LORD: A Localized, Reactive and Distributed Protocol for Node Scheduling in Wireless Sensor Networks [*]

Arijit Ghosh and Tony Givargis
Center for Embedded Computer Systems
Department of Computer Science
University of California, Irvine, CA 92697
{arijitg,givargis}@ics.uci.edu

Abstract

The lifetime of wireless sensor networks can be increased by minimizing the number of active nodes that provide complete coverage, while switching off the rest. In this paper, we propose a distributed and scalable node-scheduling algorithm that conserves overall system energy by minimizing the number of active nodes, localizing the execution to the dying sensor(s), and minimizing the frequency of execution by reacting only to the occurrence of a sensing hole. This effects an increased system lifetime while maintaining coverage over an application-defined threshold value. We compare our algorithm to a network with a centralized node-scheduling algorithm. Our results show equivalent coverage degree over a wide range of sensor networks.

Keywords

Wireless Sensor Network, Coverage, Set Cover

1. Introduction

The technological advancements in the areas of micro-electro-mechanical systems, wireless communications and digital microelectronics have ushered in the era of wireless sensor networks (WSN). The potential applications of these WSNs are innumerable. Geographical applications include detecting environmental hazards, monitoring remote terrain, wildlife observation and provide vigil over battlefield and disaster areas. Personal and societal applications include integrated patient monitoring, diagnostics and drug administration, real-time traffic patterns and personalized shopping amongst others. Other applications include robotic control in manufacturing process, warehouse inventory and environmental control in public places [7][16].

The diversity in application and the variation in operating conditions necessitate the formulation of new system design approaches. In particular, energy is a major design bottleneck and power-aware system design techniques are required at all levels of the system [3][12]. One of the critical design objectives is to increase the overall network lifetime [15]. It is highly impractical to achieve this by battery replacement/recharge as sensors are usually inaccessible. Instead, system energy conservation by reduction of active nodes has been proposed [14][17]. The high degree of redundancy is exploited to activate a reduced subset of nodes that provide the same degree of coverage.

In this work, we propose a distributed node-scheduling protocol that achieves all of the following goals. First, the protocol maintains coverage over an application-specific threshold value. Second, the number of scheduled nodes is minimized. Third, the protocol is localized. It is executed only at the point of the dying sensor. This precludes unnecessary participation of most sensors that have not had a change in their state in protocol execution, thereby saving more energy. Fourth, the protocol is reactive. It is scheduled to execute only when a change in the network state could potentially lead to a decrease in the coverage degree. This is very important as computation (in finding the schedule) and the ensuing communication (required for sending the schedule), causes a drain in the overall system energy. By minimizing the number of runs of the algorithm, we contribute an additional degree of energy saving of the whole system that helps in further extending the system lifetime.

The rest of the paper is organized as follows. In Section 2 we discuss previous work. In Section 3, we outline our proposed protocol. In Section 4, we present our experimental results. Finally, we present our conclusions in Section 5.

[*] This work was supported by NSF Grant # CCR-0205712.

2. Previous work

Conserving energy by minimizing active nodes to increase system lifetime has been an active area of research. There have been two objectives for node-scheduling: providing coverage and maintaining connectivity.

Several coverage algorithms have been proposed in literature. In [2], [11], the authors propose a linear programming based technique for a choice of a minimal set of active nodes. In [9], the authors combine computational geometry and graph theoretic techniques for coverage calculation. The maximal breach path and the maximal support path in a sensor network are computed using Voronoi diagram and Delaunay Triangulation techniques. In [10], the same problem has been solved by computing the minimal exposure path. The authors in [5] discussed strategies for node deployment that provide distributed detection. In [20], a probing-based density control algorithm is proposed. Sleeping nodes wake up occasionally to probe their local neighborhood and start working if no live sensors are detected. In [15], a coverage-preserving node-scheduling algorithm is proposed. A basic model for coverage-based off-duty eligibility rule is presented which allows a sensor to see if its sensing area is covered by the union of the sponsored sectors of its neighbors.

Similarly, many algorithms have been proposed that maintain connectivity. In [1], each node assesses its connectivity and adapts its participation in the multi-hop network topology based on the measured operating region. In [4], nodes are turned off based on the necessity for neighbor connectivity. In [18], nodes turn off their communication unit when they are not involved in sending, forwarding or receiving data phases. In [19], the proposed algorithm uses geographic location information to divide the sensing area into fixed square grids. Within each grid, there is just one node that is awake and participates in data forwarding.

Some other algorithms have been proposed for node scheduling which maintains both coverage and connectivity. In [21], the authors proposed an algorithm that allows a configurable approach to maintain both coverage and connectivity, without any guarantee on the degree. In [17], the protocol can dynamically configure the network to provide different degrees of coverage while maintaining connectivity. It proposed an eligibility rule in which sensors can turn themselves off if all intersecting points between the borders of its sensing radii are covered with the desired degree.

Finally, in addition to coverage and connectivity, some work have considered the impact a node-scheduling algorithm will have on other sensors in the network, notably from a routing perspective. In [8], the algorithm considers an additional set of routers over a minimum set of sensors necessary to both cover a region and forward the data. In [13], the algorithm used the concept of "application cost", which is based on a sensor's as well as its neighbors' residual energy, and used it to decide the set of active sensors.

While the above-mentioned algorithms achieve coverage or connectivity or both, they run at random times. Depending on when this is invoked, it could result in either sensing holes or unnecessary drain of energy. In addition, all sensors execute the algorithm, thus causing a further reduction of overall system energy. In our approach, we optimize the location and frequency of execution. The protocol runs, at the point where and at the instance when, the sensing hole is about to appear. These optimizations are added advantages inasmuch that we minimize the number of active sensors and maintain both coverage and connectivity.

3. Technical approach

3.1. Preliminaries

We consider a sensor network with η sensor nodes deployed over a continuous, arbitrary shaped area $\mathcal{A}$. Let $\mathcal{F}$ be the set of sensor nodes, $\mathcal{F} = \{s_1, s_2, \ldots, s_\eta\}$. Let $s_i.area$ be the finite, continuous and arbitrary-shaped area that can be sensed by sensor s_i.

We present a model to quantize a continuous region in a 2-D space. A 2-D space is partitioned into an imaginery grid with an arbitrarily chosen point of origin. Given the grid, any arbitrary shaped area is represented as a set, containing the coordinates of the cells that are fully or partially covered by the area. The set representation allows many operations on the area (e.g. overlap) to be expressed in terms of set operations (e.g. intersection). This improves on space and time complexity as sets can be efficiently represented and manipulated in computer systems. Additionally, it eliminates the imposition of any restriction on the shape of the sensing area or the deployment area. Further, the granularity of the grid is application-specific and can be varied to trade off between accuracy and computational complexity.

3.2. Problem formulation

The node-scheduling problem is to identify a minimal subset $\mathcal{C}$ of $\mathcal{F}$ that covers $\mathcal{A}$. More specifically, compute $\mathcal{C}$:

$$\mathcal{A} \subseteq \bigcup_{\substack{s_i \in \mathcal{C} \; and \\ \mathcal{C} \subseteq \mathcal{F} \; and \\ |\mathcal{C}| \; is \; minimal}} s_i.area.$$

This problem closely resembles the set-covering problem which is a generalization of the NP-complete vertex-cover problem and is therefore NP-hard[6]. There are well-known centralized heuristics for the set-cover problem. However, none of those solutions scale well when applied to large sensor networks. As such, we propose a

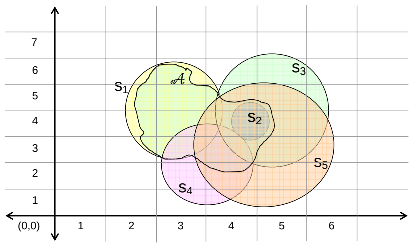

Figure 1. Coverage area with 5 sensors.

*lo*calized, *r*eactive and *d*istributed network coverage protocol (LORD), that despite being a heuristic, scales extremely well with the network size producing comparably good results. We compare our heuristic to an efficient centralized approximation algorithm.

3.3. The LORD protocol

LORD consists of two phases, the set-up phase and the steady phase. In the set-up phase, *all* the sensors individually run a heuristic algorithm to decide their initial state. Every ON sensor continues running until it either dies or receives an intimation to be turned off. In the steady phase, all *dying sensors* run a localized heuristic algorithm and nominate the sleeping sensors that can cover for themselves. The simple protocol of neighbor set-up and immediate participation ensures that scalability, both in terms of the monitored area and the number of sensors, can be achieved.

We will consider the example area $\mathcal{A}$ as shown in Figure 1. We consider 5 nodes, $s_1 \ldots s_5$, deployed over $\mathcal{A}$.

Our protocol assumes a perfect and reliable communication layer. Communication is achieved by three primitives. *Send* allows communication between any two sensors in the network. *Broadcast* allows communication between two sensors who are one hop away. Two sensors whose sensing areas overlap are considered to be one hop away from each other and can communicate with each other directly [15]. *Receive* allows messages to be received at any sensor. The primitives together implement a synchronization mechanism, the details of which have been excluded for brevity.

3.3.1. Set-up phase. When the network is being deployed, the set-up phase runs at each sensor s_i to determine its initial state (Algorithm 2). The current state of a sensor is denoted by $s_i.state$, and is either OFF or ON.

Each sensor first builds up its list of neighbors, $s_i.NL$ (Algorithm 2, Line 5). A sensor s_i is a neighbor of sensor s_j, iff their sensing areas overlap.

$$s_i.NL = \bigcup_{\substack{s_i.area \,\cap\, s_j.area \,\neq\, \varnothing \\ i \neq j}} \{s_j\}.$$

This is achieved by broadcasting a mesage containing its sensing area and listening for similar messages. Every sensor s_j, from which s_i receives a message, is added to its neighbor list $s_i.NL$ (Algorithm 1). The sensor s_i saves all its neighbors' sensing areas as they are used subsequently in the set-up phase as well as the steady phase. For example, $s_1.NL = \{s_3, s_4, s_5\}$.

Algorithm 1 create_neighbor_list()

```
1: $s_i.state \leftarrow$ OFF
2: $s_i.NL \leftarrow \varnothing$
3: broadcast($s_i.area$)
4: while not_timed_out() do
5:     $s_j.area \leftarrow$ receive()
6:     $s_i.NL \leftarrow s_i.NL \cup \{s_j\}$
7:     save($s_j.area$)
8: end while
```

The sensor s_i defers its decision on its intial state until all sensors s_j such that $s_j \in s_i.NL$ and $j < i$ have made their decisions (Algorithm 2, Lines 13-26). This simple heuristic serializes the decision making amongst sensors thereby completely eliminating assumptions between sensors on their respective states. The ordering amongst sensors can be easily imposed by leveraging the underlying grid, making it easily scalable to addition of new nodes. Simultaneously, the sensor s_i also updates its Unique Area, $s_i.UA$ (Algorithm 2, Lines 20-21). At any time t, $s_i.UA$ represents that part of the sensing area of s_i which is not covered by any of its "*ON*" neighbors.

$$s_i.UA = s_i.area - \bigcup_{\substack{s_j \,\in\, s_i.NL \ and \\ s_j.state \,=\, ON}} s_j.area$$

The sensor s_i decides to turn itself ON iff its sensing area covers some part of the coverage area $\mathcal{A}$ that is not covered by any other sensor at that instance of time, i.e., $s_i.UA > 0$ (Algorithm 2, Lines 27-31). In our example, s_1 and s_2 can make their decisions as soon as they build up their neighbor lists. However, s_3 has to wait for both s_1 and s_2, s_4 has to wait for s_1, s_2 and s_3 etc. From our definition of Unique Area, we note that all sensors from s_1 to s_4 have to turn themselves ON while s_5 can turn itself OFF.

Once all the remaining neighbors of sensor s_i have determined their states (Algorithm 2, Lines 33-41), it is possible that sensor s_i is redundant as all of its sensing area is covered by some combination of its neighbors. To eliminate this redundancy, each ON sensor s_i reevaluates its initial decision. Again, to remove ambiguity, sensor s_i waits till all its ON neighbors s_j such that $j < i$ have completed their reevaluation (Algorithm 2, Lines 42-50). This ensures that no two sensors make any assumptions about the state of the other, thereby eliminating the possibility of a sensing hole. Sen-

Algorithm 2 self_determination()

```
 1: $temp_{\text{LOW}} \leftarrow \varnothing$
 2: $temp_{\text{HIGH}} \leftarrow \varnothing$
 3: $temp_{\text{ON}} \leftarrow \varnothing$
 4: $s_i.UA \leftarrow s_i.area$
 5: create_neighbor_list()
 6: for every $s_j$ in $s_i.NL$ do
 7:     if $j < i$ then
 8:         $temp_{\text{LOW}} \leftarrow temp_{\text{LOW}} \cup \{s_j\}$
 9:     else
10:         $temp_{\text{HIGH}} \leftarrow temp_{\text{HIGH}} \cup \{s_j\}$
11:     end if
12: end for
13: while $|temp_{\text{LOW}}| > 0$ do
14:     $s_j.state \leftarrow$ receive()
15:     if $s_j \in temp_{\text{LOW}}$ then
16:         $temp_{\text{LOW}} \leftarrow temp_{\text{LOW}}$ - $\{s_j\}$
17:     else
18:         $temp_{\text{HIGH}} \leftarrow temp_{\text{HIGH}}$ - $\{s_j\}$
19:     end if
20:     if $s_j.state$ = ON then
21:         $s_i.UA \leftarrow s_i.UA$ - $s_j.area$
22:         if $j < i$ then
23:             $temp_{\text{ON}} \leftarrow temp_{\text{ON}} \cup \{s_j\}$
24:         end if
25:     end if
26: end while
27: if $|s_i.UA| > 0$ then
28:     $s_i.state \leftarrow$ ON
29: else
30:     $s_i.state \leftarrow$ OFF
31: end if
32: broadcast($s_i.state$)
33: while $|temp_{\text{HIGH}}| > 0$ do
34:     $s_j.state \leftarrow$ receive()
35:     if $s_j \in temp_{\text{HIGH}}$ then
36:         $temp_{\text{HIGH}} \leftarrow temp_{\text{HIGH}}$ - $\{s_j\}$
37:     end if
38:     if $s_j.state \leftarrow$ ON then
39:         $s_i.UA \leftarrow s_i.UA$ - $s_j.area$
40:     end if
41: end while
42: while $|temp_{\text{ON}}| > 0$ do
43:     $s_j.state \leftarrow$ receive()
44:     if $s_j.state$ = OFF then
45:         $s_i.UA \leftarrow s_i.UA$ + $s_j.area$
46:     end if
47:     if $j < i$ then
48:         $temp_{\text{ON}} \leftarrow temp_{\text{ON}}$ - $\{s_j\}$
49:     end if
50: end while
51: if $s_i.state$ = ON and $|s_i.UA|$ = 0 then
52:     $s_i.state \leftarrow$ OFF
53:     broadcast($s_i.state$)
54: end if
```

sor s_i recomputes its $s_i.UA$ and if that is equal to zero, s_i can turn itself off. (Algorithm 2, Lines 51-54). In our example, s_2, on reevaluation, finds that $s_2.UA$ is equal to zero and hence, turns itself off. In our protocol, we assume that all OFF sensors periodically wake up so that they can *receive* messages from their ON neighbors [17].

By serializing this minimization step and restricting it to ON sensors, it is guaranteed that the minimization always ends after one iteration as only the sensors s_i for which $s_i.UA$ is greater than zero is on.

3.3.2. Steady phase. The steady phase algorithm is invoked, at and when, each sensor s_i dies. We assume that s_i can internally monitor its energy state and run its own replacement algorithm before it dies completely. The algorithm generates a set, $s_i.replace$, consisting of a minimum number of sleeping neighbors, that can cover $s_i.UA$. In our example, s_3 dies at t=3. Hence the steady phase algorithm is invoked at s_3 at t=3.

Algorithm 3 find_replacement()

```
 1: At: Each dying sensor $s_i$
 2: $temp \leftarrow \varnothing$
 3: $s_i.replace \leftarrow \varnothing$
 4: $s_i.UA \leftarrow s_i.area$
 5: for each $s_j \in s_i.NL$ do
 6:     if $s_j.state$ = OFF then
 7:         $temp \leftarrow temp \cup \{s_j\}$
 8:     else
 9:         $s_i.UA \leftarrow s_i.UA$ - $s_j.area$
10:     end if
11: end for
12: $s_i.replace \leftarrow$ compute_MCS($s_i.UA$, $temp$)
13: for each $s_j \in s_i.replace$ do
14:     send($s_j$, “ON” );
15: end for
```

$$s_i.UA \subseteq \bigcup_{\substack{s_j \,\in\, s_i.replace \ and \\ s_j.state \,=\, OFF \ and \\ s_i.replace \,\subseteq\, s_i.NL \ and \\ |s_i.replace| \ is \ minimal}} s_j.area$$

The minimum set cover(MSC) algorithm [6] is used to generate a close approximate solution. It then sends a message to all $s_j \in s_i.replace$ (Algorithm 3). In our example, s_2 and s_5 are the sleeping neighbors of s_3. By running the MSC, we compute $s_3.replace = \{s_5\}$.

The waking up of one or more sensors could now generate redundant active sensors. Hence, the same reevaluation as outlined in the in the set-up phase is run at *only* the ON neighbors of every $s_j \in s_i.replace$, thereby turning OFF all redundant sensors. In our example, the waking up of sensor s_5 as a replacement for s_3 makes sensor s_4 redundant. Hence, the sensor s_4 can turn itself off to conserve energy.

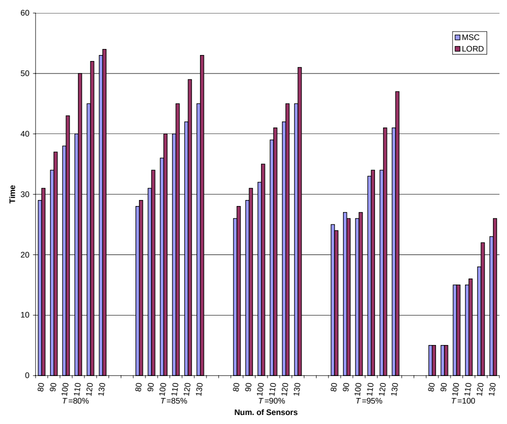

Figure 2. System lifetime: random life.

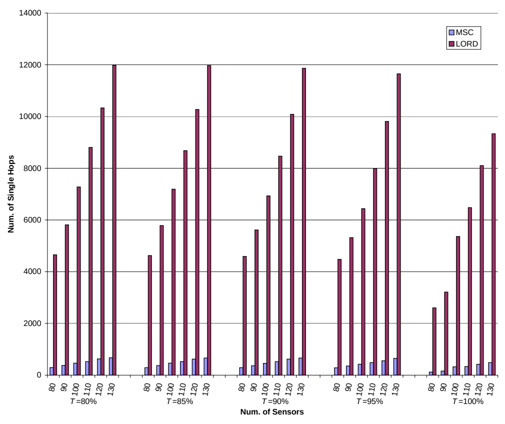

Figure 3. Communication: random life.

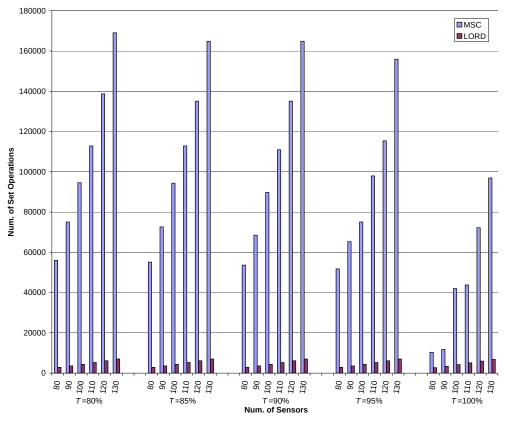

Figure 4. Computation: random life.

By providing a strict upper bound (equal to the total number of dying sensors) on the number of invocations of the steady phase algorithm and by localizing the execution, significant energy savings can be done by reducing both communication and computation.

4. Experimental results

For our experiments, we consider an area 50m $\times$ 50m. We set the granularity of the grid to 1 cm which produces a quantization error of less than 1%. We consider circular sensors placed randomly over this area. The sensing area of each is equal to its communication area and both are set to be a circle of radius 10m. This is done to simplify the experiments and has no effect on the protocol. We deploy 80, 90, 100, 110, 120 and 130 sensors. We assume that the communication is reliable and that all sensors lose their energy at a uniform rate. Observing that certain applications might require a "best-effort" coverage, we set a coverage threshold *T*, that represents the percentage of the total number of grids that can be covered by the available sensors.

We compare our algorithm to a highly efficient centralized MSC algorithm [6] that runs in an imaginary super node which has unlimited power and complete global knowledge. MSC is a greedy approximation algorithm with a logarithmic ratio bound and returns a set cover not too much larger than the optimal set cover.

Our comparisons are made on three points. First, we compare the life of the network. This directly reflects the energy saved. Second, we compare the cumulative communication cost, which is the number of message transmissions T_x, over the entire life. While T_x is the number of hops between source and destination for MSC, it is the sum of the neighbors of all s_i that change state in LORD. Since the payload of the transmitted message is constant in both cases, we ignore this for our comparisons. Third, we compare the cumulative computation costs over the entire life. Owing to the grid model, all computations in our system are mere set operations. Assuming constant time for all set-operations of same sizes, our computation costs can be expressed as the number of set operations.

Figures 2, 3, 4 represent the lifetime, communication and computation costs of the two algorithms with each sensor having different energy and *T* equal to 80%, 85%, 90%, 95% and 100% respectively.

It is observed that LORD provides the same degree of coverage for comparable time duration. The communication costs of LORD is higher than MSC as the latter communicates only with the sensors that need to change state while the former requires all sensors changing state to communicate with all of their respective neighbors. However, more than 50% of LORD's cost is incurred only once, during the set-up phase (Figures 5), indicating that the overhead during the steady phase is minimal, thereby ammortizing the communication cost over the entire lifetime. This

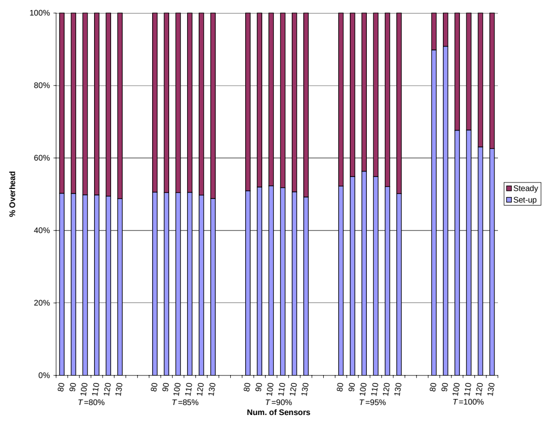

Figure 5. Communication overhead.

results in overall savings in system energy. The computation costs of MSC, however, exceeds that of LORD, since MSC tries to generate a cover for the entire area $\mathcal{A}$, from the list of all available sensors, while LORD generates a cover for only the dying sensor from the list of its OFF neighbors.

5. Conclusion

In this paper, we have presented a distributed, localized and reactive protocol for node-scheduling in a sensor network. The protocol has a two phase algorithm in which it first decides the sensors that need to be turned on and then it turns off any redundant active sensors, thereby minimizing the number of on sensors. We compared our algorithm to a highly efficient centralized algorithm and have shown that our results are comparably good. As future work, we intend to incorporate other system dependent parameters, like routing and communication, in our protocol.

References

[1] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor network topologies. In *INFOCOM*, 2002.

[2] K. Chakrabarty, S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Trans. on Computer*, 51(12):1448–1453, 2002.

[3] A. Chandrakasan, R. Amritaharajah, S. Cho, J. Goodman, G. Konduri, J. Kullik, W. Rabiner, and A. Wang. Design considerations for distributed microsensor systems. In *IEEE Custom Integrated Circuits Conference*, pages 279–286, 1999.

[4] 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 Conference on Mobile Computing and Networking(MobiCom)*, 2001.

[5] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. Saluja. Sensor deployment strategy for target detection. In *ACM Workshop on Wireless Sensor Networks and Applications*, pages 42–48, 2002.

[6] T. H. Cormen, R. L. Rivest, C. E. Leiserson, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[7] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalabale coordination in sensor networks. *Mobile Computing and Networking*, pages 263–270, 1999.

[8] H. Gupta, S. Das, and Q. Gu. Connected sensor cover: Self-organization of sensor networks for efficient query execution. In *Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, June 2003.

[9] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *INFOCOM*, 2001.

[10] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad-hoc sensor networks. In *Conference on Mobile Computing and Networking, Rome,Italy*, 2001.

[11] S. Meguerdichian and M. Potkonjak. Low power 0/1 coverage and scheduling techniques in sensor networks. Technical Report 030001, UCLA, 2003.

[12] R. Min, M. Bhardwaj, S. Cho, E. Shih, A. Sinha, A. Wang, and A. Chandrakasan. Low-power wireless sensor networks. In *Keynote Paper ESSCIRC, Florence, Italy*, 2002.

[13] M. Perillo and W. Heinzelman. Dapr: A protocol for wireless sensor networks utilizing an application-based routing cost. In *IEEE Wireless Communications and Networking Conference (WCNC)*, 2004.

[14] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. *ACM SIGMOBILE Conference on Mobile Computing and Networking*, 2001.

[15] D. Tian and N. Georganas. A node scheduling scheme for energy conservation in large wireless sensor networks. *Wireless Communications and Mobile Computing Journal*, 3(2):271–290, 2003.

[16] M. Tubaishat and S. Madria. Sensor networks: An overview. *IEEE Potentials*, 22(2):20–23, 2003.

[17] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Conference on Embedded Networked Sensor Systems*, pages 28–39, 2003.

[18] Y. Xu, J. Heidemann, and D. Estrin. Adaptive energy conserving routing for multihop ad hoc networks. Technical Report 527, USC/ISI, 2000.

[19] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *ACM/IEEE Conference on Mobile Computing and Networking*, 2001.

[20] F. Ye, G. Zhong, S. Lu, and L. Zhang. Energy efficient robust sensing coverage in large scale sensor networks. Technical report, UCLA.

[21] F. Ye, G. Zhong, S. Lu, and L. Zhang. Peas: A robust energy conserving protocol for long-lived sensor networks. In *Conference on Distributed Computing Systems (ICDCS)*, 2003.