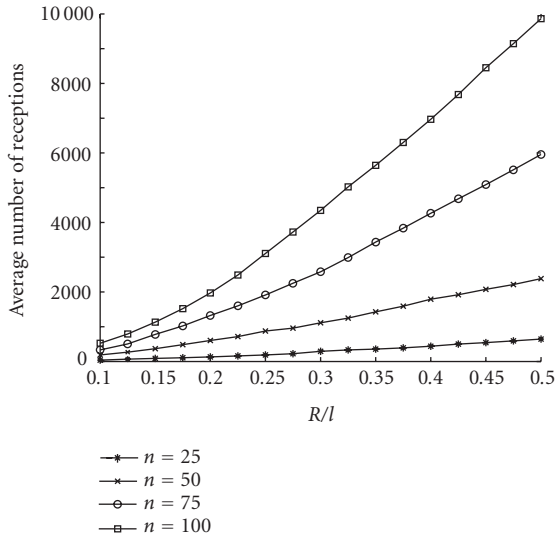

Figure 6: The number of receptions in random networks as a function of the number of sensors and R/l ratio.

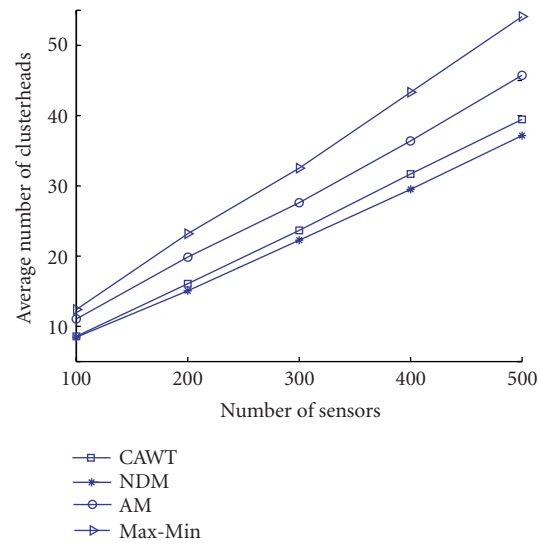

Figure 7: The average number of clusterheads as a function of the number of sensors and R/l ratio using the CAWT (and the two simplified models) and the Max-Min heuristic. $n = 100$, $R/l = 0.1732$; $n = 200$, $R/l = 0.1225$; $n = 300$, $R/l = 0.10$; $n = 400$, $R/l = 0.087$; $n = 500$, $R/l = 0.0775$.

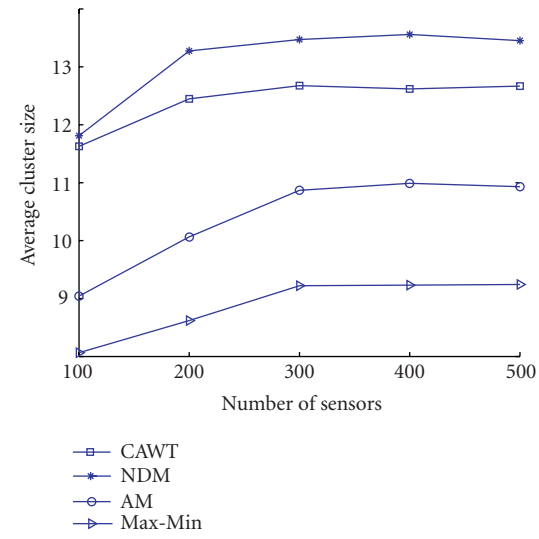

Figure 8: The average cluster size with the same network settings as in Figure 7.

strategy. Also, if the sensors are moving slowly, then the algorithm is flexible and cheap enough to be applied iteratively as the network configuration changes. This can be achieved by modifying the conditions under which the random timing counter is incremented or decremented. From an adaptive cross-layer design perspective, the random timer may be adjusted using current channel conditions (signal-to-interference-and-noise ratio (SINR), link connectivity, etc.) and energy constraints (energy level of neighboring sensors) from the physical layer. Moreover, the random timer may adapt based on the mobility of the sensor and the constraints from the MAC layer to achieve network robustness and scalability. Therefore, such "adaptive clustering protocols" may provide a reliable method of cluster organization for wireless ad hoc sensor networks.

REFERENCES

[1] 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. 19th IEEE Annual Joint Conference Computer and Communications Societies (INFOCOM '00)*, vol. 1, pp. 32–41, Tel Aviv, Israel, March 2000.

[2] A. D. Amis and R. Prakash, "Load-balancing clusters in wireless ad hoc networks," in *Proc. 3rd IEEE Symposium on*

Application-Specific Systems and Software Engineering Technology, pp. 25–32, Richardson, Tex, USA, March 2000.

[3] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *Proc. 22nd IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 3, pp. 1713–1723, San Francisco, Calif, USA, March–April 2003.

[4] S. Basagni, "Distributed clustering for ad hoc networks," in *Proc. 4th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN '99)*, pp. 310–315, Perth/Fremantle, WA, Australia, June 1999.

[5] D. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Trans. Commun.*, vol. 29, no. 11, pp. 1694–1701, 1981.

[6] M. Gerla and J. T. C. Tsai, "Multicluster, mobile, multimedia radio networks," *Wireless Networks*, vol. 1, no. 3, pp. 255–265, 1995.

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

[8] A. K. Parekh, "Selecting routers in ad-hoc wireless networks," in *Proc. SBT/IEEE International Telecommunications Symposium (ITS '94)*, pp. 420–424, Rio-de-Janeiro, Brazil, August 1994.

[9] M. Chatterjee, S. K. Das, and D. Turgut, "WCA: a weighted clustering algorithm for mobile ad hoc networks," *Journal of Cluster Computing*, vol. 5, no. 2, pp. 193–204, 2002, Special Issue on Mobile Ad hoc Networking.

[10] J. Lundelius and N. Lynch, "An upper and lower bound for clock synchronization," *Information and Control*, vol. 62, no. 2/3, pp. 190–204, 1984.

[11] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Annual Hawaii International Conference on System Sciences (HICSS '00)*, vol. 2, Maui, Hawaii, USA, January 2000.

[12] W. R. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transaction on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[13] M. J. Handy, M. Haase, and D. Timmermann, "Low energy adaptive clustering hierarchy with deterministic cluster-head selection," in *Proc. 4th IEEE International Workshop on Mobile and Wireless Communications Network (MWCN '02)*, pp. 368–372, Stockholm, Sweden, September 2002.

[14] M. N. Halgamuge, S. M. Guru, and A. Jennings, "Energy efficient cluster formation in wireless sensor networks," in *Proc. 10th IEEE International Conference on Telecommunications (ICT '03)*, vol. 2, pp. 1571–1576, Papeete, French Polynesia, February–March 2003.

[15] C. F. Chiasserini, I. Chlamtac, P. Monti, and A. Nucci, "Energy efficient design of wireless ad hoc networks," in *Proc. of IFIP Networking*, pp. 376–386, Pisa, Italy, May 2002.

[16] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *Journal on Selected Areas in Communication*, vol. 15, no. 7, pp. 1265–1275, 1997.

[17] A. B. McDonald and T. F. 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, 1999.

[18] 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.

[19] P. Santi, D. M. Blough, and F. Vainstein, "A probabilistic analysis for the range assignment problem in ad hoc networks," in *Proc. 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '01)*, pp. 212–220, Long Beach, Calif, USA, October 2001.

[20] P. Billingsley, *Probability and Measure*, John-Wiley & Sons, New York, NY, USA, 1979.

[21] S. Julier and J. K. Uhlmann, "General decentralized data fusion with covariance intersection (CI)," in *Handbook of Multisensor data Fusion*, D. L. Hall and J. Llinas, Eds., CRC Press, Boca Raton, Fla, USA, 2001.

[22] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Proc. IEEE International Conference on Communications (ICC '97)*, vol. 1, pp. 376–380, Montreal, Que., Canada, June 1997.

Chih-Yu Wen received the B.S.E.E. and M.S.E.E. degrees with high honors from the National Cheng Kung University in Taiwan in 1995 and 1997, respectively. He received an M.S.E.E. degree from the University of Wisconsin-Madison in 2002 and will complete the Ph.D. in electrical engineering from the University of Wisconsin-Madison in 2005. He has worked on cellular mobile systems emphasizing the capacity of wireless channels and networks, modulation, and error control coding. Current research interests include wireless communications, adaptive signal processing, software-defined radio, and adaptive distributed algorithms for wireless ad hoc sensor networks.

William A. Sethares received the B.A. degree in mathematics from Brandeis University, Waltham, Mass, and the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY. He has worked at the Raytheon Company as a Systems Engineer and is currently a Professor in the Department of Electrical and Computer Engineering at the University of Wisconsin - Madison. His research interests include adaptation and learning in signal processing, communications, and acoustics, and is the author of *Tuning, Timbre, Spectrum, Scale* (Springer, 1998) and the coauthor of *Telecommunication Breakdown* (Prentice-Hall, 2004).