

Partial Clustering: Maintaining Connectivity in a Low Duty-Cycled Dense Wireless Sensor Network

Chih-fan Hsin, Mingyan Liu

Electrical Engineering and Computer Science Department
University of Michigan, Ann Arbor
{chsin,mingyan}@eecs.umich.edu

Abstract— We consider a dense wireless sensor network where the radio transceivers of the sensor nodes are heavily duty-cycled in order to conserve energy. The chief purpose of the sensor network is surveillance and monitoring, where upon observation of certain event of interest, a sensor node generates a message and forwards it to a gateway located somewhere in or near the network. This forwarding relies on routes constructed using sensors whose radios are on/active. In order for such messages to reach the gateway with minimal delay, any sensor in the network should ideally have a route to the gateway consisting of active sensors at all times. Prior approaches to similar problems include clustering, virtual backbone, and connected dominating sets. Low energy consumption and good connectivity are potentially conflicting objectives. Our principal goal is to find an approach that results in the lowest possible duty cycle, and that provides better trade-off between the two objectives. In this paper we introduce the concept of *partial clustering*, which may be viewed as a generalized method of clustering. We compare the theoretical performance of different instances of partial clustering to that of standard clustering, and show that partial clustering can achieve lower duty cycle and provide greater flexibility in the trade-off between energy efficiency and connectivity. We then present a distributed algorithm based on the partial clustering method. Simulation results are provided to evaluate their effectiveness and energy efficiency.

Index Terms— System design, wireless sensor networks, connectivity, clustering, connected dominating set, energy efficiency

I. INTRODUCTION

Energy efficiency is a critical issue for the proper functioning of wireless sensor networks and has been the focus of many recent studies. It has been widely accepted that one of the most effective ways of conserving energy is to put the sensor nodes to sleep from time to time, i.e., to operate sensors at a lower *duty cycle*, defined as the percentage of on/active time. This could mean switching off the radio transceivers or the sensory devices, or both. The price we pay for letting the sensors alternate between on and off modes is performance degradation. Duty cycling radios implies intermittent communication capability while duty cycling sensory devices implies intermittent sensing capability. Alleviating methods include redundancy in sensor deployment and good scheduling

This work was supported in part by the Engineering Research Centers Program of the National Science Foundation under NSF Award EEC-9986866, NSF Award ANI-0238035, and through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011.

algorithms by which sensors determine when to switch off and for how long.

In this paper we study the problem of maintaining connectivity in a sensor network where sensors are duty-cycled¹. We will limit our attention to the duty cycling of radio transceivers only (for the rest of the paper the term duty cycle will be used in this limited sense). The goal is to develop methods that allow the network to operate at a low duty cycle with balanced energy consumption across the network, while maintaining connectivity.

Specifically, consider a wireless sensor network used for surveillance. Sensors are scattered over a field to monitor the area for events of interest. Upon occurrence of such an event, a sensor detecting it may generate a message to be delivered to a gateway (or controller) somewhere within or near the network. The path toward the gateway has to rely on sensors that are on/active. If a path consisting of active sensors does not exist, the message will have to wait till one becomes available. In order to have timely delivery whenever a need arises, it is desirable for the network to have connectivity at all times, in the sense that every node (active or inactive) should have a path consisting of active nodes to the gateway. Note that an inactive sensor detecting certain event may wish to switch on its radio to communicate.

This reduces to the problem of selecting a subset of the nodes, such that nodes within the set are reachable from the gateway and every node outside the set is connected to at least one node in the set. Once we have such a set, then only nodes in the set need to be on while all other nodes may be turned off. To balance energy consumption we then need multiple such sets, preferably mutually exclusive but collectively including all nodes, so that we can rotate among them.

This is essentially the problem of finding connected dominating sets (CDS) in a network [1], [2]. To reduce duty cycle, we need as many such sets as possible; to balance energy, we need these sets to have as few common elements as possible. This leads to the problem of finding the minimum connected dominating set (MCDS) (e.g., [3], [4]), which is known to be NP-complete in a circle graph [5] even as a centralized algorithm.

Many simpler, heuristic-based approaches have been pro-

¹An important underlying assumption of this study is that sensors are cheap enough to be deployed in large quantities. It is this redundancy in deployment that allows us to duty cycle the sensors while maintaining a desired level of performance.

posed that aim at finding reasonably good connected dominating sets rather than the minimum. There are two main types of such constructions. The first type is (virtual) cluster-based, where the sensing field is partitioned into multiple clusters (or cells) of sensors and the partition satisfies the following conditions.

Condition 1: All sensors in the same cluster can communicate with each other directly (via 1 hop).

Condition 2: All sensors in the same cluster can communicate with all sensors in the neighboring clusters directly.

Subsequently, only one sensor needs to be on in each cluster to maintain connectivity. The definition of a “neighboring” cluster varies, but has to be such that the resulting active sensors form a connected set. Sensors in the same cluster may take turns to be active in a round-robin fashion to share the responsibility. Thus a particular sensor’s duty cycle is inversely proportional to the number of sensors in the same cluster. As this number varies from cluster to cluster (for a random deployment), the network may need to be re-partitioned from time to time to balance energy consumption. Examples of this approach include GAF [6] and [7]. They partitioned the field into small square clusters, each with 4 and 8 neighboring squares, respectively. These methods will be analyzed further in Section III. [8] uses the same square partition and derives an asymptotic lower bound on the network lifetime.

The second type of construction is known as the virtual backbone, used to support multicasting [9] and fault-tolerant routing [3], [10], [11] for mobile nodes in an ad hoc network. As long as the backbone is connected, this concept can be used to duty cycle sensors while maintaining connectivity. Thus Connected dominating sets may be viewed as a special case of the virtual backbone. [2] proposed an approach called SPAN to construct a connected dominating set of sensors.

In this paper, we introduce the concept of *partial clustering*, where a field is first partitioned into clusters/cells, but only nodes within a sub-area of each cluster/cell are selected to form the connected dominating set (or virtual backbone). In many instances partial clustering is a generalized form of standard clustering, with tunable parameters. By limiting the selection to sub-areas within clusters, this approach imposes more structure on the formation of the backbone. In doing so it also achieves low duty cycle and a more flexible trade-off between energy efficiency and connectivity than standard clustering. We will also show that this approach is asymptotically order optimal.

For the rest of our discussion we will use the terms *on*, *active*, and *awake* interchangeably and the terms *off*, *inactive*, and *asleep* interchangeably.

The rest of the paper is organized as follows. Section II presents the network model and the key ideas that differentiate our approach from prior work. In Section III we compare a number of cluster-based approaches. Section IV presents in detail the partial clustering approach. We then develop distributed algorithms based on the partial clustering idea in Section V. Section VI evaluates the performance of the distributed algorithms, and Section VII concludes the paper.

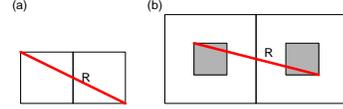


Fig. 1. (a) Cluster-based duty cycling. (b) Basic Idea of our approach.

II. NETWORK MODEL AND PRELIMINARIES

We consider a field of size A , where N sensors are randomly deployed according to some probability distribution. This distribution is assumed to be uniform in our analysis in Sections III and IV, but is not required in the distributed algorithm presented in Section V. The node density is thus $\frac{N}{A}$.

We adopt a simple binary physical layer model by assuming that the transmission radius of each sensor is fixed at R , and an active node can reach any other active node within a circle of radius R . We also assume that links are bi-directional and symmetric. We will not be concerned with interference, as the scenario under consideration is one where traffic is relatively light, and the main goal is to have connectivity. The energy consumption is considered via the on time of a node in this paper, i.e., we assume that the main consumption comes from being active rather than from data relay and processing. Sensors are assumed to have a fixed amount of initial energy.

We illustrate the idea behind our approach via the following example. In the standard cluster-based construction of connected dominating sets (CDS), the size of the cluster is typically determined by the worst case scenario (i.e., the largest possible distance between two nodes in neighboring clusters). Figure 1(a) illustrates such an approach by [6], where the largest distance between nodes in two neighboring squares is the diagonal marked by the dark line. This distance has to be at most R in order for this construction to satisfy Conditions 1 and 2 when only squares adjacent to the four sides are considered neighbors. This constraint limits the size of the squares, which in turn limits the reduction in duty cycle.

Now consider the illustration given in Figure 1(b), where we have two neighboring squares, but only nodes within the shaded sub-squares are activated one at a time, while nodes outside the shaded areas are asleep. Again the worst case is marked by the dark line, which should be at most R , and the size of the bigger square should be such that any node outside the shaded area can directly connect to any node within the shaded area. In essence, we are limiting the on sensors to be closer to the center of the square. Thus the resulting backbone or CDS will appear more like a grid. By doing so, the bigger square can now cover a larger area without losing connectivity, which leads to lower average duty cycles for the nodes. Furthermore, it can be seen that Figure 1(a) is simply a special case of Figure 1(b) if we let the shaded square and the bigger square completely overlap. Subsequently, the shaded area within the bigger square will be referred to as a *sub-area*, and the bigger square will be referred to as a *cell* or *cluster* of the partition. The drawback of this approach is that since nodes are selected from a smaller area, the probability of being able to find one is also lower, which may potentially affect connectivity depending on the density. This is nevertheless

an attractive approach for a dense network. In addition, by generalizing the idea of clustering and making the size of the sub-area adjustable, we have greater flexibility in trading off duty cycle for performance.

Note that in the case of Figure 1(b), nodes are activated in a round robin fashion within a single sub-area. Periodically the field needs to be repartitioned using the same squares but with a proper offset in their coordinates. This ensures that a different set of nodes are now within sub-areas. This process then repeats to balance the energy consumption². However, in effect we only need one sensor to be on for the area covered by the bigger square.

In the next two sections, we will first examine a number of standard clustering methods and then present the partial clustering approach. Both are evaluated by the following two metrics:

- 1) *Average duty cycle D*: the average percentage time that a sensor is on or active.
- 2) *Probability of failure P_f*: the probability of finding no sensor in a cluster or a partial cluster.

III. CLUSTER BASED METHODS

The basic idea of standard clustering methods is to partition the sensing field into many cells. Sensors in the same cell form a cluster. The partition is chosen such that Conditions 1 and 2 are satisfied. Subsequently only one sensor in a cluster needs to be on at a time. The selection of which sensor should be on can be done in a round-robin fashion (with additional consideration of residual energy if applicable). What remains to be determined is the type of partition used to form clusters.

These are fundamentally centralized algorithms in that a global partition requires global knowledge of the field. Practical implementation of these approaches usually requires the sensors to have loose time synchronization and to have geographical information with respect to the entire field [7], [6]. The field is also often pre-configured into static cells. This allows each sensor to determine which cell it belongs to.

In the discussion that follows we will assume that some centralized entity controls the partition and the duty-cycling of sensors, and ignore extra overhead. We will also ignore edge effects that may occur when a particular partition does not constitute a complete tessellation of an area of a certain shape. The edge effects become more negligible as the field becomes larger compared to R .

Suppose the partition contains cells of average size a . Since only one sensor needs to be activated in each cell, there are $n_a = \frac{A}{a}$ total number of active sensors at a given time. Then the average duty cycle measure is given by $D = \frac{n_a}{N} = \frac{A}{aN}$, where N is the total number of nodes. The probability of failure is defined as follows: $P_f = (1 - \frac{a}{A})^N$.

A. Some Typical Partitions

The first partition is also used by [7], defined as follows.

²Note that such repartition is also needed in a standard clustering approach, since under any static clustering two clusters may contain very different number of nodes. Without repartition it could also lead to unbalanced energy consumption.

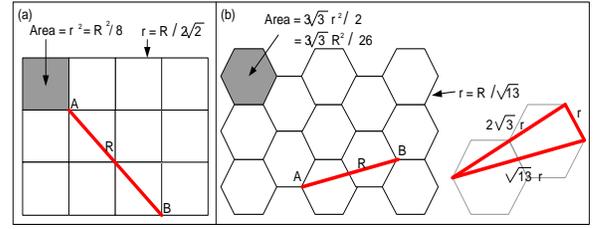


Fig. 2. (a) **S-8**: The field is partitioned into squares (cells) with side length $r = \frac{R}{2\sqrt{2}}$. (b) **Hex**: The field is partitioned into hexagons with side length $r = \frac{R}{\sqrt{13}}$.

Definition 1: (S-8) The field of interest is divided into equal squares (cells) as illustrated in Figure 2(a). Each square has a side length $\frac{R}{2\sqrt{2}}$ and area size $\frac{R^2}{8}$. For any square, its 8 surrounding squares are considered neighboring clusters.

It is obvious that **S-8** satisfies Conditions 1 and 2, and that among similar equal square tessellations this one achieves the largest cells. Since each cell has 8 neighbors, the minimum degree of the resulting connected routing graph is 8. (Degree is defined as the number of active neighbors of an active sensor.)

If we only consider 4 neighboring cells in the above, we obtain the following partition, also used in [6]:

Definition 2: (S-4) The field of interest is divided into equal squares (cells). Each square has a side length $\frac{R}{\sqrt{5}}$ and area size $\frac{R^2}{5}$. For any square, 4 squares adjacent to its sides are considered neighboring clusters.

It's obvious that this partition also achieves the largest cells among all square partitions satisfying the two conditions when only 4 neighbors are considered. The resulting connected routing graph has a minimum degree of 4.

Similarly we could use a hexagon tessellation. The following gives the largest cells among all hexagon tessellations that satisfy the two conditions.

Definition 3: (Hex) The field of interest is partitioned into equal hexagons as illustrated in Figure 2(b). Each hexagon has side length $\frac{R}{\sqrt{13}}$ and area size $\frac{3\sqrt{3}R^2}{26}$.

We summarize D and P_f of these partitions in Table I, which follow directly from the above definitions and simple trigonometry calculations. We can see that **S-4** has the best performance (note that we want both D and P_f to be small), although the difference between **Hex** and **S-4** is virtually negligible. On the other hand, the minimum degrees of **S-4**, **Hex**, and **S-8** are 4, 6, and 8, respectively. From a routing perspective, higher degree gives better and more robust routes and is thus preferred.

	S-8	S-4	Hex
P_f	$(1 - \frac{0.125R^2}{A})^N$	$(1 - \frac{0.2R^2}{A})^N$	$(1 - \frac{0.1999R^2}{A})^N$
D	$\frac{8A}{NR^2}$	$\frac{5A}{NR^2}$	$\frac{5.0037A}{NR^2}$

TABLE I

PERFORMANCE COMPARISON OF DIFFERENT PARTITIONS.

B. Order Optimality

Theorem 1: For the average duty cycle D , **S-8**, **S-4**, and **Hex** are asymptotically order optimal.

Proof: We utilize a result provided in [12] on the condition of asymptotic connectivity of a random graph. Suppose that there are N sensors uniformly distributed in a square with area A and side length l . Each sensor has communication radius R . Among these N sensors we randomly select n , which are again uniformly distributed. Asymptotic connectivity here means a sensor can find a path to any other sensor as $N \rightarrow \infty$, $A \rightarrow \infty$ while $\frac{N}{A}$ remains fixed. For the n chosen sensors to achieve asymptotic connectivity, [12] derives necessary condition $n = \beta \frac{A}{R^2}$, for some constant β . The proof of the necessary condition is done by considering point isolation probability, which is general and holds for all partitions. We now use this necessary condition to prove our result.

Following the necessary condition, we need at least $n = \beta \frac{A}{R^2}$ active sensors in order to have connectivity. This means that the duty cycle of the network D is lower bounded by $\frac{n}{N} = \beta \frac{A}{NR^2} = \theta(\frac{A}{NR^2})$. It follows from Table I that **S-8**, **S-4**, and **Hex** have $D = \theta(\frac{A}{NR^2})$, which is of the same order as the lower bound. ■

IV. PARTIAL CLUSTERING METHODS

As described in Section II, under partial clustering, the field is first partitioned into cells and then a sub-area is selected within each cell. In order to maintain connectivity, the partition and the sub-area selection need to satisfy the following 2 conditions.

Condition 3: Any sensor in a sub-area is connected directly to all sensors in the sub-areas within neighboring cells.

Condition 4: Within a cell, sensors outside the sub-area can communicate directly with any sensor in the sub-area.

Comparing partial clustering with clustering, denoting by D_p and D_c the respective average duty cycle, we have the following

$$D_p = \frac{n_p}{N}, \quad D_c = \frac{n_c}{N}, \quad (1)$$

where n_p and n_c are the number of cells in a partial clustering and standard clustering methods, respectively. Thus so long as $n_p < n_c$, partial clustering achieves lower average duty cycle.

A. A Few Examples of Partial Clustering

Definition 4: (P-S(y)) The field of interest is partitioned into equal squares (cells) with side length $\sqrt{R^2 - y^2} - y$, where $0 \leq y \leq \frac{R}{\sqrt{2}}$, as illustrated in Figure 3(a). The sub-areas are chosen as the shaded co-centered squares, as illustrated in Figures 3(a) and (b). For any square cell, the 4 square cells adjacent to its sides are considered its neighboring clusters.

From Figure 3(b), we can see that the largest distance from any sensor in a cell to a node in the sub-area of the cell is less than R . Furthermore, the largest distance between two nodes in neighboring sub-areas equals R . Thus both Conditions 3 and 4 are satisfied. Since x has to be non-negative and one can obtain that $x = \sqrt{R^2 - y^2} - 2y$ from Figure 3(b), the

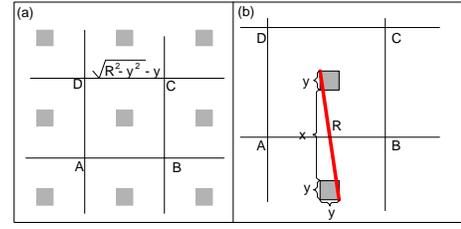


Fig. 3. (a) **P-S(y)**: One sensor needs to be chosen in each shaded sub-area. (b) Details of **P-S(y)**: Square ABCD is a cell.

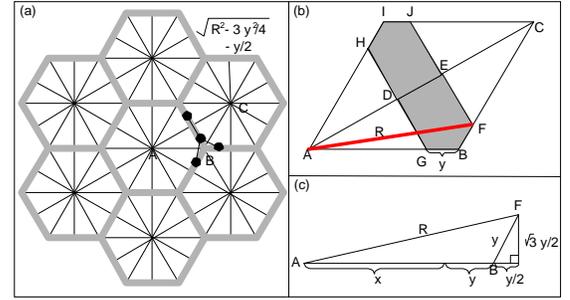


Fig. 4. (a) **P-H(y)**: An active sensor is chosen within the shaded area. (b) Details of **P-H(y)**: Triangle ABC is a cell. One node chosen in the shaded area DEF needs to be activated. (c) Triangle AFB can decide the range of y .

range of the tunable parameter y is $0 \leq y \leq \frac{R}{\sqrt{5}}$. The side length $\sqrt{R^2 - y^2} - y$ is also easily obtainable from Figure 3(b).

Definition 5: (P-H(y)) The field of interest is first partitioned into equal hexagons with side length $\sqrt{R^2 - \frac{3}{4}y^2} - \frac{y}{2}$, where $0 \leq y \leq \frac{R}{\sqrt{3}}$, and then each hexagon is further divided into 12 equal triangles, as illustrated in Figure 4(a). 2 neighboring triangles form 1 large triangle, which is the cell (e.g., the triangle ABC in Figure 4(a)(b)) in this partition method. The sub-area is shaded as illustrated in Figure 4(b) (e.g., shaded area DEF within cell ABC).

From Figure 4(b), we can see that the largest distance from any sensor in a cell to a sub-area node is R . Furthermore, the largest distance between sub-area nodes from neighboring cells is less than R (e.g., the largest distance from a sub-area node in DEF to a sub-area node in HIJED in a neighboring cell is less than R .) Thus this construction satisfies both Conditions 3 and 4. As an example, in Figure 4(a) the sub-area node (in black) of cell ABC is connected directly to 3 neighboring sub-area nodes. The range of y can be obtained from Figure 4(c). Since x is non-negative, we have $0 \leq y \leq \frac{R}{\sqrt{3}}$.

Proposition 1: The above two partial clustering approaches have duty cycles and failure probabilities as function of the tunable parameter y as follows:

$$D(y) = \frac{A}{NA_t(y)}, \quad (2)$$

$$P_f(y) = \left(1 - \frac{A_g(y)}{A}\right)^N, \quad (3)$$

where $A_t(y)$ is the size of the cell and $A_g(y)$ is the size of

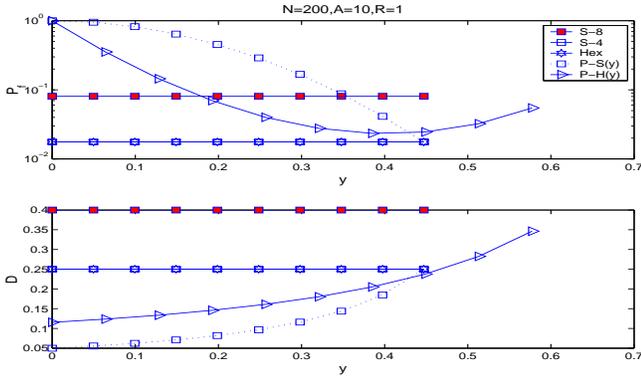


Fig. 5. P_f and D when $N = 200$, $A = 10$, and $R = 1$.

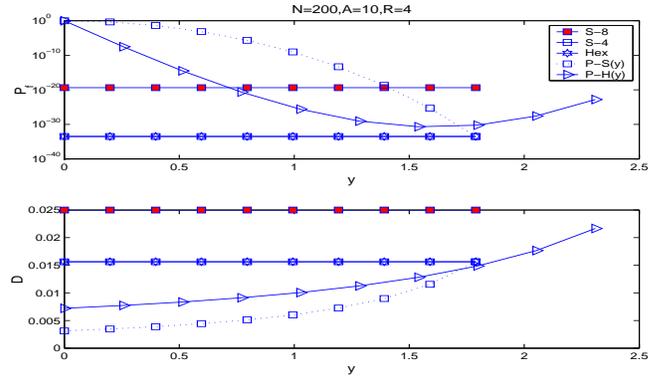


Fig. 6. P_f and D when $N = 200$, $A = 10$, and $R = 4$.

the shaded sub-area. Furthermore, we have for $\mathbf{P-S}(y)$,

$$A_t(y) = (\sqrt{R^2 - y^2} - y)^2, \quad (4)$$

$$A_g(y) = y^2, \quad 0 \leq y \leq \frac{R}{\sqrt{5}}. \quad (5)$$

For $\mathbf{P-H}(y)$:

$$A_t(y) = \frac{\sqrt{3}}{4} \left(\sqrt{R^2 - \frac{3}{4}y^2} - \frac{y}{2} \right)^2, \quad (6)$$

$$A_g(y) = \frac{\sqrt{3}y \left(\sqrt{R^2 - \frac{3}{4}y^2} - \frac{3}{2}y \right)}{2} + \frac{\sqrt{3}y^2}{4}, \quad (7)$$

where $0 \leq y \leq \frac{R}{\sqrt{3}}$.

For brevity, we do not present the detailed calculations here since they are relatively straightforward given the illustrations presented earlier. We note that for partial clustering, the measure P_f is only determined by A_g , the size of the sub-area, while the duty cycle is determined by the size of the cell since only one sensor needs to be on for the entire cell.

The following theorem can be established from the preceding definitions and the proof of Theorem 1.

Theorem 2: For the average duty cycle D , $\mathbf{P-S}(y)$ and $\mathbf{P-H}(y)$ are asymptotically order optimal.

B. Comparison and Discussion

Below we compare different approaches presented earlier. Figures 5 and 6 show P_f and D as functions of y under different duty-cycling approaches when $N = 200$, $A = 10$, and $R = 1$ and when $N = 200$, $A = 10$, and $R = 4$, respectively.

Not surprisingly, we see that both P_f and D decrease as R increases from 1 to 4, and they decrease as N increases as well. Note that here we consider only the duty cycle not the actual energy consumption; increasing R would increase the transmission power consumption as well.

In general partial clustering approaches achieve lower D than clustering at the expense of higher P_f . For a partial clustering method, as we adjust parameter y , D and P_f change in opposite directions. For example, for $y = \frac{R}{\sqrt{5}}$ $\mathbf{P-S}(y)$ is equivalent of $\mathbf{S-4}$. As we decrease y , using $\mathbf{P-S}(y)$ attains a lower D but higher P_f than using $\mathbf{S-4}$. On the other hand, for the very same reason the class of partial clustering methods

exhibit greater flexibility and offers more control over the trade-off between D and P_f .

It's worth mentioning that P_f diminishes exponentially fast as the total number of nodes increases. As a result, in dense networks partial clustering methods can achieve much lower D at the expense of very slight increase in P_f compared to clustering methods. Therefore partial clustering methods are particularly attractive in a dense network.

Note that both clustering and partial clustering approaches encompass many different variations. The ones presented here are more or less based on typical tessellations. It's worth emphasizing that the idea underlying partial clustering is to introduce extra structure (e.g., the "grid" structure in $\mathbf{P-S}(y)$, the hexagonal or "ring" structure in $\mathbf{P-H}(y)$) to the standard clustering method in order to obtain either lower duty cycle or better trade-off.

Finally both clustering and partial clustering methods as presented above are centralized algorithms, and designing corresponding distributed implementations can be very difficult and complicated. To show the complexity note that finding a cluster with only Condition 1 is the same as the CLIQUE problem which is known to be NP-complete.

As mentioned before, distributed algorithms based on standard clustering require sensors to have global geographical information. On the other hand, under partial clustering the additional structure introduced into the connected set gives rise to simple heuristics in designing a decentralized algorithm which only requires relative location information. We explore this further in the next section.

V. DISTRIBUTED PARTIAL CLUSTERING

In this section we present a distributed algorithm based on the partial clustering concept, under which nodes coordinate to decide who should be on. The main assumption we will make is that a node has knowledge about the positions of its neighbors relative to itself. We do not require nodes to be clock-synchronized.

Depending on whether we follow the hexagon-based $\mathbf{P-H}$ or the square-based $\mathbf{P-S}$ constructions, different variations of the algorithm may be derived. The resulting algorithms will be denoted DPC-H and DPC-S, respectively. Here we will use the former for presentation purposes, while we include simulation results on both in the next section.

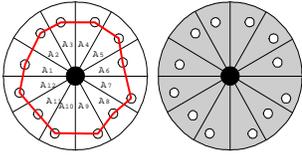


Fig. 7. Partition of the communication area of the head node (in dark) and the connected ring of supports (in white). All non-support sensors inside the big circle are potential heads.

The key to the distributed partial clustering algorithm is illustrated in Figure 7. As mentioned earlier, the hexagon-based partial clustering essentially attempts to form a “ring” of active nodes (12 of them to be precise) around an area in which all nodes can go to sleep. If a node can find such a ring of nodes (subsequently called *supports*) to be active, then the node can safely switch off (such a node is subsequently called a *head*). Once this ring of supports are identified, all other nodes surrounded by this ring can also become heads and switch off. The head node will be off for a pre-specified period of time and wake up; the ring of supports will be relieved of their role, and the process will repeat. A node that is neither a head nor a support will be called a *regular* node.

We proceed to discuss key elements of this algorithm in more detail. To maintain clarity, we will largely keep the description on a conceptual level and some of the details of the algorithm are left out.

A. Basic Operations

A regular node i periodically runs a head selection function (to be described in more detail in Section V-B) to determine whether the required number of supports can be located. If the process succeeds, it sends out a packet PKT-H containing the source ID, the set of selected nodes, and the time until switching off. Due to the broadcast nature of the transmission, this information is known to all active nodes within the neighborhood. The selected supports subsequently respond with a PKT-S packet announcing their role as supports. At the scheduled time i becomes a head and goes to sleep for a period of time specified by a constant T_s (the supports will be awake for the same period of time). Upon expiration of this period, node i wakes up and it (along with its supports) will reset itself to a regular node. The whole process then repeats. If the head selection is not successful, then node i continues to be a regular node, unless it is selected to be a support by some other node.

Sensors also use a PKT-E as a keep-alive packet to maintain a neighbor list. This packet can also be used to deliver information such as residual energy, etc.

All packet transmission times are offset by a randomly selected small delay in order to avoid synchronous transmissions. In particular, to de-synchronize neighboring nodes from trying to simultaneously selecting each other to be supports, from the time node i becomes a regular node, it has a window of T_c units of time to initiate and execute the head selection function. A regular node will randomly select a starting time to execute the head selection function within this window. The selection of this starting time can also be based on consideration of

residual energy (e.g., the less the energy the sooner to start so the node may have a better chance of becoming a head), in ways similar to that shown in Equation (8) below. In our simulation we have set $T_s = 10T_c$.

It remains a possibility that two neighboring sensors (e.g., sensors i and j) execute head selection function at approximately the same time, where sensor i may choose sensor j as the support while j decides to be a head. As long as j receives PKT-H from i it will become a support instead. However, if the packet is lost due to error or collision, then i may end up with fewer number of supports than expected unless error correction or retransmissions are used.

B. Head Selection Function

A sensor is eligible to be a head if it can locate 12 neighbors, each chosen within a cell of the partition, denoted by A_i as shown in Figure 7. Note that all non-support sensors inside the grey circle (the communication area of the dark head node) are potential heads. Any potential head i can become a head if it satisfies the following condition:

Condition 5: All its neighbors outside the circle are directly connected to at least one of the announced supports within the circle.

Testing this condition requires knowledge of the relative position of neighbors as well as the broadcast of PKT-H and PKT-S. Since any sensor in cell A_i can communicate with the chosen support in A_i directly and the 12 chosen supports are connected, the connectivity within the communication area of the head is maintained if all 12 supports remain active. The connectivity between neighboring (possibly overlapping) communication areas of different heads is maintained since the above condition is satisfied.

In order to have as many qualifying heads as possible, a potential head node should choose supports as far away as possible. In order to balance energy consumption, it’s desirable to choose supports with higher residual energy. It also makes sense to select as supports the nodes that are already chosen as supports by other heads.

There are many ways of selecting supports. In our simulation, we adopted the following simple formula of a weighted sum of distance and residual energy. This is calculated for each of the 12 cells by a potential head:

$$\text{support node} = \operatorname{argmax}_{i \in N_S} \left\{ \kappa \frac{d_i}{R} + \mu \frac{E_i}{E} \right\}, \quad (8)$$

where N_S is the set of sensors in the current cell that have already been chosen as supports, d_i is the distance between node i and this head node, E_i is i ’s residual energy, constant E is the initial energy, $\kappa + \mu = 1$ and $0 \leq \kappa, \mu \leq 1$. If N_S happens to be empty, then we repeat the same calculation over the set of all sensors in the current cell.

The distributed partial clustering algorithm based on construction $\mathbf{P-S(y)}$, DPC-S, is very similar to DPC-H that we just described. The key difference is that in the head selection process a potential head node selects 4 supports based on the illustration given in Figure 3.

The routes established based on the connected set generated by DPC-H are longer than that may be established when all

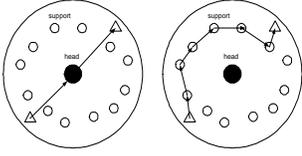


Fig. 8. An example of path length increase from 2 to 7 links. The triangular nodes are intermediate sensors of the data path.

sensors are on. But they are only longer by a constant factor in the worst case. This is illustrated in Figure 8, where the triangular nodes and the head are the intermediate sensors of the original data path. When the head is turned off, the data path needs to be re-routed through the supports. In the worst case the path length is increased from 2 links to 7 links. Therefore, in the worst case the increase of path length due to DPC-H has a constant factor 3.5. Similarly, in the worst case the increase of path length under DPC-S has a constant factor 2.

VI. PERFORMANCE EVALUATION

We simulate DPC-H and DPC-S in Matlab and compare their performances to the ideal results obtained using centralized algorithms.

The field of interest is a square with length L and size $A = L^2$. N sensors are deployed according to the uniform distribution. The sensor transmission radius is R , with $R \ll L$. The following energy model from [13] is adopted. Energy consumption on packet transmission and on reception are $\alpha_{11} + \alpha_2 d_{1,2}^2$ J/bit and α_{12} J/bit, respectively, where $\alpha_{11} = 45n$ J/bit, $\alpha_{12} = 135n$ J/bit, $\alpha_2 = 10p$ J/bit/m², and $d_{1,2}$ is the distance between the transmitting and receiving nodes. We assume that there is no energy consumption when a sensor is off and the idle energy consumption in the on state is $35mW$ [14]. We will also assume all transmissions are error/collision free. This makes the results comparable to the ideal theoretical results derived earlier.

Each sensor will execute the algorithm every $T_c + T_s$ duration. We call this duration a “round”. The initial energy is set such that each sensor can continuously run for 120 rounds without being turned off. A sensor dies when it has depleted all its energy. The sizes of packets P_H , P_S , and P_E are 13 bytes, 1 bytes, and 2 bytes, respectively. The shortest distance from sensor i to the field boundary is denoted by D_i . The results are obtained from sensors located with $D_i \geq R$ to remove the boundary effect and make the results comparable. The simulation results shown are the average of 30 random topologies.

A. Energy Efficiency and Energy Balancing

We denote the total on duration by $T_{on}(i)$ and total off duration by $T_{off}(i)$ of each sensor i . These are obtained before the first sensor’s death. Then we calculate the average duty cycle as $\frac{1}{N} \sum_{i=1}^N \frac{T_{on}(i)}{T_{on}(i) + T_{off}(i)}$, and its standard deviation. The left graph in Figure 9 compares the average duty cycles of DPC-H, DPC-S, and theoretical results on approaches presented in Sections III and IV. In the graph the upper and

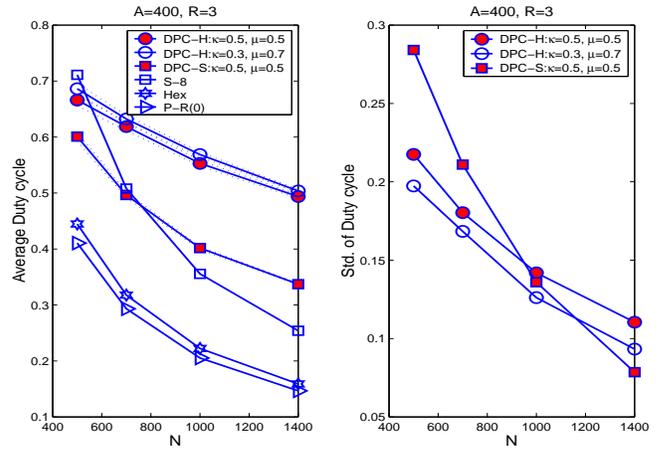


Fig. 9. Left: comparison of average duty cycles. The upper and lower dotted lines of a simulated average duty cycle (the thick line with markers) are the 95.4% confidence interval. Right: the standard deviations of duty cycles.

lower dotted lines of a simulated average duty cycle (the thick line with markers) are the 95.4% confidence interval. In all cases the duty cycle decreases as N increases.

The main observation here is the difference between theoretical centralized approaches and distributed approaches. The distributed algorithms result in much higher duty cycles, but seem to scale as well as the centralized algorithms as N increases. This difference is expected as distributed algorithm cannot achieve a perfect field partition as shown in Sections III and IV. Furthermore, the results of the theoretical centralized approaches do not include any control overhead.

There is no significant difference between the 2 sets of parameters for DPC-H. The choice of these control parameters is subject to further study. Between DPC-H and DPC-S, the latter performs better. The reason seems to be that the latter results in fewer number of regular nodes and supports.

Looking at the right graph in Figure 9 we see that for all these algorithms energy balancing improves as N increases. (The standard deviation for the ideal centralized algorithms will all be zero, thus not shown in the figure.)

We also obtain the times when 20% sensors are dead under DPC-H and DPC-S, respectively shown in the left graph in Figure 10. Similar to previous results, DPC-S performs the best.

B. Complexity and Control Packets Overhead

The complexity of DPC-H and DPC-S is $\Theta(M_i)$ for sensor i per round, where M_i is the number of neighbors sensor i has. This is because in the head selection function partitioning neighbors into 12 sub-areas has complexity $\Theta(M_i)$ and Equation (8) has the same complexity. Furthermore, checking Condition 5 has complexity $\Theta(M_i)$.

If a sensor executes the head selection function and decides to be the head, our algorithm generates 1 PKT-H and a number of PKT-S (12 for DPC-H and 4 for DPC-S). Each sensor will generate 1 PKT-E packet every round. The average control overhead per round obtained from simulation is shown in the right graph in Figure 10. Control traffic increases as

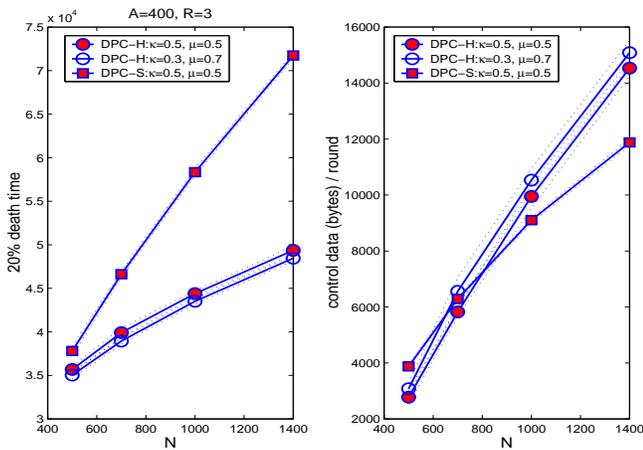


Fig. 10. Left: the time when 20% sensors are dead; Right: average amount of control overhead per round before the first sensor death. The upper and lower dotted lines of a simulated average duty cycle (the thick line with markers) are the 95.4% confidence interval.

N increases since the number of sensors running the head selection function increases. The increasing rate is less than linear. DPC-H has more control traffic than DPC-S since it generates more PKT-S.

C. Comparison with SPAN

In this section we briefly compare our algorithms with SPAN, a pure virtual backbone approach proposed by [2]. The basic idea of SPAN is for a node to stay on if it detects that it has at least two neighbors that do not share common neighbors other than itself. This algorithm does not require nodes to have the relative location information of its neighbors as required by our algorithms. But it requires a node to have the list of all nodes within two hops, which results in a large amount of comparisons with a complexity $O(M^4)$ where M is the average node degree³.

The scenarios simulated in Section VI-A have node degrees that are too large to run SPAN in Matlab. We thus reduce the field size ($A = 64$), transmission radius ($R = 1$), and number of sensors (N). From the simulation results, SPAN has an average duty cycle 0.52 when $N = 100$ (about 1–2 nodes in a single transmission circle); however, DPC-S with $\kappa = \mu = 0.5$ has an average duty cycle 0.98. This is due to the obvious reason that when the density is too low, the requirement of 4 supports is almost never satisfied. The performance of DPC-S improves as N increases; when $N = 1200$, DPC-S has an average duty cycle 0.43. On the other hand, we were not able to obtain results for SPAN when $N > 200$ due to the complexity.

VII. CONCLUSION

In this paper we introduced the concept of partial clustering to construct connected dominating sets in a dense wireless sensor network, in order to achieve connectivity and low duty cycle. We examined three cluster-based approaches and

compared them with two partial clustering methods in terms of average duty cycle and the probability of failure. We showed that all these schemes achieve asymptotically order optimal average duty cycles. We then developed a distributed algorithm based on the partial clustering idea and showed its performance via simulation. We demonstrated that partial clustering is an attractive method of construction especially for dense networks, as it provides low duty cycle and greater flexibility in the trade-off between duty cycle and failure probability.

REFERENCES

- [1] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *Proceedings of the Third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 1999.
- [2] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, 2001.
- [3] Bevan Das and Vaduvur Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *IEEE International Conference on Communications (ICC)*, 1997.
- [4] Peng-Jun Wan and Khaled M. Alzoubi Ophir Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in *Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.
- [5] J. Mark Keil, "The complexity of domination problems in circle graphs," *Discrete Applied Mathematics*, 1993.
- [6] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, 2001.
- [7] Paolo Santi and Janos Simon, "Silence is golden with high probability: Maintaining a connected backbone in wireless sensor networks," in *Proc. IEEE European Workshop on Wireless Sensor Networks*, 2004.
- [8] Douglas M. Blough and Paolo Santi, "Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks," in *ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, 2002.
- [9] Chaiporn Jaikaeo and Chien-Chung Shen, "Adaptive backbone-based multicast for ad hoc networks," in *IEEE International Conference on Communications (ICC)*, 2002.
- [10] Alec Woo and David E. Culler, "A transmission control schemes for media access in sensor networks," in *ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, 2001.
- [11] Kaixin Xu, Xiaoyan Hong, and Mario Gerla, "An ad hoc network with mobile backbones," in *IEEE International Conference on Communications (ICC)*, 2002.
- [12] Paolo Santi and Douglas M. Blough, "The critical transmitting range for connectivity in sparse wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, 2003.
- [13] Manish Bhardwaj and Anantha P. Chandrakasan, "Bounding the lifetime of sensor networks via optimal role assignments," in *Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.
- [14] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2002.

³This is obtained from the pseudo-codes provided in [2]. [7] also makes a similar claim that SPAN does not scale with sensor density.