

# A Framework for Resilient Online Coverage in Sensor Networks

Ossama Younis, Marwan Krunz, and Srinivasan Ramasubramanian

Department of Electrical & Computer Engineering

University of Arizona, Tucson, AZ 85721

E-mail: {younis,krunz,srini}@ece.arizona.edu

**Abstract**—We consider surveillance applications in which sensors are deployed in large numbers to improve coverage fidelity. Previous research has studied how to select active sensor covers (subsets of nodes that cover the field) to efficiently exploit the redundant node deployment. Little attention was given to studying the tradeoff between fault tolerance and energy efficiency in sensor coverage. In this work, our objective is to rapidly restore coverage of the field under unexpected node failures. For this purpose, we explore different adaptable coverage strategies. We then propose design guidelines for applications employing distributed cover-selection algorithms to control the degree of redundancy at local regions in the field and achieve energy-efficient coverage. In addition, we develop a new distributed technique to facilitate switching between active covers without the need for node synchronization. Distributed cover selection protocols can be integrated into our framework (referred to as “resilient online coverage” (ROC)). We evaluate the effectiveness of ROC through analysis and extensive simulations.

## I. INTRODUCTION

In surveillance applications, sensors are deployed in large numbers to improve field coverage and prolong network lifetime. Network lifetime is prolonged by putting nodes to sleep as much as possible to save their batteries. For this purpose, two approaches were proposed. The first is a “MAC layer” approach, in which a node uses a duty cycle to schedule its sleep and wake-up periods based on the traffic pattern (e.g., [1], [2]). The other approach is an “application layer” approach, in which a subset of sensors (referred to as a “cover”) actively monitors the field and the rest of the nodes are put to sleep (e.g., [3], [4], [5], [6]). This approach can balance the load among all sensors by periodically selecting new covers. It can also control the quality of surveillance. We focus on application-layer coverage in this work.

This work was supported by the National Science Foundation under grants CNS-0627118, CNS-0313234, 0325979, and 0435490.

We study monitoring applications that require every point (target) in the field be covered by at least one sensor, and that a “hole” (unmonitored region) caused by a failing node be healed within a specified time interval, referred to as the “tolerance interval” ( $M$ ).  $M$  represents the duration of time in which detecting an event or a phenomenon is still possible (e.g., chemical or radiation activity). We consider scenarios in which any node may fail due to environmental factors, e.g., lava coming from a volcano, or adversarial issues, e.g., bombing in a military zone. For energy conservation, the application employs a distributed cover selection protocol (see Section II). Two issues need to be addressed in this setting: (1) how to rapidly cover holes within the tolerance interval at a reduced energy cost, and (2) how to adapt to non-uniform failure patterns.

Several design alternatives can be used for failure recovery. The most reliable approach is to perform cover refreshment *more frequently* to allow new active nodes replace the failing ones. This approach has significant overhead since cover refreshment is a network-wide process. In addition, it is difficult to compute the required duration between successive cover refreshments if the node failure probability is arbitrary. Another approach for failure recovery is to select a “ $k$ -cover,” which is one that ensures that every point (target) in the field is covered by at least  $k$  sensors, where  $k > 1$ . Despite its simplicity, this approach has a couple of drawbacks. First, a  $k$ -cover that is active for a long duration of time cannot adapt to different failure patterns, especially if failures tend to be clustered around specific regions in the field. Second, a  $k$ -cover generally requires activating  $O(kN_c)$  sensors, where  $N_c$  is the number of sensors in a 1-cover (typically, a  $k$ -cover has less number of nodes than  $k$  node-disjoint covers). This approach is obviously not energy efficient.

If every sensor in a 1-cover monitors the region at the rate of one observation per time unit, then a  $k$ -

cover may be employed where each sensor monitors the region at the rate of 1 observation per  $k$  units. Such frequency relaxation maintains the average number of observations made per point in the cover over a large time-scale. Every point is guaranteed to be observed at least  $2k$  times in an interval of  $2k$  units, while it is not guaranteed that a point will be observed by at least one sensor in every  $k$  units, as achieved by a 1-cover, unless the  $k$ -cover may be decomposed into a set of  $k$  node-disjoint covers. Thus, the quality of coverage is reduced when a  $k$ -cover with reduced frequency of observation is employed. In addition, the failure rates of components are typically higher while in active mode than in sleep mode. Therefore, activating a large number of nodes increases the failure rate of the system.

A desirable approach is to select a minimal set of sensors to cover the region and an additional backup set which is periodically woken up to check on the active set. In this approach, only a small number of nodes are active in the network, while each active node is protected by a small collection of backup nodes. It is also desirable to have adaptable coverage when the failure rate varies at different regions in the field, as depicted in Fig. 1. Providing an adaptable coverage behavior that exploits the tradeoff between fault tolerance and energy efficiency has not been given enough attention in the literature.

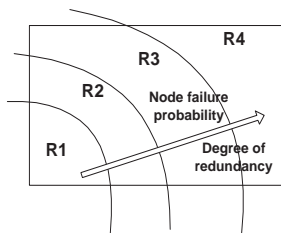


Fig. 1. Four regions in the field with different failure rates.

In addition to fault tolerance, cover-selection protocols have not adequately tackled how to switch between active covers in a distributed environment. To have all the nodes participate in cover selection, synchronized wake-up is necessary. Node synchronization is practically difficult in networks with sleeping nodes. Clock drifts may result in some nodes being always early and participate in the active cover until they die. This may have a detrimental effect on network lifetime, as explained in the example in Fig. 2. In this example, five nodes with different sensing ranges are deployed in the field. Assume that each node has a maximum lifetime of 1 time unit while active. The nodes form three 1-covers:  $\{A,B\}$ ,  $\{A,C,D\}$ , and  $\{B,C,E\}$ . If the

covers are switched every 0.5 units, the network lifetime is 1.5 time units (which is the optimal lifetime for this configuration). However, if the switching process selects the same cover until this cover dies, the network can only live for 1 time unit.

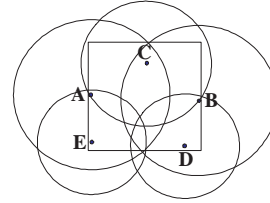


Fig. 2. Five sensors covering a square region. The circles denote the sensing region of each sensor.

**Contributions.** In this work, we propose design guidelines that leverage cover-selection algorithms to dynamically maintain 1-coverage under unexpected node failures. Our approach lets every active node independently select a set of backup covers for its sensing region, and schedule them to periodically check for holes. This results in a “node-defined” control of the degree of redundancy and surveillance, which is more adaptable than global, “network-defined” recovery. Adaptability stems from two points. First, every node can control the degree of redundancy in its region to satisfy regional surveillance requirements. Second, every node can autonomously adjust the schedule of its backups to maintain a certain speed of recovery from failures in its region. Furthermore, we develop an intelligent technique (OCU) for switching between sensor covers under realistic assumptions, such as node asynchrony. OCU performs the negotiations required for selecting a new cover during the operation of the current active cover. Our integrated approach for online, fault-tolerant surveillance and switching between covers is referred to as *resilient online coverage* (ROC).

Note that this work is **not** about designing new protocols for cover selection, nor is it our interest to study the performance of existing cover-selection protocols. Our main goal is to exploit the fault-tolerance/energy-efficiency tradeoff and provide adaptable control of redundancy and surveillance.

**Organization.** The rest of this paper is organized as follows. Section II briefly surveys related work. Section III states our assumptions and formulates the problem. The ROC framework is introduced in Section IV and is analyzed in Section V. Section VI describes the OCU technique for switching between active covers. Section VII evaluates the performance of ROC. Finally,

Section VIII provides concluding remarks.

## II. RELATED WORK

Fault-tolerance in wireless sensor networks has seen attention lately. In [7], the authors proposed algorithms for using heterogeneous types of sensors as backup for each other to enable efficient multimodal data fusion. In PEAS [8], probabilistic sleep/wake up was proposed to maintain network connectivity under unexpected failures. Selecting a  $k$ -cover was typically proposed to avoid having holes in the monitored region [9], [10]. Previous research has not studied the tradeoff between fault tolerance and energy efficiency in field coverage.

Research on determining an active set of nodes in a sensor network can be classified into two categories. We refer to the first category as the *deterministic cover selection (DCS)* approach, which comprises the majority of research in this domain. It depends on electing a set of nodes that guarantees coverage of at least  $\alpha\%$  of the field, where  $\alpha \leq 100$  [11], [12], [3], [4], [5], [6], [13], [9]. The DCS approach has the advantage of guaranteeing the maximum coverage that the network can offer. It is also not sensitive to how nodes are distributed in the field, and can achieve minimal covers (in terms of the necessity of every active node). Its primary drawback is that unexpected node failures may reduce the quality of coverage until a new cover is selected. We refer to the second category of schemes for selecting active covers as the *stochastic cover selection (SCS)* approach. This approach includes RIS [9] and PECAS [14]. In these protocols, nodes decide whether or not to sleep and for how long based on certain probability distributions. The SCS approach requires little node coordination and only coarse node synchronization. However, it only provides asymptotic guarantees on field surveillance under certain node distributions, which might not be reliable in practice. The active cover at any instant in time can also be far from minimal.

We focus on the DCS approach because of its suitability for critical surveillance applications. Our work exploits the benefits of having a fixed working cover and schedules wake-up for selected backup nodes to achieve fault tolerance, while reducing the required energy cost.

## III. PROBLEM STATEMENT

**Assumptions.** We consider applications that require continuous field surveillance in a hostile environment (e.g., to detect chemical activity in a military field). We assume the following about the network:

- The application deploys nodes such that every point is at least  $k$ -covered (see [9]).
- At every sensor, the time is slotted and the slot size =  $t_s$  units. The granularity of  $t_s$  represents the duration between two successive sensor reports.
- The network application employs a DCS algorithm for cover selection. The cover operates for  $T$  time slots. Undetected events due to unmonitored areas (holes) can be tolerated up to an interval of at most  $M$  slots, where  $1 \leq M \leq T$ .  $M$  determines the required speed of recovery (typically  $M \ll T$ ).
- Issues such as knowledge of node locations or connectivity constraints are dependent on the employed DCS algorithm, and are not requirements in our framework. For example, the DCS protocol in [15] selects covers without relying on node locations, unlike the ones in [3], [5]. Detecting a hole in the active cover is done using the same approach that the DCS algorithm uses to find uncovered regions.
- Nodes are stationary, can be added anytime during network operation, and may not be synchronized.
- Failures may be random across the field or may be clustered in certain regions.

**Objectives.** We aim at leveraging the DCS algorithms to provide:

- 1) Energy-efficient, fault-tolerant operation under unexpected failures. More specifically, it is required to design a framework for maintaining 1-coverage of every point in the field under the above model. The framework should allow for controlling the degree of redundancy at different regions in the field and should minimize energy consumption.
- 2) Efficient, low-overhead switching between active covers without relying on node synchronization.

We assume that connectivity is guaranteed by the conditions in [4], [5]. However, our framework is general and can incorporate techniques that compute connected covers under no range constraints (e.g., [6], [13]).

## IV. RESILIENT ONLINE COVERAGE (ROC)

In this section, we present ROC and analyze its impact on the quality of field coverage.

### A. Design Rationale and Overview

ROC requires selecting an active 1-cover ( $V_A$ ) using any DCS algorithm. The remaining nodes are assigned to the *sleeping set* ( $V_S$ ). Initially, all nodes participate in selecting  $V_A$ . Each node keeps a list of its neighbors, and periodically announces this list to its 1-hop neighbors. Through periodic neighbor list announcements from

nodes in  $V_A$ , a newly deployed node is made aware of the existence of sleeping neighbors.

To achieve an adaptable coverage model as shown in Fig. 1, we adopt a novel recovery mechanism in which every active node selects a subset of its neighbors assigned to  $V_S$  (referred to as a “backup cover”) and schedules them to periodically wake up. The backup sets are proactively selected after a DCS algorithm constructs a 1-cover. Our backup selection mechanism provides opportunity for “node-defined” instead of a “network-defined” coverage recovery. Energy is also conserved by intelligently scheduling the activation of the nodes in the backup sets. Note that in our network model, it is not possible to adopt a reactive recovery mechanism after a hole occurs. This is because we do not assume that nodes carry any special hardware to allow activation while being asleep [3]. Backup nodes periodically wake up and go to the “PROBING” state. When a backup node discovers an unmonitored region, it joins  $V_A$ . Non-backup nodes in  $V_S$  have no role during the operation of  $V_A$  and can remain asleep for the entire  $T$  slots. A state diagram of the node states is specified in Fig. 3. Details of the recovery mechanism in ROC are given below.

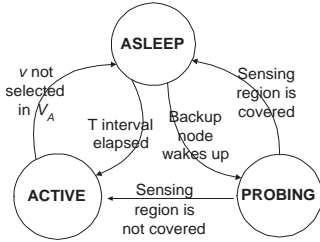


Fig. 3. State diagram for a node  $v$  employing ROC.

### B. Selecting Backup Covers

We define a *backup cover* of a node  $v$  as a set of  $v$ 's neighbors that covers the entire sensing range of  $v$ . If  $v$  decides to be part of  $V_A$ , it computes a maximum of  $S$  backup cover sets ( $S \geq k - 1$ ) that are as node-disjoint as possible. An example backup cover is shown in Fig. 4(a) for  $S = 1$ . Note that having  $k$ -coverage in the field does not necessarily mean that  $S$  independent backup covers can be found (we will show the relation between constructing  $S$  backup covers and  $k$ -coverage in Section V). If  $v$  can not find  $S$  backup covers, it computes the maximum possible number of covers it can find. If the node density drops during network operation, the  $S$  sets may only be partial covers, as in Fig. 4(b).

Node  $v$  does not need to enumerate all its possible backup covers since the covers need not be minimal

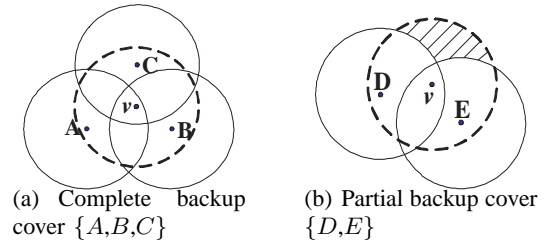


Fig. 4. Two backup covers of node  $v$ .

in terms of size. This is because not all of the nodes in a backup cover will need to remain awake to cover a resulting hole. Therefore,  $v$  employs the following greedy algorithm which has the sole objective of computing covers that are as node-disjoint as possible. Assume that  $N(v)$  is the set of neighbors of  $v$ . Node  $v$  first sorts  $N(v)$  according to any optimization parameter (e.g., remaining battery or proximity), and selects the top-listed neighbor to be in backup cover  $s_1$ . Then,  $v$  sequentially adds neighbors to  $s_1$  to cover its sensing region, similar to how the DCS algorithm selects a 1-cover in the network. When the nodes in  $s_1$  form a cover of  $v$  (or adding more nodes does not increase the covered area),  $v$  starts computing the next set  $s_2$ . The first node  $u$  added to any backup cover  $s_i$  ( $i > 1$ ) should satisfy that  $u \in N(v) - (\cup_{j < i} s_j)$ . Neighbors are then added to  $s_i$ , giving preference to those having the least occurrences in the already computed backup covers. An example is shown in Table I for selecting  $S = 3$  backup covers of a node  $v$  shown in Fig. 5. The numbers under each neighbor denote the frequency of occurrence of that neighbor in the  $S$  covers. Node  $H$  was not selected in  $s_3$  because it became redundant after  $G$  was added.

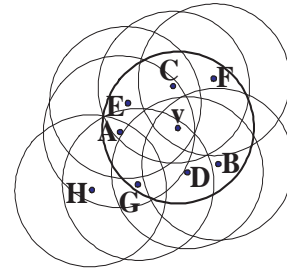


Fig. 5. A configuration containing node  $v$  and its neighbors.

### C. Scheduling Backup Covers

Nodes in the backup covers of  $v$  are notified of their roles and their sleep/wake up schedule. Backup nodes can be assigned either a *conservative* or an *opportunistic* sleep schedule. In the conservative schedule, all the

TABLE I  
SELECTING THREE BACKUP COVERS FOR A NODE  $v$  FROM A LIST  
OF EIGHT NEIGHBORS.

Step	A	B	C	D	E	F	G	H	Cover set
0	0	0	0	0	0	0	0	0	
1	1	1	1	0	0	0	0	0	$s_1=\{A,B,C\}$
2	1	1	1	1	1	1	0	0	$s_2=\{D,E,F\}$
3	1	2	1	1	2	2	1	0	$s_3=\{G,B,E,F\}$

backup nodes should wake up periodically every  $M'$  slots, where  $M' \leq M$ . Waking up all the  $S$  covers might not be necessary, however, since a node's probability to fail within  $M$  slots is typically small. In addition, a backup node  $v_1 \in V_A$  can heal the hole caused by a failing node  $v_2 \in V_A$  whose backups are still asleep. Thus, we propose the following opportunistic approach.

Let  $s_i$  be the  $i^{\text{th}}$  backup cover of  $v$ ,  $1 \leq i \leq S$ . The opportunistic approach activates the nodes in  $s_1$  within  $M$  slots, the nodes in  $s_2$  within  $2M$  slots, and so on (analysis of this approach is given below). More specifically, let the nodes in  $s_i$  be indexed according to their order in  $s_i$ . Node  $v$  schedules a neighbor  $v_j \in s_i$  to sleep for a number of slots equal to  $\max(iM - n_{ci} + j, 0)$ , where  $n_{ci}$  is the number of nodes in  $s_i$ . After that,  $v_j$  sleeps for  $\min(iM, T')$  cycles, where  $T'$  is the remaining number of time slots for the current  $V_A$  (according to  $v_j$ 's clock). A sleeping node  $u$  might be a backup for several neighbors in  $V_A$ , each of which assigning  $u$  a different sleep interval. Node  $u$  selects the smallest sleep interval assigned by any neighbor in  $V_A$ . The opportunistic scheduling approach is best used when the node failure probability can be estimated and  $M$  is small compared to  $T$  (see our results in Section VII).

#### D. Backup Node Probing

When a backup node  $u$  wakes up, it probes the active nodes in its neighborhood. Node  $u$  can perform active probing by sending a probing message and waiting for replies, or passive probing by listening to periodic updates from neighbors. Passive probing is advantageous in that it does not require message exchange. However, it would typically require a node to remain awake for the entire slot duration ( $t_s$ ) to ensure that reports are sent by active neighbors. If  $v$  discovers that its *current* neighbors in  $V_A$  completely cover its sensing region, it checks if any of them has assigned it a new sleep schedule and then goes back to sleep. A node in  $V_A$  might decide to shorten  $u$ 's sleep schedule if, for example, it loses

a certain number of its backup nodes<sup>1</sup>. If  $u$ 's sensing region is not covered, it declares itself a member of  $V_A$  and remains active for the remaining duration of  $V_A$ .

To demonstrate the operation of the opportunistic approach, consider a node  $v$  that has two backup covers  $s_1=\{A,B,C\}$  and  $s_2=\{D,E,F\}$ . The sleep schedule of these covers is illustrated in Fig. 6. When node  $v$  fails, the earliest backup nodes that wake up are  $A$  and  $D$ . Assume that they both do not reduce the size of the hole caused by  $v$ 's failure, (i.e., their sensing regions are covered by other nodes in  $V_A$ ) and thus they go back to sleep. Node  $B$ , however, could cover the entire hole and thus becomes active. Now, when nodes  $C$ ,  $E$ , and  $F$  wake up after  $B$ 's recovery action, they find that their sensing regions are completely covered by active nodes and go back to sleep.

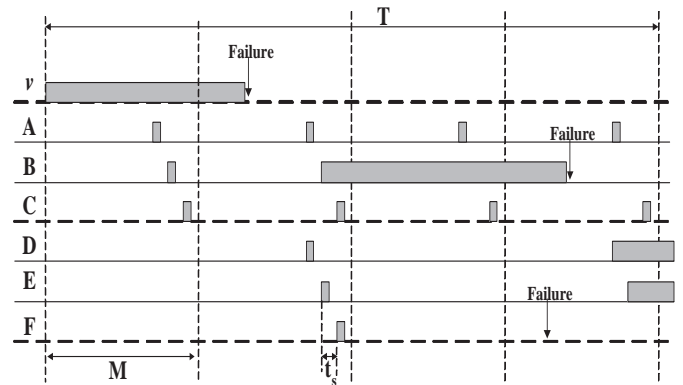


Fig. 6. Opportunistic sleep schedule for backup covers of  $v$ :  $s_1=\{A,B,C\}$  and  $s_2=\{D,E,F\}$ .

## V. PROPERTIES OF ROC

ROC's adaptability benefits can be easily deduced from the backup scheduling mechanism, described above. In this section, we study the impact of scheduling backup nodes as described above on the network's resilience and perceived energy savings.

### A. Fault tolerance

We first establish the relationship between having  $S$  backup covers for every node and network-wide  $k$ -coverage. Then, we compute the probability that a node's sensing region is covered by these backup covers.

**Proposition 1:** If the initial node deployment ensures that a point  $pt$  that lie in the sensing range of a node  $v$  is at least  $k$ -covered, then  $pt$  is also  $k$ -covered if  $S \geq k - 1$  backup covers of  $v$  are activated.

<sup>1</sup>We do not exploit this design option in our simulations.

**Proof.** The following three cases prove the proposition:

**Case I.** Assume that  $v$  could discover  $S = k - 1$  node-disjoint backup covers. In this case,  $pt$  will be covered by at least one node from each cover in addition to  $v$ .

**Case II.** Assume that  $v$  could not discover node-disjoint backup covers although  $pt$  is originally  $k$ -covered. As described in Section IV-B, ROC adds backup nodes that are least used in already-computed covers. This maximizes the number of backup nodes of  $v$ , resulting in at least  $k$ -coverage (see the example in Table I).

**Case III.** Assume that  $pt$  is only  $k'$ -covered, where  $k' < k$ . In this case,  $v$  will construct  $S < k - 1$  backup covers. These covers will contain *all* the neighbors of  $v$ , resulting in the maximum possible coverage of  $pt$  regardless of the degree of redundancy around it.

Now, we study the fault-tolerance properties of ROC. We focus on the opportunistic scheduling approach since the conservative approach provides more fault tolerance. Assume that the upper bound on the probability of a node's failure within one time slot is  $p_f$ . The objective is to have the sensing region of a failing node  $v$  covered by backup nodes within  $M$  time slots from  $v$ 's failure with reasonably high probability (e.g., above 95%). The probability that  $v$  fails within  $M$ -slots interval is  $P_f(M) = 1 - (1 - p_f)^M$ . If each of  $v$ 's covers has  $n_c$  nodes on average, then the probability that a cover is alive within  $M$ -slots interval is  $(1 - p_f)^{n_c M}$ . Consequently, the probability that the  $i^{\text{th}}$  backup cover is alive and active in any  $M$ -slots interval is  $P_a^{(i)}(M) = (1/i)(1 - p_f)^{n_c M}$  (the  $1/i$  term is removed in the conservative approach). Therefore, assuming that covers are node-disjoint, the probability that any backup cover is alive and active during an  $M$ -slots interval can be computed as:

$$P_a(M) = 1 - \prod_{i=1}^S [1 - P_a^{(i)}(M)]. \quad (1)$$

It follows that the probability that  $v$ 's region is covered in  $M$ -slots interval,  $P_c(M)$ , is given by:

$$P_c(M) = 1 - P_f(M) \times (1 - P_a(M)). \quad (2)$$

As an example, consider  $V_A$  operating under  $p_f=0.005$ ,  $M=20$  slots,  $S=3$ , and  $n_c=4$ . This results in  $P_f(M) \simeq 0.1$ .  $P_c(M) \simeq 98.37\%$  for opportunistic scheduling and 99.96% for conservative scheduling. Note that our approach is conservative in activating  $v$ 's covers since node  $v$  fails during the  $M$  slots with a probability that is much smaller than 1.

If  $p_f$  can not be estimated, it is thus favorable to activate all the  $S$  covers within every  $M$  slots. When

the next  $V_A$  is selected, non-failed nodes are selected as backup covers, so the above analysis holds for successive  $V_A$ 's. Compared to constructing a  $k$ -covered network for the entire operation of  $V_A$ , our ROC framework provides dynamic recovery, where every node can control the number of its backup covers and their probing frequency.

## B. Overhead

1) *Messaging and Processing:* Our backup scheduling mechanism does not require any extra message overhead than that required for the underlying DCS algorithm. All backup scheduling information can be piggy-backed on heartbeat or routing update messages, in addition to the control messages of the DCS algorithm. Thus, the ROC framework is efficient in terms of overhead communications. For computing backup covers, ROC incurs a processing overhead that is linear in the number of neighbors of  $v$ , which is insignificant because  $S$  is typically a small integer (e.g., 2 or 3).

2) *Energy Efficiency:* We study the energy efficiency of our design and compare it to constructing a  $k$ -cover. To focus only on the energy-efficiency aspects, assume that no failures occur during operation. We measure energy consumption by computing the average number of active nodes at each time slot ( $n_a$ ). Assume that a 1-cover typically contains  $N_c$  nodes. In a  $k$ -covered network,  $n_a = O(kN_c)$  nodes. In ROC opportunistic scheduling approach,  $n_a$  depends on several parameters, namely,  $M$ ,  $S$ , and  $n_c$ . Thus, it can be computed as follows:  $n_a = N_c(1 + \sum_{i=1}^S \frac{n_c}{iM})$ . The summation part represents a harmonic series of  $S$  elements. Therefore,  $n_a = N_c[1 + (n_c/M) \times (\ln S + A)]$ , where  $A$  is a constant. Note that we have assumed that a probing node will remain awake for an entire slot  $t_s$ , which provides a very conservative estimate of energy consumption.

Assuming large  $k$  and  $S$  to ignore the constants, the ROC approach is asymptotically more energy efficient than a proactive  $k$ -coverage approach if:

$$\frac{n_c \times \ln S}{M} < k \quad (3)$$

Assuming  $M = 20$ ,  $S = 2$ , and  $n_c = 4$ , the left-hand side of (3) is less than 2. In conservative scheduling, the  $\ln S$  term in (3) should be replaced by  $S$ , which is still efficient if  $n_c \times S < M$ . Controlling  $S$  provides the tradeoff between fault tolerance and energy efficiency. Choosing a large  $S$  favors fault tolerance at the expense of energy consumption, and vice versa. Our simulation experiments (Section VII) show that using a value of  $S = k - 1$  significantly improves fault tolerance with an energy cost close to that of a 1-cover.

## VI. SWITCHING BETWEEN COVERS IN ROC

We develop a novel technique for *offline cover update* (OCU). OCU selects a new cover based on any cover-selection algorithm. It lets the nodes in the current  $V_A$  compute the next network cover that should take charge after  $V_A$ 's operation interval is over. Since the nodes in  $V_A$  are always awake, they participate in the selection of a new cover by exchanging information about their 1-hop neighborhoods and simulating the roles of their sleeping neighbors (proxying). Below, we describe the operation of OCU and extend it for failure-prone networks.

### A. Initialization and Proxy Selection

The goal of OCU is to involve all the nodes in selecting a new cover without the need for having all of them awake. We assume that at initial deployment every node discovers its 1-hop neighbors and their sleep schedules, and advertises them. Every active node updates its neighbor list whenever a neighbor failure is detected or a new one is added, and periodically advertises this list. (Neighbor failure can be determined by keeping track of the neighbor's expected heartbeat messages.) Thus, 2-hop neighborhood information of active and sleeping nodes is known at every active node.

A node  $v \in V_A$  becomes the *proxy* of any neighbor  $u \in V_S$ , such that  $v$  uniquely satisfies a global criterion for  $u$ . For example,  $v$  can be the proxy of  $u$  if it is the closest neighbor of  $u$ . Node  $v$  thus decides the role that  $u$  will take in the next active cover. An example of proxy selection is provided in Fig. 7. Proxy selection resembles populating clusters in a clustered network. The difference in our case is that  $v$ , which resembles a cluster head, autonomously decides to be a proxy for some of its neighbors based on a pre-specified criterion.

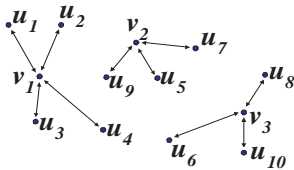


Fig. 7. Proxy selection based on neighbor proximity for non-active nodes  $u_1, \dots, u_{10}$ . Nodes  $v_1, v_2$ , and  $v_3 \in V_A$ .

### B. Details of OCU

For now, let's assume that there are no node failures until the end of  $V_A$ 's operation interval. Each node in  $V_A$  autonomously determines for which neighbors it is a proxy. The nodes in  $V_A$  start computing a new cover

after  $T'$  slots of their operation elapse, where  $T' < T$ .  $T'$  is set such that the remaining  $T - T'$  slots can accommodate the maximum expected convergence time of the employed DCS algorithm. Since nodes are not synchronized, nodes that finish  $T'$  slots first broadcast a message, triggering their neighbors to execute OCU. Every node that receives this trigger message broadcasts it once. We categorize the DCS algorithms into two broad categories and describe the OCU operation in the context of each of them.

1) *Collaborative DCS Algorithms (C-DCS)*: In C-DCS algorithms, a node makes a decision on whether to join  $V_A$  or not based on negotiations with its neighbors (e.g. [4], [5], [15]). In this case, a node  $v$  in the current  $V_A$  checks whether or not a decision can be made by itself or by any of the neighbors for which it is a proxy, according to the C-DCS algorithm. If so,  $v$  announces the decision so that other nodes can proceed with the C-DCS algorithm. For example, the DCS technique in [15] requires nodes with the highest remaining battery among their neighbors to decide first. In this case,  $v$  should estimate the remaining batteries of all its 2-hop neighbors by the end of the  $T$  slots to be able to decide which of them should make a decision. If  $v$  decides that neighbor  $u$  is ready to make a decision, it checks  $u$ 's sensing region. If  $u$  is completely covered by *future* active neighbors, then  $v$  announces that  $u$  will be asleep. Otherwise,  $v$  announces that  $u$  will be active. This process continues until all the nodes in the network are assigned their states in the next  $V_A$ .

2) *Autonomous DCS Algorithms (A-DCS)*: In A-DCS algorithms, a node makes a decision on whether to join  $V_A$  or not independently. For example, in [16], every node  $u$  sets a timer to a random duration  $T_u$ . After  $T_u$  expires,  $u$  executes some tests and makes its decision. Thus, in A-DCS algorithms, a node  $v \in V_A$  should simulate setting independent timers for each neighbor that it proxies. When a timer of a neighbor  $u$  expires,  $v$  uses the currently advertised neighbor information to simulate  $u$ 's actions. Node  $v$  then broadcasts  $u$ 's decision. A-DCS algorithms are easier to simulate by members of the current  $V_A$  since they require no neighborhood coordination, unlike the C-DCS algorithms.

### C. Failure-prone Networks

Two problems arise in failure-prone networks. The first problem is failure of an active node carrying the proxy role of several sleeping nodes. The second problem is the failure of a sleeping node that should be active in the next  $V_A$ . To address the first problem,

we extend the proxy selection approach given above as follows. Every active node competes for the proxy role of all of its sleeping neighbors that are within its sensing range by broadcasting a *bidding message*. Proxy bids are forwarded to neighbors that are two hops away to ensure that a unique proxy is elected for a sleeping node. The active neighbor that satisfies the proxy criterion for a sleeping node  $u$  wins the bid for  $u$ . For example, assume that  $v_1$  and  $v_2$  in Fig. 7 both bid for the proxy role of  $u_9$  and that the closer node should win. Although node  $v_1$  is aware of the neighbor list of  $u_9$  and thus is aware that  $v_2$  is closer to  $u_9$  than itself, it still bids for being the proxy of  $u_9$  since  $v_2$  may have failed. When  $v_1$  gets  $v_2$ 's bid, it gives up its candidacy for being  $u_9$ 's proxy. After bidding messages are exchanged, the cover-selection algorithm is executed as described above.

The second problem is handled as follows. We let all the sensors be awake at the start of a new  $V_A$  to heal any holes in the new cover. A node  $v \in V_A$  refreshes its neighbor list, selects its backup covers, and sets their probing schedule. A node  $u \notin V_A$  waits for a random interval of time to account for node asynchrony and to discover its active neighbors. If  $u$  discovers that its sensing range is not completely covered by neighbors in  $V_A$ , it adds itself to  $V_A$  and follows the procedure taken by active nodes. Otherwise, it gets its sleep schedule from its active neighbors and goes to sleep.

#### D. Benefits of OCU

1) *Flexibility*: OSU alleviates the need for having sensor nodes synchronized. It also significantly reduces the number of nodes involved in the cover selection process, resulting in less channel contention and collisions.

2) *Negligible overhead*: Cover selection is carried out only by active nodes in  $V_A$ . Thus, OCU significantly reduces the overhead of participation in cover selection.

3) *Energy efficiency*: Cover selection does not require extra energy cost by the nodes in  $V_A$  since they are already active. In addition, transition between covers only requires that every node checks that its neighbors in  $V_A$  cover its sensing range then go to sleep.

## VII. PERFORMANCE EVALUATION

In this section, we evaluate the ROC framework and contrast it to protocols constructing  $k$ -covers, where  $k \geq 1$ . We assume that a generic DCS protocol is employed to select an active set  $V_A$ , and we focus on the operation of one  $V_A$ . The DCS protocol (which we refer to as “ $k$ -cov”) operates as follows. Every node  $v$  sets a timer for a random backoff delay and listens to

its neighbor messages. When  $v$ 's timer expires, it checks whether its entire sensing region is  $k$ -covered. If not,  $v$  joins  $V_A$ . After cover selection,  $k$ -cov lets every  $v \notin V_A$  go to sleep. ROC uses the DCS algorithm to select a 1-cover, and computes the sleep schedule of every  $v \notin V_A$ . Note that the performance of ROC is insensitive to the efficiency (i.e., minimality) of the selected covers. We compare ROC to  $k$ -cov and RIS [9] for ( $k > 1$ ). RIS is an SCS protocol in which a node independently decides to be active or asleep based on a fixed probability. We use the technique in [9] to compute the necessary wake up probability for a fixed number of nodes.

Our metrics are fault tolerance and energy efficiency. “Coverage quality” is a measure of fault tolerance and is defined as the minimum fraction of area coverage at any time slot. We compute the consumed energy during the cover operation and use it as a measure of energy efficiency. The following parameters are varied: (1) the sensing radius  $R_s$ , which determines the sensing density in the network, (2) the tolerance interval  $M$ , and (3) the failure probability  $p_f$ . We compare ROC to  $k$ -cov and RIS under random failures, in addition to location-based (clustered) failures. We use opportunistic scheduling in all the experiments in which  $p_f$  can be estimated.

We assume that  $N=500$  nodes are randomly deployed in a  $50 \times 50$  meters<sup>2</sup> field. A cover operates for  $T = 1000$  slots, and the slot interval  $t_s = 1$  minute. When a backup node wakes up, it remains awake for  $0.2t_s$  seconds to probe its active neighbors. A node consumes  $24 \times 10^{-3}$  Watts while active and  $3 \times 10^{-6}$  Watts while asleep. Note that the active interval is dominated by idle-listening and thus the above value includes the energy consumed in transmission and reception. To focus on unexpected failures, we assume that no nodes deplete their energy during the  $T$  slots. We developed an event-driven simulator in which ROC,  $k$ -cov, and RIS are implemented. Every result below is the average of 10 experiments of different random configurations. For every configuration, we set a *death schedule* for the nodes in order to compare all the techniques under the same conditions.

#### A. Energy efficiency of ROC

To fairly compare all approaches in terms of energy efficiency, we study their operation under no failures (i.e.,  $p_f = 0$ ). In this experiment, the network is completely covered throughout the entire cover operation. We use  $M = 50$  slots for ROC and vary  $k$  and  $S$ . Figure 8 shows that the consumed energy in ROC is almost similar for both  $S = 1$  and  $S = 2$ . ROC's energy consumption is 40-80% less than that of  $k$ -cov and RIS for  $k > 1$ . This

is a significant improvement, especially that ROC's fault tolerance properties also outperforms the two protocols.

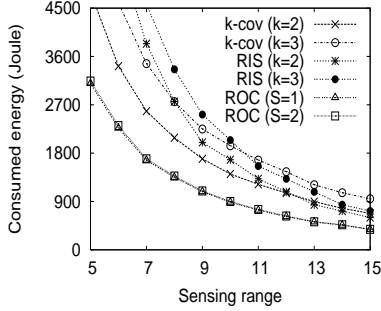


Fig. 8. Energy consumption under no failures.

### B. Random Failures

In this section, we study fault tolerance and energy efficiency of ROC under a random failure model. Note that ROC heals the uncovered holes during the operation of  $V_A$  by activating backup nodes. Thus, more nodes are activated in ROC than the other protocols, making energy comparison unfair to ROC. We use the minimum reported quality of coverage during the operation of  $V_A$  as the measure of fault tolerance.

We first study the effect of node density.  $R_s$  controls the resulting density of active nodes. We fix  $M=50$  slots and  $p_f=0.001$ . Figure 9(a) illustrates that ROC outperforms  $k$ -cov and RIS in terms of coverage quality, even for 3-coverage. We examined running snapshots of these approaches and they showed that ROC revives coverage at the end of every  $M$  interval and does not let it deteriorate as in  $k$ -cov or RIS. The perceived resilience gains of ROC requires less energy cost than the other protocols for both  $k = 2$  and  $k = 3$  (energy consumption results are not shown for brevity). Experiments with higher values of  $S$  and  $k$  showed that ROC is about 75% more energy-efficient than  $k$ -cov and is about 90-100% more energy-efficient than RIS.

We also study the effect of  $M$  on ROC reliability. Figure 9(b) shows coverage quality when  $M$  varies from 1% to 20% of  $T$ . In this experiment, we used  $p_f = 0.001$  and  $R_s = 10$  meters.  $k$ -cov is not sensitive to  $M$ . As expected the fault tolerance of ROC slightly deteriorates as  $M$  increases because backup nodes wake up less frequently. However, ROC still outperforms  $k$ -cov and RIS for corresponding values of  $S$  and  $k$ . Figure 9(c) shows that ROC is significantly more energy efficient than  $k$ -cov and RIS for  $k > 1$ . The figure also shows that as  $M$  becomes smaller, the overhead imposed on the

backup covers is magnified. For critical applications, we recommend that the application either use small values of  $M$  (compared to  $T$ ), or use the conservative scheduling approach for large values of  $M$ .

Finally, we study the effect of the failure probability ( $p_f$ ). In this experiment, we fix  $R_s=10$  meters,  $M=50$  slots, and use  $k=2$  and 3, and  $S=1$  and 2. Figure 9(d) shows coverage quality for  $p_f = 10^{-4}$  to 0.004. The figure indicates that ROC outperforms the two other protocols for all values of  $k$ . The improvement in minimum coverage in ROC is about three times that of  $k$ -cov or RIS at high failure rates. This is attributed to ROC's dynamic nature. Energy consumption results show a similar trend to the ones mentioned above.

### C. Clustered Failures

In this section, we study the performance of ROC under non-uniform probability of failure. We assume that failure is dependent on the sensor location, where nodes closer to the bottom leftmost coordinate (0,0) have the least probability of failure, and vice versa at the top rightmost coordinate (as in Fig. 1). For a node  $v$  residing at  $(x,y)$ , it has a  $p_f = 0.004 \times (x + y)/2L$ , where  $L$  is the length of the field. Since failures are not random, we use the conservative scheduling approach and let all the  $S$  backup covers wake up every  $M=50$  slots. In this setting, we demonstrate the operation of ROC with variable  $S$  values (denoted by "ROC-VAR"). Every node selects its own number of backup covers between 1 and 4 based on its distance from the origin. Far-distance nodes (away from (0,0)) use larger  $S$  than closer ones. Results in Fig. 10(a) illustrate the minimum coverage quality at different sensing ranges and indicate that ROC is significantly more resilient than  $k$ -cov and RIS under clustered failures. Figure 10(b) shows that although ROC activates more nodes for recovery, its energy consumption is still significantly less than  $k$ -cov and RIS at  $k = 3$ . Using ROC-VAR shows the best fault tolerance, especially at smaller ranges. We also plot a snapshot of field coverage as time progresses in Fig. 10(c). It is clear that ROC-VAR's performance is more stable than  $k$ -cov throughout the entire cover operation. The unstable behavior of RIS in the figure is due to its probabilistic nature.

## VIII. CONCLUSIONS

In this work, we presented ROC (resilient online coverage), a framework that leverages cover-selection algorithms to achieve adaptability, fault tolerance, and energy efficiency. ROC allows different degrees of redundancy

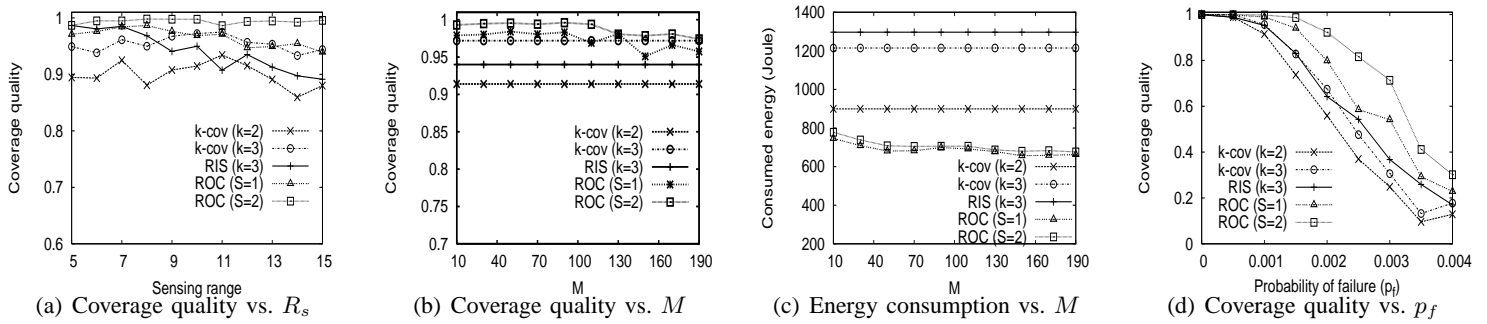


Fig. 9. Performance of ROC under random failures.

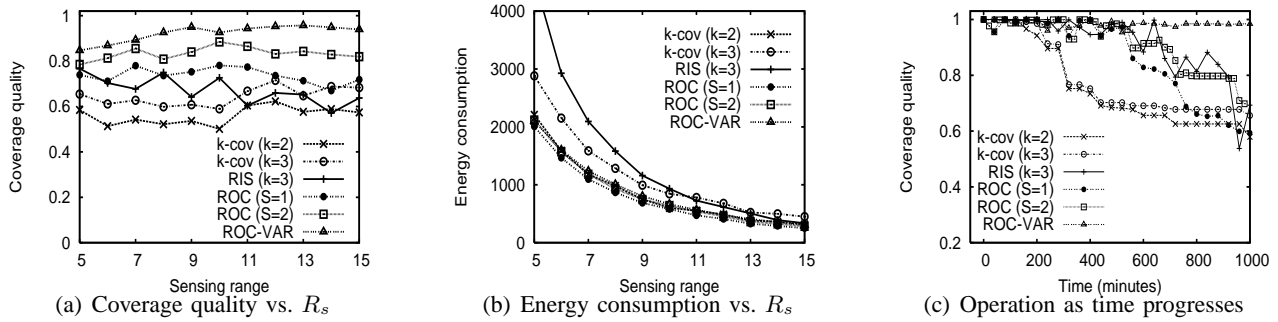


Fig. 10. Performance of ROC under clustered failures.

across the field, in addition to controlled speed of recovery of failures. It also incorporates a novel technique for offline cover update (OCU) to facilitate asynchronous transition between covers. Simulation experiments show that ROC is more efficient than network-wide  $k$ -coverage and probabilistic activation mechanisms.

## REFERENCES

- [1] W. Ye, J. Heidenmann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. of the IEEE INFOCOM Conf.*, New York, June 2002.
- [2] G. Lu, N. Sandagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *Proc. of the IEEE INFOCOM Conf.*, March 2005.
- [3] T. Yan, T. He, and J. Stankovic, "Differentiated surveillance for sensor networks," in *Proc. of the ACM Conf. on Embedded Networked Sensor Systems (ACM SenSys)*, November 2003.
- [4] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration for energy conservation in sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 1, pp. 36–72, August 2005.
- [5] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 1, pp. 89–124, January 2005.
- [6] H. Gupta, S. Das, and Q. Gu, "Connected sensor cover: Self organization of sensor networks for efficient query execution," in *Proc. of the ACM International Symposium on Mobile and Ad-Hoc Networking and Computing (MobiHoc)*, June 2003.
- [7] F. Koushanfar, M. Potkonjak, and A. S. Vincentelli, "Fault tolerance techniques for wireless ad hoc sensor networks," in *IEEE Sensors*, 2002.
- [8] F. Ye, G. Zhong, S. Lu, and L. Zhang, "PEAS: A robust energy conserving protocol for long-lived sensor networks," in *Proc. of the IEEE Int'l Conf. on Distributed Computing Systems*, 2003.
- [9] S. Kumar, T. H. Lai, and J. Balogh, "On  $k$ -coverage in a mostly sleeping sensor network," in *Proc. of the ACM MobiCom Conf.*, September 2004.
- [10] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *Proc. of the ACM Workshop on Sensor Networks and Applications (ACM WSNA)*, Sep. 2003.
- [11] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *Proc. of the IEEE INFOCOM Conf.*, Anchorage, Alaska, April 2001.
- [12] M. Cardei, M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *Proc. of the IEEE INFOCOM Conf.*, March 2005.
- [13] R. Iyengar, K. Kar, and S. Banerjee, "Low-coordination topologies for redundancy in sensor networks," in *Proc. of the ACM International Symposium on Mobile and Ad-Hoc Networking and Computing (MobiHoc)*, May 2005.
- [14] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *Proc. of the ACM MobiCom Conf.*, September 2004.
- [15] O. Younis, M. Krunz, and S. Ramasubramanian, "On maximizing coverage-time in location-unaware wireless sensor networks," University of Arizona, Tech. Rep., July 2006. [Online]. Available: <http://www.ece.arizona.edu/~younis/papers/luc-tr.pdf>
- [16] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *Proc. of the First ACM Workshop on Wireless Sensor Networks and Applications*, September 2002.