Sacrificing a Little Coverage Can Substantially Increase Network Lifetime [1]

Limin Wang Sandeep S. Kulkarni
Software Engineering and Network Systems Laboratory
Department of Computer Science and Engineering
Michigan State University
East Lansing MI 48824 USA
Email: {wanglim1,sandeep}@cse.msu.edu

Abstract

We present a simple, local protocol, pCover, *which provides partial (but high) coverage in sensor networks. Through* pCover, *we demonstrate that it is feasible to maintain a high coverage (∼90%) while significantly increasing coverage duration when compared with protocols that provide full coverage. In particular, we show that we are able to maintain 94% coverage for a duration that is 2.3-7 times the duration for which existing protocols maintain full coverage. Through simulations, we show that our protocol provides load balancing, i.e., the desired level of coverage is maintained (almost) until the point where all sensors deplete their batteries.*

Keywords: sensor networks, node scheduling, partial coverage, lifetime, power conservation

I. Introduction

A surveillance sensor network needs to operate unattended for a long time, usually from several weeks to several months. However, sensor nodes are usually battery-powered, and, hence, can only operate continuously for a few days. Also, it is often very difficult to change batteries after deployment due to the large number and the embedded nature of sensor nodes. Therefore, energy conservation operations are critical for extending network lifetime. In this paper, we consider the approach where sensor nodes are over-deployed in a given area, only a subset of the nodes are in active mode to maintain a certain degree of sensing coverage (based on the desired system functionality), and the remaining ones are put in sleeping mode.

The work on sensing coverage can be broadly classified in terms of those that provide full coverage (single coverage or multiple coverage [1]–[3]) and those that provide partial coverage. In full coverage, every point in the network is covered by at least one sensor. While such coverage is desirable in sensitive environments such as military surveillance, it requires a large number of sensors to be awake. In partial coverage, by contrast, only a subset of points in the sensor network are covered and, hence, the number of sensors that need to be awake is reduced.

In this paper, we focus on *high partial coverage* (∼90% coverage), that we expect to combine benefits of full coverage (better surveillance) and partial coverage (longer lifetime). With high partial coverage, at any time, the intruder is likely to be detected with a high probability. Moreover, because the active nodes change over time, a stationary intruder will be detected with certainty in a short period. Finally, a moving intruder will also be detected within short distance. Such high partial coverage is desirable in many scenarios. For example, in a forest fire detection system, high partial coverage will ensure that most fires are detected immediately and all fires are detected within a short duration. Likewise, high partial coverage could also be used in applications such as ExScal [4], an application for intruder detection, classification and tracking.

We show that providing high (partial) coverage can significantly increase the lifetime of a given sensor network compared to the cases that provide full coverage. To illustrate this, we present a simple local protocol, *pCover*, that provides partial coverage. *pCover* is not dependent on global properties such as time synchronization. The degree of coverage is controllable locally. This is needed when network characteristics change. For example, sensing range changes due to environmental changes (grass, temperature,

[1] This work was partially sponsored by NSF CAREER CCR-0092724, DARPA Grant OSURS01-C-1901, ONR Grant N00014-01-1-0744, NSF equipment grant EIA-0130724, and a grant from Michigan State University.

rain, etc.) or reduced battery level. In such case, we should be able to dynamically tune the degree of coverage by simply sending a parameter to all the sensor nodes in the network. In a network with uniform deployment, the desired degree of coverage is maintained (almost) to the point where all sensor nodes fail. As discussed in Section VIII, along with a local algorithm, this feature can also be used to provide gradually decreasing coverage.

Our protocol provides a simple and practical solution for a partially covered sensor network. It can be used to provide full coverage as well. When used to provide guaranteed 100% coverage, *pCover* performs similarly as the state-of-art protocol that provides full coverage [5]. However, the main purpose of this paper is not to propose a protocol that provides partial coverage, but to use the proposed protocol as a tool to study the tradeoff between the degree of coverage and the lifetime of the network. Therefore, we put more emphasis on the analysis of lifetime-coverage tradeoff rather than presenting the protocol itself.

Contributions of the paper. We present a protocol, *pCover*, which maintains a certain degree of sensing coverage through sleep-awake scheduling of sensor nodes. Through simulation of *pCover*, we analyze the tradeoff between sensing coverage and network lifetime. We show that sacrificing a little sensing coverage can substantially increase the lifetime of the network, compared to protocols that provide full coverage (e.g., [5], [6]). In particular, in a random uniform deployment, our protocol maintains 94% coverage for a duration that is 2.3 times (respectively, 7 times) the duration for which the protocol in [5] (respectively, [6]) maintains full coverage. If the desired coverage is 91%, the duration that this coverage is maintained is increased to 2.6 times (respectively, 7.9 times) the duration that [5] (respectively, [6]) maintains full coverage. If the desired coverage is 84%, the duration that this coverage is maintained is 3.4 times (respectively, 10.3 times) the duration that [5] (respectively, [6]) maintains full coverage. And we show that *pCover* maintains load balancing, i.e., the degree of sensing coverage is maintained at desired level (almost) until the point where all sensor nodes deplete their batteries. Increasing or decreasing the desired degree of coverage shortens or prolongs the network lifetime accordingly.

Organization of the paper. In Section II, we identify the system model and assumptions of the partial coverage problem. In Section III, we present our partial coverage protocol, *pCover*. In Section IV, we analyze the effect of the configurable parameters in our protocol. In Section V, we show the tradeoff between coverage and network lifetime. In Section VI, we discuss several design issues including communication overhead and quality of surveillance. We survey related work in Section VII, and conclude in Section VIII.

II. System Model and Assumptions

We consider a network with stationary sensor nodes. Each node knows its location (through some localization service) and its sensing range. The sensing ranges of sensor nodes do not necessarily need to be the same. The different sensing ranges can be caused by the hardware (antenna, radio unit, etc.), the remaining power levels (which can change during nodes' lifetime), or the territory conditions.

We define the *neighbor set* of a node i as the set of nodes whose sensing areas intersect with the sensing area of node i. For simplicity, we assume that the sensing area of a node i is a circle with radius r_i centered at the location of the node. Hence, the neighbor set of node i is

$$N(i) = \{j \in R | d(i,j) \leq (r_i + r_j), j \neq i\}$$

where R is the entire set of sensor nodes in the target region, $d(i,j)$ is the distance between node i and node j.

A simple case is where all nodes have the same sensing range r. In this case, the neighbor set of a node i is

$$N(i) = \{j \in R | d(i,j) \leq 2r, j \neq i\}$$

We assume that the radio communication range is at least the distance to the farthest neighbor (i.e., no less than $2r$, in the case of equal sensing range). This assumption holds for most existing sensor products, such as [7]. Furthermore, some existing operating systems for sensor network, such as TinyOS, allow a sensor node to control the transmission power it uses in radio communication. In such a case, to reduce energy utilization and network congestion, a sensor node should use the lowest power level that reaches all its neighbors.

The partial coverage problem we address in this paper can be described as follows. The sensor nodes are deployed with redundant density. Sensor nodes self-organize to achieve a desired degree of coverage. This degree of coverage is maintained until some of the nodes have drained out power such that there are not enough nodes left to satisfy the coverage requirement.

III. *pCover*: Protocol Description

In this section, we present our partial coverage protocol, *pCover*. In *pCover*, a sensor node is either in working mode or sleeping mode. In Section III-A, we present the local rule that a sensor node follows to determine whether it should go to sleep or stay awake. In Section III-B, we describe state transition and how this local rule is used.

A. Sleep Eligibility Rule

We first define the terms we are going to use when we describe the local rule and the protocol.

Global coverage is the percentage of the target area that is covered by the working nodes. This is also called the *degree of coverage*.

Local coverage [2] of a node is the percentage of the node's sensing area that is covered by its awake neighbors.

Contribution of a node is the percentage of the node's sensing area that is *not* covered by its awake neighbors, i.e., $\mathit{Contribution} = 1 - \mathit{LocalCoverage}$.

The basic idea of the sleep eligibility rule is that: a sensor node should turn itself off if and only if its *local coverage* (respectively, *contribution*) is higher (respectively, lower) than a certain threshold, *SleepThreshold* (respectively, $1-$*SleepThreshold*).

To facilitate calculation, we imagine that the target area is covered by a virtual grid (Figure 1), where the size of a grid point is significantly smaller than the sensing area of a sensor node. In Figure 1, when A calculates its eligibility to sleep, it counts the number of grid points, say *TotalGrid*, that are within its sensing range. It also calculates the set of grid points, say *CoveredGrid*, that are covered by its *awake* neighbors. Thus, A can compute its local coverage ($CoveredGrid/TotalGrid$) and contribution, and decide if it should go to sleep or stay awake.

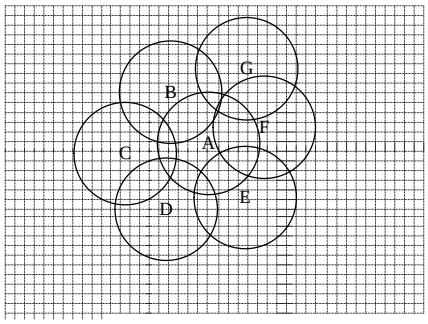


Fig. 1. Virtual grid

B. State Transition

In this section, we describe the state transition in *pCover* (Figure 2). Each node is in one of 4 states: *probing*, *awake*, *readyoff*, and *sleep*. Nodes switch among these states until they run out of power. Each node keeps a neighbor table, which maintains one record (ID, location, sensing range, status) for each neighbor.

[2]Note that the definition does not assume geometry of a node's sensing area. Hence, *pCover* can be applied to the case where sensing area is not a circle as well.

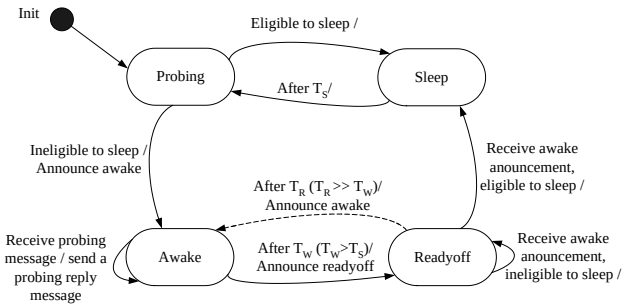


Fig. 2. State transition in *pCover*

A node in *probing* state probes the environment, evaluates the sleep eligibility rule, and decides if it should start working or go to sleep. A node in *awake* state is a working node. It monitors the area within its sensing range, and contributes to global coverage. A node in *readyoff* state is one that wants to sleep but cannot do that until its local coverage is higher than a certain threshold. It is also a working node, and contributes to global sensing coverage. It announces its intention to sleep so that its neighbors do not count it when they calculate their sleep eligibility, and, hence, are more likely to stay awake. A node in *sleep* state only keeps a timer on, and turns off all the sensing, communication, and computation devices, in order to save energy. Now we describe the actions a sensor node performs in these states.

Actions in Probing State. In the initial stage, all the nodes are in probing state. Each node waits for a random back-off time, and then broadcasts a probing message. The probing message contains the node's ID, location and sensing range. It is used to collect *awake* neighbors' information. Whenever a neighbor node receives this probing message, if it is in *awake* state, it sends back a probing reply message, which also contains its ID, location and sensing range. Nodes update their neighbor tables based on the neighbor information they have collected, compute the sleep eligibility rule, and decide if they should go to sleep or stay awake.

When a node in probing state computes the sleep eligibility rule, it applies a *SleepThreshold*, which we call **on-threshold**. A probing node starts working if it finds that its local coverage is lower than on-threshold.

The random back-off time is used for two purposes. First, the probing messages are sent at different time, so that they are unlikely to collide. Second, nodes should decide their states in sequential order. For example, consider node A and node B in Figure 1, if A and B try to decide their states at the same time, since neither of them is in awake state at the moment (both are in probing state), they are not counted for each other's local coverage. As a result, the redundancy increases.

If a node decides to stay awake, it broadcasts an "AnnounceAwake" message, and switches to *awake* state. On the other hand, if a node decides to go to sleep, it turns off its radio and sensing units, and goes to *sleep* state.

Actions in Awake State. A node in *awake* state actively monitors the area within its sensing range. It remains in *awake* state for T_w period of time (T_w could be chosen deterministically or randomly), then changes its state to *readyoff*, and broadcasts a "ReadyToOff" message. By sending a "ReadyToOff" message, the node indicates a desire to turn off.

Actions in Readyoff State. Intuitively, *readyoff* state is a reception only state, i.e., a node in readyoff state does not respond to any probing messages. Therefore, a readyoff node appears to its neighbors as a sleeping node. However, a readyoff node is in fact a working node, and provides sensing coverage. A node in readyoff state does not turn itself off until enough of its neighbors are awake to take over its sensing area. However, since it does not reply to probing messages, its neighbors do not count it when they compute their sleep eligibility. Also, a readyoff node watches for "AnnounceAwake" and "ReadyToOff" messages from its neighbors, and recomputes its sleep eligibility whenever it notices an increase in its local coverage. The *SleepThreshold* we apply here is **off-threshold**. If a readyoff node finds that its local coverage is higher than off-threshold, it will go to sleep state.

We also include an optional transition from readyoff state to awake state. A node that is in readyoff state for a long duration can switch to awake state and send an *AnnounceAwake* message. This transition allows one to deal with the case where a lot of nodes are in readyoff state although none of them can go to sleep state. Through simulations, we however found that a transition from readyoff state to awake state occurs only when there are no other alive nodes that can enable the node in readyoff state to sleep. In other words, in such a case, a node will continue to switch between awake state and readyoff state, thereby continually contributing to sensing coverage. However, in a real network, if the state of a node is corrupted for some reason, this transition will allow one to recover from the state where all nodes (large number of nodes) are erroneously in readyoff state.

Actions in Sleep State. A node in *sleep* state wakes up every T_s period of time. When it wakes up, it performs two tasks. First, it senses its surrounding area (within its sensing range) to see if there is any interesting event. This guarantees that any event that lasts longer than T_s will always be detected as long as there are alive sensor nodes in the area where the event occurs. Second, it changes its state to probing (and proceeds to execute actions in probing state).

We note that a node refreshes its neighbor table periodically. Therefore it provides a certain level of fault tolerance. We assume that when a node fails, it simply stops working and does not send or receive any messages. If a node that was in awake state fails, its neighbors will be able to detect it within T_s, at the time when they wake up, probe the neighborhood and get the updated neighbor information.

IV. Analysis of on-threshold and off-threshold

Since the goal of this work is to identify the tradeoff between full coverage and high partial coverage, we use the setting in [5], where the authors have proposed a protocol that is able to provide full coverage. In particular, the sensor nodes are deployed in a 160m$\times$160m field (called deployment area). The target area is the 140m$\times$140m square in the center of the field. Moreover, when we calculate the number of nodes, we only consider those in the central 100m$\times$100m area (called central area). Although our algorithm allows different sensing ranges, for simplicity, in these simulations, we set all the sensor nodes' sensing range r to be 10m.

In all the experiments, we set T_s to be 2 minutes, and T_w to be 10 minutes. We assume the lifetime of a sensor node is 1000 minutes, i.e., it is able to continuously work for 1000 minutes. We consider a network as "dead" when the global coverage of the network is less than a certain threshold even if all the alive nodes are working. We define the lifetime of a sensor network as the duration from the beginning of deployment until the network is dead. We use 50% as the threshold in our experiments. However, we found that the result is almost the same if a different threshold is used, as all the nodes ran out of power within a short period of time at the end of the simulation.

We analyze off-threshold and on-threshold in Sections IV-A and IV-B respectively. The simulations are conducted on a grid topology. For simplicity, we ignore communication cost. In Section VI-A, we show that this is reasonable in that communication cost is very small and affects network lifetime by less than 1%.

In our simulation, we also assume that there is no message collision or loss. (The same assumption was made in [5].) In *pCover*, we use a random back-off scheme to avoid message collisions. However, message collision is still possible. In case that the probing or probing reply messages collide, we can extend our protocol by allowing a sensor node to send a probing message twice. In the second probing message, the node includes the IDs of the neighbors it has already heard from, so that these neighbors will not send reply messages repeatedly.

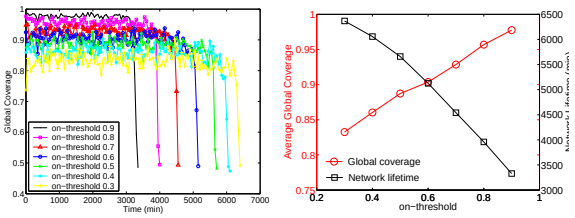
A. Varying Off-threshold

To identify the effect of off-threshold, we fix on-threshold at a given value, and vary off-threshold. From our experiments, we concluded that off-threshold does not have much impact on global coverage or network lifetime, as long as it is not set to too high (e.g., 0.9). In particular, if the off-threshold is within the range of 0.3-0.6, global coverage and lifetime of a given network are only decided by the on-threshold. Hence, for reason of space, we refer a reader to [8] for the detailed experiment results.

B. Varying On-threshold

Since off-threshold does not affect global coverage and network lifetime much, in this section, we fix off-threshold at 0.6, and study the effect of on-threshold on network lifetime and global coverage when it varies from 0.3 to 0.9. We set the distance between two neighbor nodes to be 7m, thus 196 sensor nodes (in a 14×14 grid) fall in the central 100m×100m area (node density is 1.96 nodes/r*r). The samples of global coverage are obtained periodically.

In Figure 3, we show global coverage and lifetime of the network. We find that global coverage (respectively, network lifetime) increases (respectively, decreases) linearly with the increment of on-threshold. This indicates that on-threshold can be used to locally control global coverage and network lifetime.



(a) (b)

Fig. 3. off-threshold is 0.6, and on-threshold varies from 0.3 to 0.9. Node density: 1.96 nodes/r*r. (a) global coverage over time (b) average global coverage vs. on-threshold and network lifetime vs. on-threshold.

We note that global coverage varies within a relatively short range (14%, from 84% to 98%), while network lifetime varies significantly (from 3300 minutes to 6400 minutes, almost doubles). From Figure 3 (a), we also found that under all on-thresholds, the global coverage is well maintained until one point, after which, the global coverage drops suddenly, and the network dies in a short period. This indicates that our algorithm maintains a balanced energy consumption as all sensor nodes run out of power at around the same time.

V. High Partial Coverage vs. Full Coverage

In Sections V-A and V-B, we study the lifetime-coverage tradeoff provided by *pCover* under two distribution patterns: deterministic distribution and random distribution. In the first case, sensors nodes are deployed in a grid topology. In the second case, nodes are deployed in a random manner. In Section V-C, we use these results to determine the tradeoff between full coverage and high partial coverage.

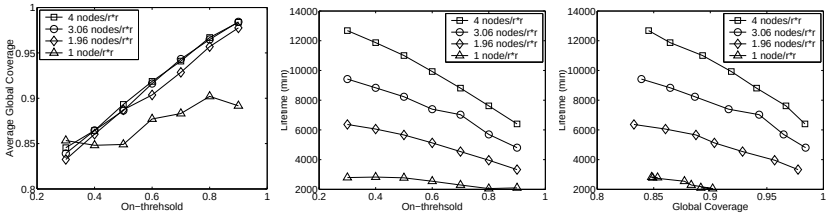
A. Networks with Deterministic Distribution

We experiment on different node densities. In Figure 4 (a), we show the relationship between global coverage and on-threshold. We can see that, global coverage linearly increases with the increment of on-threshold. Moreover, the curves that represent global coverage under different node densities are almost overlapping, with the exception of the node density being 1 node/r*r. In the case when the node density is as low as 1 node/r*r, the network is too sparse to meet the coverage requirement. It shows that global coverage is largely decided by on-threshold. Furthermore, when node density is reasonably high (e.g., not lower than 2 nodes/r*r), global coverage is independent of node density. In Figure 4 (b), we show that network lifetime decreases linearly when on-threshold increases. The space between the lines is even. Thus, if we fix the on-threshold, the lifetime of the network is proportional to the node density.

In Figure 4 (c), we show the tradeoff between global coverage and network lifetime, under different node densities. As we have already noticed from Figure 4 (a) and (b), when node density is as low as 1 node/r*r, the highest global coverage of a network that can be achieved is around 90%. When node density is higher, the limitation on the highest achievable global coverage no longer exists. Depending on the node density, decreasing the global coverage by 1% increases the network lifetime by 210 minutes (at 1.96 nodes/r*r) to 450 minutes (at 4 nodes/r*r).

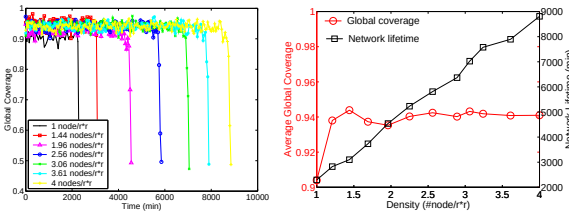
We now set on-threshold to 0.7, off-threshold to 0.6, and vary the density of the network from 1 node/r*r to 4 nodes/r*r. We show the change of global coverage over time for different node densities in Figure 5 (a). We find that global coverage is maintained at the same level (around 94%) under all densities, except the case when node density is 1 node/r*r (also shown in Figure 5 (b)).

In Figure 5 (b), we show that the global coverage is decided by the sleep thresholds (mainly on-threshold), independent of the density of the network. This is desirable, since it enables sensor nodes to control the global coverage locally by tuning the sleep threshold (on-threshold) without knowing the density of the network. The lifetime of



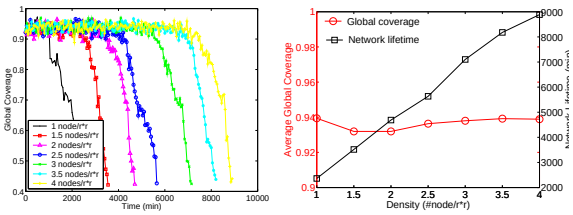
(a) (b) (c)

Fig. 4. Grid topology: off-threshold is 0.6. (a) average global coverage vs. on-threshold (b) network lifetime vs. on-threshold (c) network lifetime vs. global coverage.



(a) (b)

Fig. 5. Grid topology: on-threshold is 0.7, off-threshold is 0.6. (a) global coverage over time under different node densities (b) average global coverage vs. node density and network lifetime vs. node density.



(a) (b)

Fig. 7. Random topology: on-threshold is 0.7, off-threshold is 0.6. (a) global coverage over time under different node densities (b) average global coverage vs. node density and network lifetime vs. node density.

the network is linearly increasing with the increment of node density. This indicates that if we want to double the lifetime of the network without changing the global coverage, we could simply double node density in the target area.

B. Networks with Random Distribution

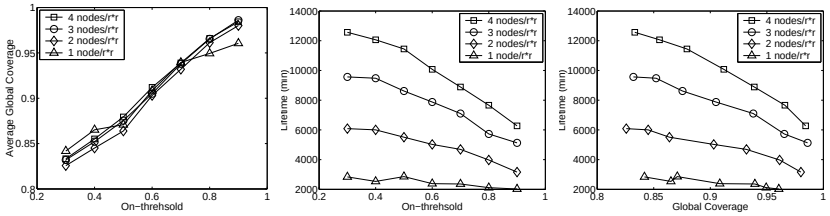
In this section, we study the lifetime-coverage tradeoff in a random topology. We repeat the same set of experiments as in Section V-A in a network with random topology. We fix the off-threshold to 0.6, and vary the on-threshold and node density. The results are shown in Figures 6 and 7.

Based on these results, the performance of *pCover* under random topology is similar to that where grid topology is used. Moreover, by comparing Figure 5 and Figure 7, we found that, the duration from the point where coverage starts dropping to the point where the network dies is slightly longer in random topology than in grid topology. This is caused by the unevenness of random deployment. In random topology, some areas are covered by more number of sensors, and some areas are covered by fewer number of sensors. The sensor nodes that are deployed in "sparse" areas stay in working mode for a longer period of time than those deployed in "dense" areas. Therefore, the energy consumption is not as well balanced as that in grid topology. However, in random topology, the energy consumption among sensor nodes is still reasonably balanced. For example, when node density is 3.5 nodes/r*r, global coverage is maintained at the desired level (around 94%) for about 7000 minutes. It then drops to less than 50% after another 1000 minutes.

C. Comparison of High Partial Coverage and Full Coverage

As we mentioned earlier, although *pCover* is designed to provide high partial coverage, it also provides guaranteed 100% coverage if we set both thresholds to 1. In this section, we compare the durations that *pCover* maintains desired degrees of partial coverage with the duration that it maintains full coverage. In addition, we also compare *pCover* with two other node scheduling protocols proposed in [6] and [5], which provide guaranteed full coverage. The simulations were conducted on networks with random topology. Node density varies from 2 nodes/r*r to 4 nodes/r*r.

In Figure 8, the left three bars in each density group represent the durations that the protocols proposed in [6], [5], and *pCover* maintain full coverage, respectively. The remaining four bars on the right represent the durations that *pCover* maintains different degrees of high partial coverage, from 94% to 84%, in decreasing order from left to right. It shows that by reducing the desired degree



(a) (b) (c)

Fig. 6. Random topology: off-threshold is 0.6. (a) average global coverage vs. on-threshold (b) network lifetime vs. on-threshold (c) network lifetime vs. global coverage.

of coverage slightly, we can substantially increase (more than double) the duration that the effective coverage is maintained. For example, when the node density is 4 nodes/r*r, the algorithm proposed in [5] is able to maintain full coverage for about 3000 minutes, the global coverage drops below 95% at around 4000 minutes, and drops below 90% at around 4500 minutes. The algorithm proposed in [6] is able to maintain full coverage for about 1000 minutes, and the global coverage drops below 90% at around 1500 minutes. *pCover* maintains full coverage for a similar amount of time as the algorithm in [5]. If the desired global coverage is 94%, *pCover* can maintain the effective coverage for 7000 minutes, 2.3 times (respectively, 7 times) the duration for which the protocol in [5] (respectively, [6]) maintains full coverage.

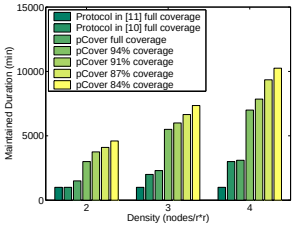


Fig. 8. Comparison of high partial coverage and full coverage.

Moreover, with the decrement of global coverage, the duration that the coverage is maintained strictly increases. For example, under the same density (4 nodes/r*r), if we reduce the coverage to 91%, the duration that the coverage is maintained at above 91% is extended to 7850 minutes; if we further reduce the coverage to 87%, the duration that the desired coverage is maintained is extended to 9350 minutes. This illustrates the tradeoff between coverage and network lifetime.

VI. Discussion

In this section, we discuss some design issues concerning communication overhead, quality of surveillance.

A. Communication Overhead

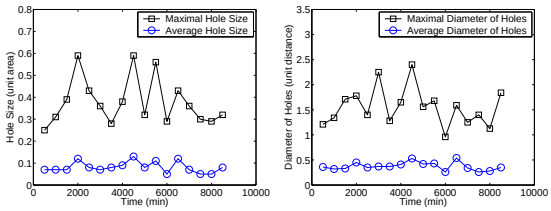
In Sections IV and V, we did not consider communication overhead, as the energy consumed in communication is much less than the energy consumed in idle listening. To illustrate this, consider the case where sensor nodes are deployed in a grid topology. Let the node density be 1.96 nodes/r*r, on-threshold be 0.7, and off-threshold be 0.6. For this setting, the simulation runs for 4541 minutes before the network dies. It turns out that the average number of transmissions per node is 10193, and the average number of receptions per node is 18202. Now we use the data from [9] to compute the communication cost on Mica motes. (Similar results can be found for other devices considered in [10], [11].) According to [9], the charge required by a Mica mote to transmit a packet is 20 nAh, to receive a packet is 8 nAh, and to idly listen for 1 millisecond is 1.25 nAh. Based on this data, the total communication cost (including transmissions and receptions) in the above simulation is equivalent to the cost that a node stays in idle mode for 4.66 minutes, which is only 0.1% of the network lifetime (4541 minutes).

B. Quality of Surveillance

Degree of coverage, i.e., the percentage of target area that is covered by working sensor nodes, provides one metric to measure the level of network surveillance. In a partially covered sensor network, the quality of surveillance is often measured in terms of how fast the sensors can detect a target object. The target object can be either stationary or moving. Because the sensor nodes wake up every 2 minutes to check the changes in the environment and neighbors' status, the stationary objects can always be detected within 2 minutes. Moreover, since sensor nodes rotate their working load so that different sets of active nodes are selected at different time. The coverage "holes", the areas that are uncovered by any sensor node, change locations continuously. Hence, an uncovered object in a "hole" is likely to be detected at the time nodes change working schedule.

In the case of moving objects, the quality of surveillance is affected by the sizes (or diameters) of the "holes", i.e., the areas that are not covered by any sensor node. We measure the sizes and the diameters of the holes. For comparison, we define the a node's sensing area (πr^2) as a *unit area*, and the diameter of a node's sensing area ($2r$) as a *unit distance*. Since we have set the nodes' sensing range r to 10m, the unit area is $314m^2$, and the unit distance is 20m.

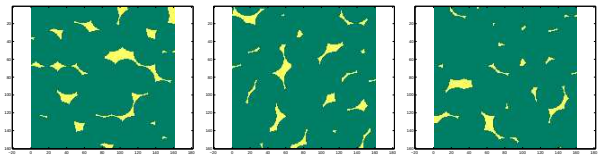
We analyze the quality of surveillance on the case where nodes are deployed in a grid topology, density is 4 nodes/r*r, on-threshold and off-threshold are set to 0.7 and 0.6, respectively. The global coverage is maintained at 94%, and the lifetime of the network is 8824 minutes. We obtain nodes' on/off status periodically. In Figure 9, we show the sizes and diameters of the coverage holes over time. In Figure 10, we show the snapshots of the network at 2500 minutes, 5000 minutes, and 7500 minutes. As shown in Figure 9 (a), the sizes of the holes are small. The maximal size of the holes is close to 0.6 unit area, and the average size of the holes is only 0.05-0.13 unit area. This is reasonable considering the thresholds we are using are 0.7 (on-threshold) and 0.6 (off-threshold), which means a sensor node starts working if it finds that an uncovered area within its sensing range is larger than 0.3 unit area, and it does not turn off if the uncovered area within its sensing range is larger than 0.4 unit area. The holes occasionally become larger when the uncovered areas that are adjacent to each other are connected. However, this happens infrequently.



(a) (b)

Fig. 9. on-threshold is 0.7, off-threshold 0.6. Node density: 4 nodes/r*r. (a) maximal and average size of the holes (b) maximal and average diameter of the holes.

In Figure 9 (b), we can see that the diameters of the holes are also short. The average diameter of the holes is only 0.26-0.54 unit distance. The maximal diameter of the holes is 2.4 unit distance. This means that if the target object moves fast (before nodes' rescheduling), the maximal distance it can travel before being detected is 2.4 times the diameter of a sensor's sensing area. In a typical surveillance sensor network such as intrusion detection system, the intruder does not know the locations of the holes as they changes continuously. It is unlikely that the intruder can choose to move along the longest uncovered path. Hence, the detection latency is short in average.



(a) (b) (c)

Fig. 10. on-threshold is 0.7, off-threshold 0.6. Node density: 4 nodes/r*r. (a) 2500min (b)5000min (c) 7500min.

VII. Related Work

In recent years, several node scheduling approaches [5], [6], [12]–[14] have been proposed for surveillance sensor networks to minimize energy consumption while maintaining sensing coverage at the desired level. These approaches try to maintain a subset of sensor nodes in working mode and put the remaining nodes to sleep. In [6], a node scheduling scheme is proposed to preserve full sensing coverage by using an off-duty eligibility rule. Based on the rule, a sensor node is allowed to turn off if and only if its neighbor nodes are able to completely cover its sensing area. When a node computes its neighbors' contribution to its sensing coverage, it simplifies the calculation by using "sponsored sectors". Although it guarantees 100% coverage, it underestimates the area neighbor nodes can cover, and, hence, leads to redundancy and energy waste, as pointed out in [5]. Moreover, to balance energy consumption, the node scheduling operation is divided into rounds, and nodes perform rescheduling at the beginning of each round. This per-round based operation requires global synchronization service.

In [5], each node decides its sleep-awake schedule at the initialization phase. The set of working nodes are able to provide full coverage of the target area at any time. Moreover, a differentiated sensing coverage can be achieved by proportionally extending (k-coverage, $k > 1$) or shrinking (partial coverage) the work periods. Although this provides a solution on partial coverage as well, the study in [5] was only conducted on 1-coverage or k-coverage (where $k > 1$). Also like [6], the algorithm proposed in [5] requires time synchronization.

We have compared the performance of the algorithms proposed in [5] and [6] with *pCover* in Section V-C. We showed that *pCover* performs similarly as the algorithm in [5] when used to provide full coverage and is able to provide substantially longer network lifetime when used to provide partial (but high) coverage.

Also, the coverage duration in our protocol is close to the theoretical estimate identified in [15]. In particular, if the coverage is 95% and the node density is 4 nodes/r*r,

the *theoretical upper bound* for coverage duration is 9.8 times the lifetime of a single sensor node. In comparison, our protocol maintains 94% coverage for a duration that is 8.9 times the lifetime of a single sensor node.

In [14], Wang *et.al.* prove that an area is k-covered if all the intersection points in the same area are k-covered, where $k \geq 1$. Based on this, they propose the Coverage Configuration Protocol (CCP), in which a sensor node is eligible to turn off if all the intersection points inside its sensing circle are at least k-covered. CCP is designed to provide full coverage or higher degrees of coverage. According to the simulation results from [14], the number of active nodes required by CCP to provide 1-coverage in an r*r square area is about 0.88-0.92, which is similar to that required by [5], and is more than twice the number of active nodes required by our protocol, in order to provide 94% coverage. Therefore, our protocol can provide significantly longer network lifetime (by reducing a little coverage) than CCP.

pCover's probing mechanism is inspired by PEAS [12], which is a density control protocol. In PEAS, a sleeping node wakes up periodically and broadcasts a probing message to its neighbors within a certain range. Any node that is in working mode responds to the probing message by transmitting a reply message. If a probing node receives a reply message before a timeout, it goes back to sleep. Otherwise, it starts working. In PEAS, a working node keeps awake continuously until its physical failure or depletion of power. This is undesirable since it causes unbalanced energy consumption. To address this problem, in [13], the authors propose PECAS as an extension to PEAS. PECAS lets a node go to sleep after it has been working for a given period of time. When a working node sends a probing reply message, it includes the remaining time it will continue working before going to sleep. Thus, the neighbor nodes will wake up in time to take over the sensing area. PEAS and PECAS control the density of working nodes (hence, the degree of coverage) by controlling the probing range. However, the estimate is inaccurate, and, thus, they are unable to provide guarantees on degree of coverage. By contrast, *pCover* can provide guaranteed full coverage (by setting both thresholds to 1) and the desired level of partial coverage.

Other node scheduling protocols include Randomized Independent Sleeping (RIS) ([1]) and MESH ([13]). Since these algorithms do not involve neighbor cooperation, they are not able to self-configure when the environment changes (e.g., node failure, sensing range changes). In a related area, several node scheduling protocols have been proposed for topology control, such as LEACH [16], GAF [17] and SPAN [18] for ad hoc networks, and ASCENT [19] for sensor networks. Although they perform in a similar way as to put the redundant nodes to sleep in order to save energy, these topology control protocols only consider communication connectivity, and try to establish a routing backbone. By contrast, the coverage problem addresses the issue of selecting active sensor nodes to cover every physical point in the target area.

The quality of surveillance in a partially covered sensor field has been studied in [13], [20]–[23]. In [13], Gui and Mohapatra define the quality of surveillance metric using the expected traveled distance of a target before it is first detected by any sensor. They propose two node scheduling protocols, PECAS and MESH (as we mentioned earlier), and then evaluate the performance of various sleep planning protocols using the quality of surveillance metric. Unlike the work in [13], we study the tradeoff between full coverage and high partial coverage, and show the increment in network lifetime by reducing a little coverage. And, our protocol can maintain a network lifetime that is close to (about 90%) of the *theoretical upper bound* [15]. In [20], Ren *et.al.* present a mathematical model to analyze the object detection quality under probabilistic coverage. This work is complementary to our work in that the main focus in [13], [20]–[23] is to define the *quality* of partial coverage.

VIII. Conclusion

Maintaining a high degree of sensing coverage and prolonging the system lifetime are two conflicting goals for sensor networks. In this paper, we focused on the effect of providing a partial (but high) coverage in sensor networks on network lifetime. We demonstrated that it is possible to obtain a high level of sensing coverage ($\sim$90%) while substantially increasing lifetime when compared with protocols that provide a full (100%) sensing coverage. To illustrate this, we presented a protocol, *pCover*, that depends only on local information and does not depend on services such as time synchronization.

We simulated *pCover* on networks of grid topology and random topology. The simulation results show that under random topology, our protocol maintains 94% global coverage for a duration that is 2.3-7 times the duration for which existing protocols ([5], [6]) maintain full coverage. Under grid topology, our protocol maintains the same level of coverage for a duration that is 21.4% longer than that under random topology. Also, as discussed in Section VII, our protocol achieves a lifetime that is about 90% of the *theoretical upper bound* on lifetime identified in [15]. Moreover, we show that our protocol provides load balancing. In a network with uniform distribution, all the sensor nodes die within a short period. The desired degree of global coverage is maintained (almost) until the point where all sensor nodes die.

Based on the simulation results, we found that the

relationship between global coverage and on-threshold is fixed, and is independent of node density (as long as the density is high enough to provide the desired coverage). Thus, we can locally control the global coverage by simply tuning the local parameter on-threshold. For a given on-threshold (or global coverage), we can predict the lifetime of the network under a certain node density, or compute the required node density in order to achieve a desired lifetime. We show that network lifetime is proportional to node density, under a certain on-threshold (or global coverage). Our protocol is also useful in dynamic reconfiguration. To illustrate this, consider the scenario when the sensor nodes have been deployed, we find that the network is required to last longer than originally planned. In this case, we can reduce the value of on-threshold in order to reduce the degree of coverage and extend network lifetime. The new value of on-threshold can be distributed to all the sensor nodes in the network through a reprogramming service (e.g., [24]).

Also, in *pCover*, the coverage is maintained (almost) until the point where energy of all sensors is depleted. *pCover* can be easily modified to achieve a gradually decreasing coverage. To achieve this, we need to reduce the on-threshold when the battery level of nodes drops below a certain point. (Alternatively, nodes can change their on-threshold when a certain amount of time has passed since deployment.) Since the nodes' battery is consumed at approximately the same rate, this would cause all nodes to change their on-threshold approximately at the same time. This would gradually reduce the level of coverage and result in increased lifetime.

References

[1] S. Kumar, T. H. Lai, and J. Balogh. On k-coverage in a mostly sleeping sensor network. *In Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*, pages 144–158, October 2004.

[2] S. Kumar, T. H. Lai, and A. Arora. Barrier coverage with wireless sensors. *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2005.

[3] C. Huang and Y. Tseng. The coverage problem in a wireless sensor network. *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 115–121, 2003.

[4] A. Arora et. al. Exscal: Elements of an extreme scale wireless sensor network. *The International Conference on Real-Time and Embedded Computing System and Applications (RTCSA)*, August 2005.

[5] T. Yan, T. He, and J A. Stankovic. Differentiated surveillance for sensor networks. *In Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 51–62, November 2003.

[6] D. Tian and N. D. Georganas. A node scheduling schedule for energy conservation in large wireless sensor networks. *Wireless Communications and Mobile Computing Journal*, May 2003.

[7] Crossbow Technology, Inc. *MSP410, TelosB, MICA2 Datasheets*. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/{MSP410,Tel%osB,MICA2}_Datasheet.pdf`.

[8] L. Wang and S. Kulkarni. pCover: Partial coverage for long-lived surveillance sensor networks. Technical report, Department of Computer Science and Engineering, Michigan State University, November 2005.

[9] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, September 2002.

[10] V. Shnayder, M. Hempstead, B. Chen, G. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. *In Proceedings of ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2004.

[11] M. Stemm and R. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Transactions on Communications*, E80-B(8):1125–1131, August 1997.

[12] F. Ye, G. Zhong, J. Cheng, S. W. Lu, and L. X. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. *In Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, May 2003.

[13] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. *In Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*, October 2004.

[14] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. *In Proceedings of ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[15] H. Zhang and J. Hou. On deriving the upper bound of α-lifetime for large sensor networks. *In Proceedings of the Fifth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2004.

[16] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. *In Proceedings of the 33rd Hawaii International Conference on System Sciences*, Janauary 2000.

[17] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. *In Proceedings of the 6th ACM MOBICOM Conference*, July 2001.

[18] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *In Proceedings of the 6th ACM MOBICOM Conference*, July 2001.

[19] A. Cerpa and D. Estrin. ASCENT: Adaptive self-configuring sensor networks topologies. *IEEE Transactions on Mobile Computing*, 3(3):272–285, 2004.

[20] S. Ren, Q. Li, H. Wang, X. Chen, and X. Zhang. Analyzing object detection quality under probabilistic coverage in sensor networks. *In Proceedings of the 13th International Workshop on Quality of Service (IWQoS)*, June 2005.

[21] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad hoc sensor networks. *In Proceedings of 7th ACM International Conference on Mobile Computing and Networking (Mobicom)*, pages 139–150, July 2001.

[22] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. *In Proceedings of IEEE INFOCOM*, 3:1380–1387, April 2001.

[23] G. Veltri, Q. Huang, G. Qu, and M. Potkonjak. Minimal and maximal exposure path algorithms for wireless embedded sensor networks. *In Proceedings of ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[24] S. S. Kulkarni and L. Wang. MNP: Multihop network reprogramming service for sensor networks. *In Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS)*, pages 7–16, June 2005.