ROC: Resilient Online Coverage for Surveillance Applications

Ossama Younis, *Member, IEEE,* Marwan Krunz, *Fellow, IEEE,* and Srinivasan Ramasubramanian, *Member, IEEE*

***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 redundant node deployment and tolerate unexpected node failures. Little attention was given to studying the tradeoff between fault tolerance and energy efficiency in sensor coverage.**

In this work, our objectives are two-fold. First, we aim at rapidly restoring field coverage under unexpected sensor failures in an energy-efficient manner. Second, we want to flexibly support different degrees of redundancy in the field without needing centralized control. To meet these objectives, we propose design guidelines for applications that employ distributed cover-selection algorithms to control the degree of redundancy at local regions in the field. 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 referred to as "resilient online coverage" (ROC) framework. A key novelty in ROC is that it allows every sensor to control the degree of redundancy and surveillance in its region according to current network conditions. We analyze the benefits of ROC in terms of energy efficiency and fault tolerance. Through extensive simulations, we demonstrate the effectiveness of ROC in operational scenarios and compare its performance with previous surveillance techniques.

***Index Terms*—Sensor networks, coverage, fault-tolerance, energy-efficiency, distributed algorithms.**

I. Introduction

In surveillance applications, sensors are deployed in large numbers to improve the quality of field coverage. Under such redundant deployment, network lifetime can be prolonged by periodically putting nodes to sleep to save their batteries. Two approaches were proposed to exploit the benefits of putting nodes to sleep. The first is a "MAC-based" approach, in which a node uses a duty cycle to schedule its sleep and wakeup periods based on the expected traffic pattern (e.g., [24], [15], [17]). This approach has been implemented in real systems, and performance results have shown that it is effective in reducing energy consumption under sustained load patterns. However, it does not adapt to changing network conditions, such as varying load or unexpected node failures. The other approach is an "application-based" 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., [22], [21], [27], [9], [3], [18], [19], [13], [11]). This approach can balance the load among all the sensors by periodically selecting new covers. It can also control the quality of surveillance and adapt to changing network conditions. In this paper, we focus on application-layer coverage.

We study monitoring applications in which a field (or target) is to be covered by at least one sensor, and that the time required to heal a "hole" (unmonitored region) is upper-bounded. We refer to this time as the "tolerance interval" (M). M represents the time duration in which detecting an event or a phenomenon is still possible (e.g., chemical or radiation activity). We consider scenarios in which a sensor may fail due to environmental factors, e.g., lava coming from a volcano or forest fire. Failures can also be due to adversarial factors, e.g., an explosion in a combat region[1]. For energy conservation, we assume that 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.

To accommodate failures, the network designer increases the "degree of redundancy" in the field, defined as the number of nodes that simultaneously monitor a point in the field. The degree of redundancy can also be defined in terms of the frequency of checking whether a point in the field is monitored or not. Several design alternatives can be employed for failure recovery in a failure-prone environment. The most reliable approach is to refresh covers *more frequently* to allow new active nodes to replace failing ones. This approach has high overhead because cover refreshment is a network-wide process. It is also difficult to estimate the required time between successive cover refreshments when the probability of node failure takes arbitrary values or if the traffic pattern and network mission dynamically change. Another failure recovery approach is to select a "k-cover," which ensures that every point (target) in the field is covered by at least k sensors, where $k > 1$. Despite its simplicity, this approach has two drawbacks. First, a k-cover that is active for a long time cannot adapt to different failure patterns, especially if failures tend to be clustered around specific areas as in combat regions. 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 fewer 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 such that each sensor monitors the region at the rate of 1 observation per k units. Such frequency relaxation

This research was supported in part by NSF (under grants CNS-0721935, CNS-0904681, IIP-0832238), Raytheon, and the "Connection One" center. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

[1]Research on configurable surveillance is important for projects, such as DARPA Future Combat Systems (FCS) and related programs, such as Warfighter Information Network-Tactical (WIN-T) [1].

maintains the average number of observations made per point in the cover over a large time scale. Although every point is guaranteed to be observed at least $2k$ times in an interval of $2k$ units, 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 can be decomposed into k node-disjoint covers. Typically, this is not possible. 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 when a node is in the active mode than in the sleep mode. Therefore, activating a large number of nodes increases the failure rate of the system, resulting in more node failures per time unit.

A desirable approach is to select a minimal set of sensors to cover the region and an additional backup set that is periodically woken up to check on the active set. In this approach, only a small number of nodes is active, but each active node is protected by a few neighboring (backup) nodes. It is also desirable to have adaptable coverage when the failure rate varies from one region to another, as depicted in Figure 1. In this figure, the node failure rate in region R_i is smaller than than in region R_j, where $1 \leq i, j \leq 4$ and $i < j$. It is thus required to have a higher degree of redundancy in R_j than in R_i. Providing an adaptable coverage behavior that exploits the tradeoff between fault tolerance and energy efficiency has not been given enough attention in the literature.

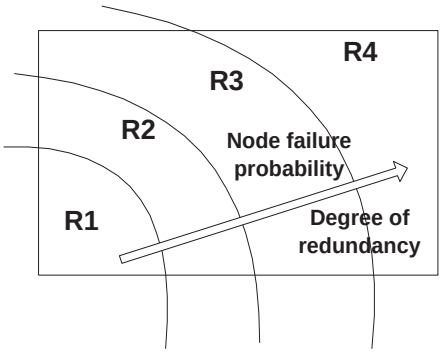


Fig. 1. Four regions in the field with different failure rates.

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 wakeup is necessary. Node synchronization is practically difficult in networks with sleeping nodes. In addition, clocks drift as time progresses, and this effect is magnified when covers operate for several days or weeks. These clock drifts may result in some nodes being always late in participating in the cover-selection process, while others are always early and participate in the active cover until they die. This has a detrimental effect on network lifetime, as illustrated in Figure 2. In the figure, five nodes with different sensing ranges are deployed. 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. This example calls for periodic cover refreshment that involves all the nodes so as to maximize the network lifetime.

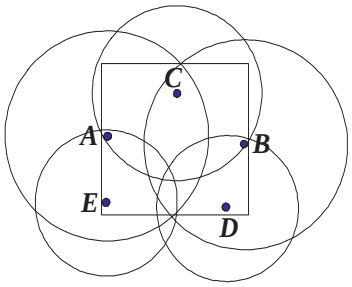


Fig. 2. Five sensors covering a square region. The circles denote the sensing region of each sensor.

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 autonomously select a set of backup covers for its sensing region, and schedule them to periodically check for holes. This results in "node-driven" control of the degree of redundancy and surveillance by allowing every node to self-configure at run-time. This approach is more flexible and adaptable than "network-driven" recovery, in which a fixed strategy is used by all nodes. Adaptability stems from two points. First, every node controls the degree of redundancy in its region to satisfy regional surveillance requirements while accounting for the surrounding network conditions, such as surviving neighbors and frequency of occurring events. Second, every node autonomously adjusts the schedule of its backups to maintain a certain speed of recovery from failures in its region. 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. This alleviates the overhead and possible application interruption when switching between covers. Our integrated approach for online, fault-tolerant surveillance and switching between covers is referred to as *resilient online coverage* (ROC). We show through analysis and simulations that ROC is more efficient and flexible than constructing static k-covers or employing randomized sleep schedules as in [13].

Note that our interest is *not* to design new protocols for cover selection, nor is it to study the performance of existing cover-selection protocols. The benefits of ROC extends beyond designing centralized or localized coverage algorithms. Even with localized algorithms, the goal is to satisfy regional coverage constraints to ensure that a global coverage objective is achieved. No pre-configuration of the required coverage quality is forced on the sensors throughout its operation. ROC can support different types of coverage problems, e.g., area coverage (as in most related work) and barrier coverage (as in [20]). It also supports different coverage requirements, e.g., full coverage (as in regional surveillance for chemical or biological activity) as well as partial coverage (as in the tracking of moving objects). ROC assumes that each sensor

is an autonomous system that has the authority to "decide how to cover" its sensing range. This flexibility enables every sensor to exploit the tradeoff between energy efficiency and fault-tolerance at run-time.

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, integrated into an existing cover-selection algorithm. Finally, Section VIII provides concluding remarks.

II. Related Work

Fault-tolerance in wireless sensor networks has received significant attention. In [12], the authors proposed algorithms for using heterogeneous types of sensors as backup for each other to enable efficient multimodal data fusion. In PEAS [23], probabilistic sleep/wakeup was proposed to maintain network connectivity under unexpected failures. Selecting a k-cover was typically proposed to avoid having holes in the monitored region [13], [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 broad categories. We refer to the first category as the *deterministic cover selection (DCS)* approach, which comprises the majority of research in this domain. DCS relies on selecting a set of nodes that guarantees coverage of at least α% of the field, where $\alpha \leq 100$ [16], [3], [22], [21], [27], [11], [13]. Protocols in this category can be further classified into centralized (e.g., [16], [3]) and distributed (e.g. [4], [21], [27], [11]). We focus on distributed protocols in this work. Since determining an optimal cover is NP-hard [6], cover-selection protocols often employ heuristics, such as timer expiration [27], [19] or specific node preferences [4], [7]. The work in [10], [19] provided the necessary conditions for a node to determine whether its area is covered by other nodes. In [25], we provided sufficient conditions for determining redundancy in the absence of location information.

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. Its primary drawback is that unexpected node failures may reduce the quality of coverage until a new cover is selected. Constructing a k-covered network may not solve this problem if failures tend to be clustered. The protocol in [22] allows neighboring nodes to coordinate to determine their working schedules. However, the algorithm requires the availability of special hardware to activate sleeping nodes for fault tolerance.

We refer to the second category of schemes for selecting active covers as the *stochastic cover selection (SCS)* approach. This approach includes RIS [13] and PECAS [8]. In these protocols, nodes decide whether or not to sleep and for how long based on a certain probability distribution. The SCS approach requires little (or no) node coordination and only coarse node synchronization. It is quite easy to implement and deploy. However, it provides only asymptotic guarantees on field surveillance under certain conditions on node distribution, which may not be satisfied in practice. The active cover at any instant in time can also be far from minimal. In addition, some protocols employ frequent sleep/wakeup that makes the topology very dynamic, resulting in continuous invalidation of the discovered routes. Kumar et al. [13] provided theoretical bounds on the number of nodes required to achieve k-coverage under different models of node deployment. They also proposed a randomized sleep scheduler (RIS), which we later use for baseline comparison. RIS is easy to deploy. However, nodes have to use a *conservative* wakeup probability, depending on how critical the mission is. The surveillance approach in [8] allows nodes to switch among sleep/wakeup states to collaboratively maximize the lifetime of individual sensors. The protocol was optimized for tracking target trajectories, and not for field coverage under arbitrary node distributions and failures.

A sub-category of coverage problems targets barrier coverage, in which sensors are deployed to surround a confined region. An example of a barrier coverage algorithm is given in [20], which executes a number of rounds to deploy the minimal number of sensors. In terms of requirements, some applications may only need partial coverage. For example, the "Trap Coverage" model [2] allows holes in a cover, but guarantees a maximum diameter of any hole in the cover to ensure a bounded time for the detection of objects. For partial coverage, the trigger to re-compute sensor covers is more related to how the diameter of coverage holes varies as more nodes fail.

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 wakeup for selected backup nodes to achieve fault tolerance, while reducing the required energy cost. It can incorporate any underlying cover selection algorithm but controls when the algorithm is invoked, which active nodes should invoke it, and at what level should coverage be computed (sensing range vs. local region vs. network-wide).

III. Problem statement

The notation used throughout the paper is given in Table I.

A. Assumptions

We consider applications that require continuous field surveillance in a hostile environment (e.g., detecting chemical activity in a military field or early indication of fire in a forest). We assume the following:

- Node deployment ensures that every point in the field is at least k-covered. In [13], the authors prove that all the points in the field are almost always k-covered if a certain inequality holds between k, n (number of nodes), r (coverage radius), and p (probability that a node is active at any time instance). Assuming random uniform

TABLE I
NOTATION

k	Degree of coverage (typically ≥ 1)
V_A	Active 1-cover
V_S	Sleeping set in the network
S	Number of node-disjoint backup covers of a node
$s_i(v)$	Backup cover i of node v
t_s	Duration of a time slot (in msec.)
T	Active duration of a cover (in slots)
M	Number of slots in which lost monitoring is tolerated
$N(v)$	Set of neighbors of node v
$n_b(v)$	Number of neighbors of v
$R_s(v)$	Sensing range of node v (or simply R_s)
$R_c(v)$	Communication range of node v (or simply R_c)
P_{v,R_s}	Prob. that a point is covered by v having range R_s
$f_v(r)$	Prob. distribution function for point coverage of v
p_f	Prob. of failure in one time slot
$P_f(M)$	Prob. that v fails within an M-slots interval
$P_a(M)$	Prob. that backup cover is alive during M-slots interval
γ	Minimum coverage requirement ($0 \leq \gamma \leq 1$)

deployment and using a slowly growing function $\phi(np)$, the inequality is defined as:

$$\frac{np\pi r^2}{\log(np)} \geq 1 + \frac{\phi(np) + k\log\log(np)}{\log(np)}. \qquad (1)$$

- At every sensor, the time is divided into large-scale slots of size = t_s units. Each slot represents the expected duration between two successive sensor reports, as defined by the application.
- 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 slot durations, where $1 \leq M \leq T$. M determines the required speed of recovery (typically $M \ll T$). An example that shows the relationship between T, M, and t_s is given in Fig. 3
- 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 [25] selects covers without relying on node locations, unlike the ones in [22], [11]. Detecting covered regions or holes in the active cover is done using the same approach that the DCS algorithm uses to construct covers.
- Nodes are stationary, can be added anytime during network operation, and may not be synchronized.
- Node failures may occur at random locations across the field or may occur only in confined regions. Random failures correspond to sensor hardware impairments that may occur anywhere in the field, independent of node location. On the other hand, clustered failures occur due to external stimulation in specific regions in the field, such as attacks in a military field or originating fire in a forest. Our approach does *not* require knowledge of the failure rate.
- Connectivity is achieved by enforcing constraints on the ratio between the sensing and communication ranges (see classification in Section V-D). Our framework is general and can incorporate techniques that compute connected covers under no range constraints (e.g., [11], [7]).

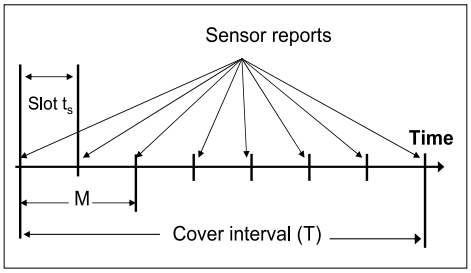


Fig. 3. An example showing the relationship between T, M, and t_s.

B. 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.
2) A framework to allow controlling the degree of redundancy in the field. Every node autonomously decides the degree of redundancy in its neighborhood, as described in Section IV. Thus, coverage in different regions may have different degrees of redundancy.
3) Efficient, low-overhead switching between active covers without the need for synchronized nodes.

In our model, the degree of redundancy refers to the frequency per unit time of checking whether a point in the field is monitored or not. As indicated earlier, a node determines the degree of redundancy based on network conditions. For example, a node can monitor the traffic in its neighborhood or the failure rate of its neighbors during the operation of a cover. Other metrics to consider are the cost (energy or communication) associated with node activation and the deployed node density.

IV. RESILIENT ONLINE COVERAGE (ROC)

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

A. Overview and Design Rationale

At the time of deployment, every nodes periodically broadcasts "HELLO" messages to inform its neighbors about itself. Each node also keeps a list of its 1-hop neighbors, and periodically announces this list. Thus, every node is made aware of its two-hop neighbors. ROC requires an active 1-cover (V_A) be selected using any DCS algorithm. The remaining nodes are assigned to the tentative *sleeping set (V_S)*. Initially, all the nodes participate in selecting V_A. 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 Figure 1, we adopt a novel recovery mechanism in which every active node selects a subset of its neighbors that are assigned to V_S (referred to as a "backup cover") and schedules

them to periodically wake up. The backup sets of every active node are proactively selected after a DCS algorithm constructs a 1-cover. Our backup selection mechanism provides opportunity for a node to self-configure autonomously according to surrounding network conditions. Energy is also conserved by intelligently scheduling the activation of the nodes in the backup covers.

A finite state machine that describes the states and transitions of a node is shown in Figure 4. All nodes initially start in the "ASLEEP" state. If a node is selected in V_A, it moves to the "ACTIVE" state and remains there for T time slots. Otherwise, it remains in the ASLEEP state. Among nodes in the ASLEEP state, some nodes are selected by their neighbors to serve in backup covers. Backup nodes periodically wake up and go to the "PROBING" state. When a backup node discovers an unmonitored region, it joins V_A and moves to the ACTIVE state. Non-backup nodes in V_S have no role during the operation of V_A and remain asleep throughout the T slots.

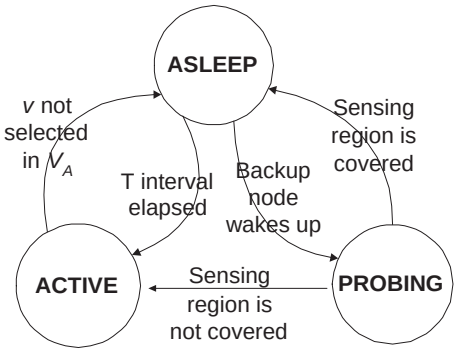


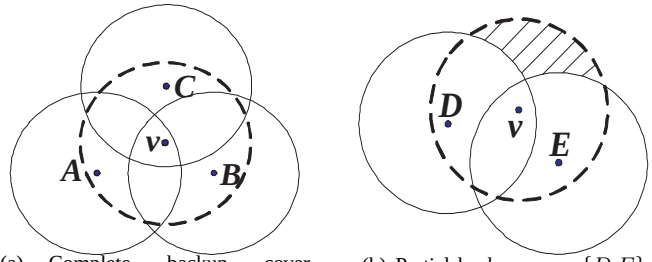
Fig. 4. Finite state machine for a node v employing ROC.

It is important to emphasize that it is *not* possible in our model to adopt a reactive recovery mechanism after a hole occurs. This is because there is no centralized entity that can invite sleeping nodes to wake up. In addition, we do not assume that nodes carry any special hardware to allow activation while being asleep (e.g., [22]). Also note that if the tolerance interval is large, it is possible to use all the neighbors of v as backups. This is not favorable, however, because every neighbor needs to exchange handshake messages with v when it wakes up to ensure that v is alive. In addition to being an unnecessary overhead, these redundant messages increase the opportunity of packet collisions and data losses in the network, especially under high node densities. Switching between activity modes also makes a node more prone to failures. Thus, it is desirable to minimize the number of nodes in the backup covers.

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 Figure 5(a) for $S = 1$.

Note that constructing S node-disjoint backup covers is a sufficient condition for k-coverage, while having k-coverage in the field does not necessarily mean that S independent backup covers can be found (we study the relation between having S backup covers and k-coverage in Section V). If v cannot find S backup covers, it computes the maximum possible number of covers that it can find. If the node density drops during the network operation below k, then the S covers may only be partial covers, as demonstrated in Figure 5(b).



(a) Complete backup cover $\{A,B,C\}$ (b) Partial backup cover $\{D,E\}$

Fig. 5. Two backup covers of node v.

Node v does not need to enumerate all the possible backup covers as the covers do not have to be minimal 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, we try to compute covers that are as node-disjoint as possible. To construct a backup cover (e.g., s_1), v employs the following greedy algorithm. Assume that $N(v)$ is the set of neighbors of v. Node v first sorts $N(v)$ according to an optimization parameter. For example, v can choose a neighbor that has the largest percentage of area in common. A neighbor that is a member of backup covers for many nodes other than v is also a favorable candidate, compared to one that is not selected by any other node. Based on the sorted list, v selects the top-listed neighbor to be in s_1. Then, v sorts the neighbors again according to the same optimization criterion and selects another neighbor to add to s_1. This is similar to how a typical DCS algorithm selects a cover in the network.

When the nodes in s_1 form a cover of v (or adding more neighbors does not increase v's covered area), v starts computing the next set s_2 and so on. The first node u that is added to any backup cover s_i ($i > 1$) should satisfy that $u \in N(v) - (\bigcup_{j<i} s_j)$. Neighbors are then added to s_i, giving preference to those having the least occurrences in the already computed backup covers of v. Figure 6 provides the pseudo-code of the algorithm used for selecting backup covers of a node v.

An example is shown in Table II for selecting $S = 3$ backup covers of a node v that is shown in Figure 7. 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. Note that our approach ensures that no partial covers of v are selected unless all full covers are.

Proposition 1: Assume that a node v has n_b neighbors ($n_b = \|N(v)\|$) and the optimal neighbor cover size is C_{opt}. Also assume that a neighbor is added to the cover if it is the one that covers most of v's remaining uncovered region, as described in algorithm A (Figure 6). A 1-cover that is

```
01. N_b = 1-hop neighbors of v
02. For i = 1 to S    //backup covers
03     BackupCoverSet[i] = φ
04. For i = 1 to N    //number of nodes
05     Freq[i]=0      //occurrence of i in backup covers
06. For i = 1 to S
07.   isCoverDone = FALSE
08.   N_c = 1-hop neighbors of v
09.   While (isCoverDone == FALSE)
10.    IF not empty N_b
11.        Sort N_b based on an optimization parameter
12.        Pick u: the node on top of N_b
13.        Add u to BackupCoverSet[i]
14.        N_b = N_b − u
15.        Freq[u] = Freq[u] + 1
16.        IF area(v) is covered
17.              isCoverDone = TRUE
18.    ELSE
19.        Pick u: {u ∈ N_c & N_c ∉ BackupCoverSet[i]
                    & u has the least Freq
                    & u maximizes the covered area}
20.        IF u exists
21.          Add u to BackupCoverSet[i]
22.          N_c = N_c − u
23.          Freq[u] = Freq[u] + 1
24.          IF area(v) is covered
25.                isCoverDone = TRUE
26.        ELSE
27.          isCoverDone = TRUE
```

Fig. 6. Algorithm A: Selecting S backup covers for node v.

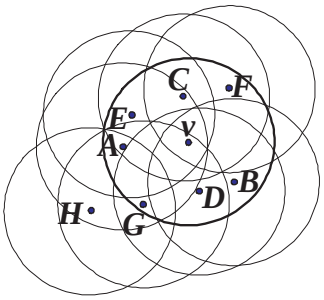


Fig. 7. A configuration containing node v and its neighbors.

TABLE II
SELECTING THREE BACKUP COVERS FOR A NODE v THAT IS SHOWN IN FIGURE 7.

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}

To assess the quality of our proposed greedy approach, we compare the cover size that is computed by our approach to the optimal one computed by exhaustive search. Since exhaustive search has exponential time complexity, we are limited to small-sized networks of about 11 nodes. Figure 8 shows the average cover size for 500 different experiments. In each experiment, a target node v is put at (0,0) and a number of nodes (neighbors) are randomly deployed within its sensing range. The theoretical lower bound on the cover size is three. The figure shows that the average optimal cover size in these experiments is very close to three. Our greedy approach generates covers that are about 6-10% larger than the experimental optimal or the lower bound. We also tried to select covers based on other optimization parameters, such as the node's remaining battery. Results were slightly less efficient but very close to those of the greedy approach used in the original experiment. We analyze the fault tolerance and overhead properties of this algorithm in Section VII.

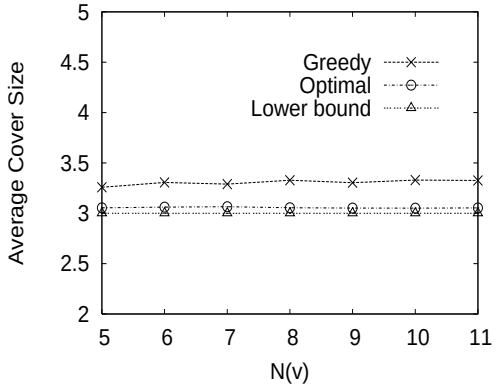


Fig. 8. Comparing the cover size of the greedy approach to the optimal one.

computed by the above greedy algorithm has a size C, where $C \leq \log n_b C_{opt}$.

Proof. The problem of computing covers can be mapped to the set cover problem that has been well studied in set theory. To prove the proposition, assume that the sensing range of v is discretized into a number of points, and each of the n_b neighbors covers a subset of these points. The problem is thus to select the minimum number of subsets whose union spans all the points. According to [6], a greedy algorithm that gives preference to subsets carrying more uncovered points would generate covers of size $\leq \log n_b C_{opt}$, where n_b is the number of subsets. Details of the proof can be found in [6] and are omitted here for brevity.

C. Scheduling Backup Covers

After selecting the backup covers, v notifies the nodes in the backup covers 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 backup nodes should wake up periodically every M' slots, where $M' \leq M$. This ensures that ROC provides enough backups as those of k-coverage (as proven below). 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 of an active node in V_A can cover the hole caused by another failing node in V_A whose backup nodes

are still asleep. Thus, we propose the following opportunistic approach for successive activation of backup covers.

Let s_i be the i^{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. Subsequently, 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 cycle assigned by any of its 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). On the other hand, the conservative approach should be employed when node failures are arbitrary. It is worth noting here that node synchronization is *not* necessary. However, a backup node u follows the schedule that is proposed by its master node v and based on v's current clock without having to update u's clock.

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 routing updates from its 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 routing updates have been sent by its active neighbors. If u 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[2]. If u's sensing region is not completely covered, it declares itself a member of V_A and remains active throughout the remaining duration of V_A's operation.

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 Figure 9. When node v fails, the earliest backup nodes that wake up are A and D. Assume that both A and D do not reduce the size of the hole that has resulted from v's failure and thus they go back to sleep. Node B, however, can cover the entire hole and thus remains active after probing. 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 they go back to sleep.

V. Properties of ROC

ROC's adaptability benefits can be easily deduced from the backup scheduling mechanism, described above. In this

[2]We do not exploit this design option in our simulations.

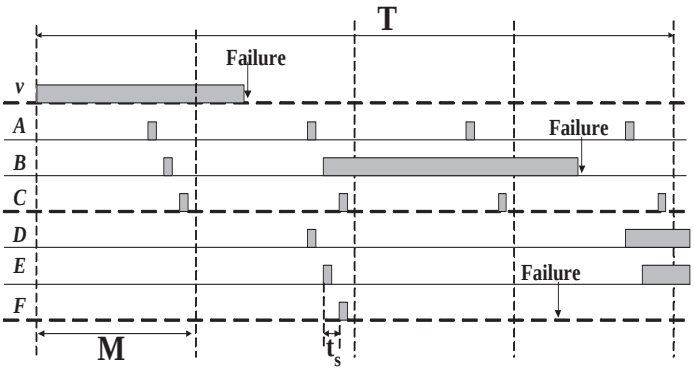


Fig. 9. Opportunistic sleep schedule for backup covers of v: s_1=$\{A,B,C\}$ and s_2=$\{D,E,F\}$.

section, we first provide a generalized methodology for a sensor to compute its area coverage, assuming a probabilistic coverage model. Then, we study the impact of ROC on the network's resilience and perceived energy savings. Note that we assume an upper bound on node failure rate for the analysis of fault-tolerance. However, ROC operation does not require knowledge of the error rate.

A. **Probabilistic Area Coverage**

Assume that a point pt that lies at a distance D_0 from v is covered with probability $P_{v,R_s(v)}(pt)$, which depends on $R_s(v)$ and D_0. If $D_0 > R_s(v)$ then $P_{v,R_s}(pt) = 0$. Otherwise, $P_{v,R_s(v)}(pt)$ depends on v's probability distribution function for point coverage, $f_v(r)$, that is specified by v's manufacturer. $f_v(r)$ is a continuous distribution function that decreases as the distance from v increases. To compute whether or not pt is covered by node v, we consider a small distance δ around pt and slightly modify the definition of $P_{v,R_s(v)}(pt)$ to be:

$$\begin{aligned} P_{v,R_s(v)}(pt) &= \int_{D_0-\delta/2}^{D_0+\delta/2} f_v(r)dr, \qquad D_0 \leq R_s(v) \\ &= 0, \qquad \text{Otherwise,} \end{aligned} \tag{2}$$

where δ is the distance between two adjacent points in the field. A point pt is covered by a set of nodes S if the probability of coverage exceeds a threshold γ ($0 \leq \gamma \leq 1$), i.e.,

$$(P_{cov}(pt) \stackrel{\text{def}}{=} [1 - \prod_{j \in S}(1 - P_{j,R_s(j)}(pt))]) \geq \gamma, \tag{3}$$

where $R_s(j)$ is the current sensing range for node $j \in S$ and $P_{j,R_s(j)}(pt)$ is computed by Equation 2. To generalize, a node v decides that pt is covered if the probability that pt is covered by v or any neighbor j of v exceeds γ, i.e.,

$$(P_{cov}(pt) \stackrel{\text{def}}{=} [1-(1-P_{v,R_s(v)}(pt)) \prod_{j \in Nbrs(v)} (1-P_{j,R_s(j)}(pt))]) \geq \gamma, \tag{4}$$

where $Nbrs(v)$ is the set of v's neighbors.

The above method models probabilistic sensor coverage in which the area covered by a sensor is not known a priori. Another probabilistic model was also proposed in [14] in

which the area and perimeter of the sensing region of a sensor are arbitrary (i.e., non-uniform) but known. We can map this approach to our problem as follows. Assume that the sensing region F_0 around node v is convex and has an area A_0 and perimeter L_0. Let A_1 and L_1 be the area and perimeter, respectively, of another convex sensing region F_1 of a neighbor of v that is placed randomly in the field. Finally, assume that F_1 intersects F_0. The probability that a randomly selected point pt in A_0 is covered by F_1 (i.e., pt is covered by v's neighbor) is given by:

$$P(pt, F_1) = \frac{2\pi A_1}{2\pi(A_0 + A_1) + L_0 L_1} \quad (5)$$

We can thus plug Eq. 5 in both Eq. 3 and Eq. 4 to compute the probability of coverage when multiple nodes are available. Note that every point within a given distance is covered by a probability computed by these equations *independently* from other points in the region.

B. Coverage and 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 backup covers within a duration of M slots.

Proposition 2: 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.

Proof. The following 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 II for demonstration). □

On the other hand, 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 stronger resilience. 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 an 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 an M-slots interval is $(1-p_f)^{n_c M}$. Consequently, the probability that the i^{th} backup cover is alive and active in any M-slots interval is $P_a^{(i)}(M)$, where $P_a^{(i)}(M) = (1/i)(1-p_f)^{n_c M}$ (the $1/i$ term is removed in the conservative approach). Without loss of generality, assume that the backup covers are node-disjoint. The probability that any backup cover is alive and active during an M-slots interval, $P_a(M)$, can be computed as:

$$P_a(M) = 1 - \prod_{i=1}^{S} [1 - P_a^{(i)}(M)]. \quad (6)$$

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

$$P_c(M) = 1 - P_f(M) \times (1 - P_a(M)). \quad (7)$$

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.

If p_f cannot be estimated, it is 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, ROC provides dynamic recovery, where every node can control the number of its backup covers and their probing frequency.

C. Overhead

1) Communication 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 message overhead. 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 network operation. We assume that energy consumption is directly proportional to the duration in which a node is active. Thus, 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, $n_a = O(kN_c)$. In ROC opportunistic scheduling approach, n_a depends on several parameters, namely, M, S, and n_c. Thus, it can be computed as follows:

$$n_a = N_c(1 + \sum_{i=1}^{S} \frac{n_c}{iM}). \quad (8)$$

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. This provides a very conservative estimate of energy consumption because a node typically wakes up for only a fraction of t_s if active probing is employed.

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

$$\frac{n_c \times \ln S}{M} < k \tag{9}$$

Assuming $M = 20$, $S = 2$, and $n_c = 4$, the left-hand side of (9) is less than 2. This means that ROC is more energy-efficient than proactively providing 3-coverage in the field. In conservative scheduling, the $\ln S$ term in (9) should be replaced by S, which is still more efficient than k-coverage if $(n_c \times S)/M < k$. 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 vise 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.

D. Network Connectivity

Based on the results in [21], [27], [26], we highlight the sufficient conditions for network connectivity with different relationships between sensing and communication ranges. We extend the results in [21], [27] to consider the cases where the sensing and communication ranges are variable or non-convex. In particular, assume that a node v has a set of active neighbors $Nbrs(v)$ that form a 1-cover of v, then:

- If every $u \in Nbrs(v)$ has circular sensing range R_s and circular communication range R_c, then $Nbrs(v)$ form a connected graph if $R_c \geq 2R_s$.
- If every $u \in Nbrs(v)$ has convex sensing range with minimum Rs_{min} and maximum Rs_{max}, and convex communication range with minimum Rc_{min} and maximum Rc_{max}, then $Nbrs(v)$ form a connected graph if $Rc_{min} \geq 2Rs_{max}$.
- Assume that every $u \in Nbrs(v)$ has convex communication range with minimum $Rc_{min}(u)$ and maximum $Rc_{max}(u)$. Also assume that every $u \in Nbrs(v)$ has convex sensing range with minimum $Rs_{min}(u)$ and maximum $Rs_{max}(u)$. $Nbrs(v)$ form a connected graph if $\forall u \in Nbrs(v)$, $Rc_{min}(u) \geq Rs_{max}(v) + Rs_{max}(u)$.
- Assume that any of the following conditions is true: (1) $\exists u \in Nbrs(v)$ that has circular sensing range R_s and communication range R_c, and $R_c < 2R_s$; (2) $\exists u \in Nbrs(v)$ that has convex sensing range with maximum Rs_{max} and convex communication range with minimum Rc_{min}, and $Rc_{min} < 2Rs_{max}$; (3) $\exists u \in Nbrs(v)$ that has convex sensing range with maximum $Rs_{max}(u)$ and convex communication range with minimum $Rc_{min}(u)$, such that $Rc_{min}(u) < Rs_{max}(v) + Rs_{max}(u)$; (4) $\exists u \in Nbrs(v)$ that has non-convex communication range. Under any such condition, v should explicitly test whether or not $Nbrs(v)$ form a connected graph. This can be done by collecting the neighbor tables and checking whether or not there is a path between every pair of neighbors. If not, then v should *not* go to sleep.

E. Limitations of ROC

ROC has a couple of limitations. First, if multiple nodes within close proximity fail simultaneously, recovery may cause more nodes than necessary to wake up, resulting in unneeded redundancy. Simultaneous failures also cause slowed recovery if non-failing nodes are not aware of neighbor failures and rely on such neighbors to heal the resulting holes. Second, unnecessary redundancy may occur due to link failures as a result of fluctuating link conditions and node mobility. Link failure is not a cause of redundancy, however, if it occurs due to obstacles between close nodes. Third, random node mobility can mislead ROC into unnecessary over-recovering, leading to unstable operation and significant overhead. It is better for ROC that the sensors are semi-stationary or move in swarms.

VI. Switching Between Covers in ROC

To select covers without the need for synchronized nodes, 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. The nodes in V_A 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. 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.

The nodes in V_A are selected using any distributed DCS protocol. A node $v \in V_A$ becomes the *proxy* of a 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 (assuming real-valued distances) or the neighbor of u that has the lowest/highest ID (assuming unique identifiers). Node v is thus capable of determining the role that u will have in the next active cover. Proxy selection resembles populating clusters in a clustered network. The difference here is that v, like a cluster head, autonomously decides to be a proxy for some of its neighbors based on a pre-specified criterion. Thus, every non-active node has only one proxy 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. 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. [21], [27], [25], [26]). 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 serves as 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 techniques in [25], [26] require that nodes with the highest remaining battery among their neighbors to decide first. 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 new states in the next V_A.

2) Autonomous DCS Algorithms (A-DCS): In A-DCS algorithms, a node decides whether or not to join V_A independently from other nodes. For example, in [19], every node sets a timer to a random duration T. After T expires, the node 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 the members of the current V_A since they require no neighbor coordination, unlike the C-DCS algorithms. They are also more flexible in failure recovery, as will be discussed below.

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 the 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. 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 the 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. If some nodes in the new V_A fail, their neighbors will find their sensing regions uncovered will be involved in a "healing" process, rather than a complete cover selection process.

2) Negligible overhead: Cover selection is carried out only by active nodes in V_A. Thus, OCU significantly reduces the overhead of cover selection. Moreover, the messages that are required for cover selection can be piggy-backed on the periodic routing update packets (if any).

3) Energy efficiency: Cover selection does not require extra energy cost by the nodes in V_A since they are already active. In addition, the transition between covers only requires that every node checks that its neighbors in V_A cover its sensing range before going to sleep. Therefore, the average active period per node at the start of a new OCU cover is significantly smaller than that required for cover selection.

VII. Performance Evaluation

In this section, we evaluate the ROC framework and contrast it to protocols constructing k-covers, where $k \geq 1$. We select representatives of the two categories of coverage algorithms that have been described in Section II, namely the DCS and SCS approaches.

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 duration and listens to its neighbor messages. When v's timer expires, it checks whether its entire sensing region is k-covered by active neighbors. If not, v joins V_A. After cover selection, k-cov lets every $v \notin V_A$ go to sleep. k-cov allows sensor collaboration for better coverage and failure recovery at the expense of enforcing network-wide requirements that may cause shorter lifetime. ROC uses k-cov's underlying cover selection 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. Note that we do not let non-backup nodes wake up during the operation of V_A. This is done to examine the effect of the S backup covers on coverage quality, without support from non-backup nodes.

We compare ROC and k-cov to RIS [13] 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 [13] to compute the necessary wake up probability for a fixed number of nodes. RIS determines the wake up probability prior to deployment, which makes a

sensor quite autonomous at the expense of slow (or none) reaction to failures.

Our metrics are fault tolerance and energy consumption. "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 for every approach. The following parameters are varied: (1) the sensing radius R_s, which determines the actual node density in the network, (2) the tolerance interval M, and (3) the failure probability p_f. We use opportunistic scheduling for ROC in all the experiments in which p_f can be estimated. We compare ROC to k-cov and RIS under random failures, in addition to location-based failures.

We assume that N=500 nodes are randomly deployed in a 50×50 meters^2 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. An active node consumes 24×10^{-3} Watts, while a sleeping node consumes 3×10^{-6} Watts. The active interval is typically dominated by idle-listening and thus the above value includes the energy consumed in communication. This model resembles that of MICA II sensor motes [5]. To study the effect of 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. Since the communication overhead in ROC is not significant and the sensor reporting rate is also low, packet collisions are not likely. However, we show experiments where packet losses occur and cause nodes to be unaware of the presence of their neighbors. We also consider cases where the required coverage can be partial. Every result is the average of 10 experiments of different randomly generated configurations. For every configuration, we preset a *failure schedule* for selected 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 first study their operation under no node 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 10 illustrates 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 outperform the two protocols.

We also quantify energy consumption when partial field coverage is sufficient. In this experiment, we simply change the objective in the underlying cover selection algorithm of both k-cov and ROC to stop adding nodes to a cover when the target field coverage is satisfied. Since it is not know how to extend RIS for partial coverage, we excluded it to maintain a fair comparison. Figure 11 (a) and (b) illustrate that ROC outperforms k-cov in both energy efficiency and fault tolerance. This is because k-cov tends to be over-conservative, while ROC tries to minimally satisfy the coverage objective at every node.

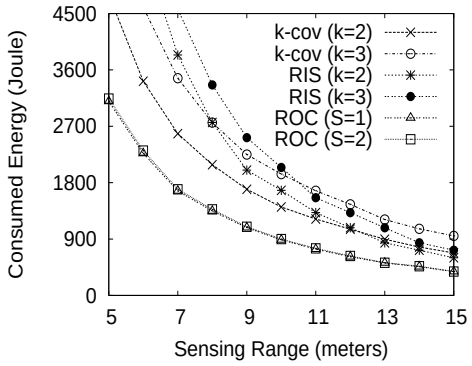
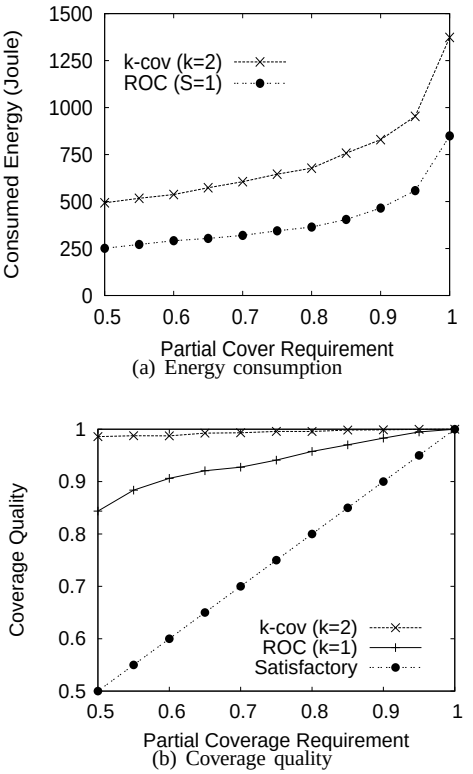


Fig. 10. Energy consumption under no failures.



(a) Energy consumption

(b) Coverage quality

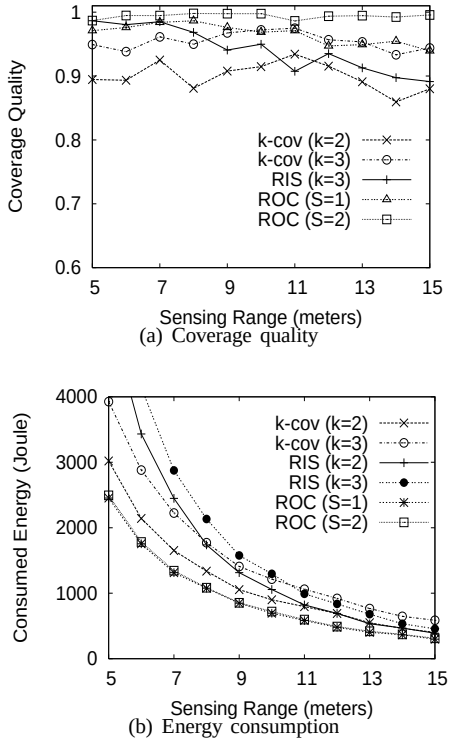
Fig. 11. Performance under no failures and partial coverage requirement.

B. Random Failures

In this section, we study the performance 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, which is controlled by R_s. We set M=50 slots and p_f=0.001. Figure 12(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 $k = 3$, as depicted in Figure 12(b). 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.



(a) Coverage quality

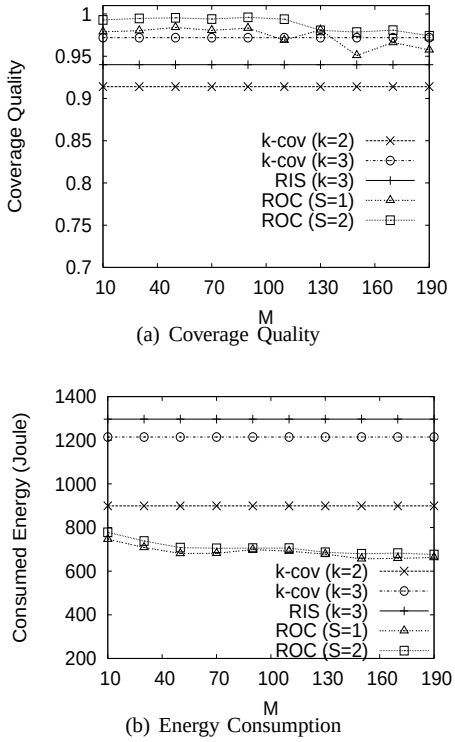
(b) Energy consumption

Fig. 12. Performance under random failures and variable sensing range.

We also study the effect of M on ROC reliability. Figure 13(a) shows the coverage quality when M varies from 1% to 20% of T. In this experiment, we used $p_f = 0.0001$ 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 13(b) 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. However, this overhead is not significant compared to the improvement in coverage quality. For critical applications, we recommend that the application use small values of M (compared to T).

Now, we study the impact of failure probability (p_f). We set R_s=10 meters, M=50 slots, k= 2 and 3, and S=1 and 2. Figure 14 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 has a similar trend to the one above.

Finally, we extend the above experiment and assume that $p_f = 10^{-4}$ and that broadcast packets can be lost, resulting in nodes being unaware of their neighbors. Other types of packets, such as those assigning backup covers and schedules,



(a) Coverage Quality

(b) Energy Consumption

Fig. 13. Performance under random failures and variable M.

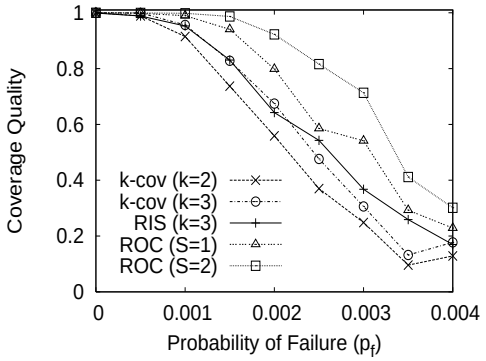
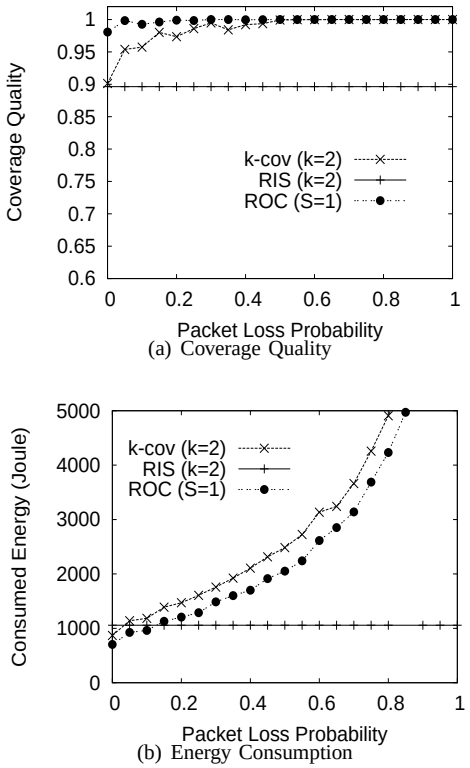


Fig. 14. Coverage quality vs. p_f (random failures).

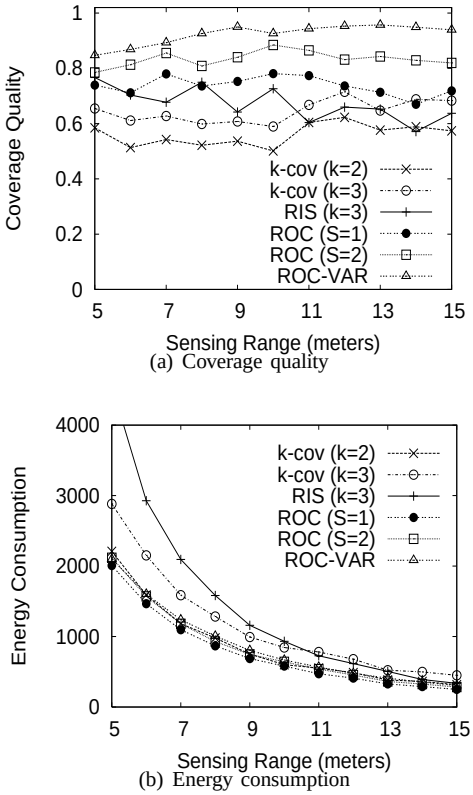
are typically transmitted reliably in ROC and k-cov (i.e., using MAC-level acknowledgments and re-transmissions) and therefore we do not assume their loss. Packet losses has no impact on RIS in terms of behavior (although it impacts network connectivity). As expected, results (Figure 15 (a) and (b)) show that the quality of coverage remains consistent with both ROC and k-cov, which outperform RIS. However, to keep up with full coverage, consumed energy increases significantly and ROC degrades to k-cov in this scenario.

Note, however, that it is quite unrealistic to assume that sensors executing ROC are unaware of the presence of their neighbors throughout the entire mission. This is because active nodes typically broadcast heartbeat messages periodically.



(a) Coverage Quality
(b) Energy Consumption

Fig. 15. Performance under random failures and packet losses.



(a) Coverage quality
(b) Energy consumption

Fig. 16. Performance under clustered failures and variable sensing range.

C. Clustered Failures

We study the performance of ROC under non-uniform probability of node failure. We assume that failure is location-dependent, where nodes closer to the bottom leftmost coordinate (0,0) have the less susceptibility to failure than the top rightmost coordinate (as in Figure 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 the detected failure rate, which corresponds to the node's distance from the origin. Far-distance nodes from (0,0) use larger S than closer ones.

Results in Figure 16(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 16(b) shows that although ROC activates more nodes for recovery, its energy consumption is still significantly less than k-cov and RIS at $k = 3$. As expected, ROC-VAR shows the best fault tolerance, especially at smaller ranges.

We also plot a snapshot of the field coverage as time progresses in Figure 17. 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 attributed to its reliance on an expected failure pattern, which is not the case under this scenario.

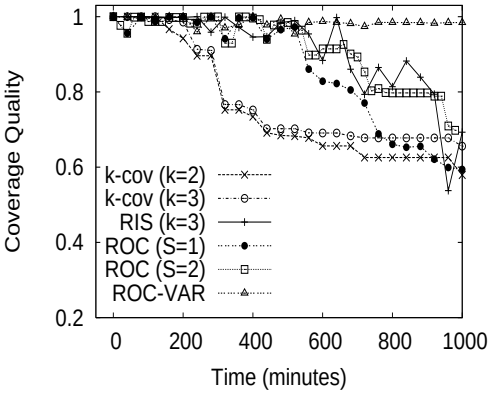


Fig. 17. Snapshot of coverage quality as time progresses.

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 can employ to both area coverage and barrier coverage techniques. It allows different degrees of redundancy 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. Simulations show that ROC is more efficient than network-wide k-coverage and probabilistic activation mechanisms, especially when the failure pattern is non-uniform. In addition, significant energy saving benefits are achieved. For

future work, we plan to study the performance of ROC under application-imposed constraints, such as response time.

REFERENCES

[1] "Warfighter information network-tactical (WIN-T)," http://www.globalsecurity.org/military/systems/ground/win-t.htm.

[2] P. Balister, Z. Zheng, S. Kumar, and P. Sinha, "Trap coverage: Allowing coverage holes of bounded diameter in wireless sensor networks," in *Proc. of the IEEE INFOCOM Conf.*, Apr. 2009.

[3] 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.*, Mar. 2005.

[4] J. Carle and D. Simplot-Ryl, "Energy-efficient area monitoring for sensor networks," *IEEE Computer*, no. 2, pp. 40–46, Feb. 2004.

[5] Crossbow Technology Inc., http://www.xbow.com/, 2007.

[6] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, July 1998.

[7] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenovic, "Localized sensor area coverage with low communication overhead," in *The IEEE International Conf. on Pervasive Computer and Communications (PerCom)*, Mar. 2006.

[8] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *Proc. of the ACM MobiCom Conf.*, Sep. 2004.

[9] 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.

[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] 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.

[12] F. Koushanfar, M. Potkonjak, and A. S. Vincentelli, "Fault tolerance techniques for wireless ad hoc sensor networks," in *IEEE Sensors*, 2002.

[13] S. Kumar, T. H. Lai, and J. Balogh, "On k-coverage in a mostly sleeping sensor network," in *Proc. of the ACM MobiCom Conf.*, Sep. 2004.

[14] L. Lazos and R. Poovendran, "Stochastic coverage in heterogeneous sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 3, pp. 325–358, Aug. 2006.

[15] G. Lu, N. Sandagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *Proc. of the IEEE INFOCOM Conf.*, Mar. 2005.

[16] 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, Apr. 2001.

[17] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. of the ACM Conf. on Embedded Networked Sensor Systems (ACM SenSys)*, Nov. 2004.

[18] S. Slijepsevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *Proc. of the IEEE International Conf. on Communications (ICC)*, June 2001.

[19] 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*, Sep. 2002.

[20] G. Wang and G. Qiao, "Multi-round sensor deployment for guaranteed barrier coverage," in *Proc. of the IEEE INFOCOM Conf.*, Apr. 2009.

[21] 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, Aug. 2005.

[22] 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)*, Nov. 2003.

[23] 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.

[24] 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.

[25] O. Younis, M. Krunz, and S. Ramasubramanian, "Coverage without location information," in *Proc. of the IEEE International Conf. on Network Protocols (ICNP)*, October 2007.

[26] O. Younis, S. Ramasubramanian, and M. Krunz, "Operational range assignment in sensor and actor networks," *Ad Hoc and Sensor Wireless Networks Journal*, vol. 5, no. 1-2, pp. 69–100, 2008.

[27] 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, Jan. 2005.

Ossama Younis is a Senior Research Scientist in Applied Research, Telcordia Technologies, Inc. at Piscataway, NJ, USA since 2007. He is currently leading projects on automated network design and optimization. He has been an Assistant Research Scientist in computer engineering at the University of Arizona from July 2005 till June 2007. He received the Ph.D. degree in computer science from Purdue University, USA in 2005. He also received the B.S. and M.S. degrees in computer science from Alexandria University, Egypt, in 1995 and 1999, respectively. Dr. Younis has served on the Technical Program Committee for several international conferences, in addition to serving as the web chair of ICNP 2008. His research interests include wireless sensor network protocols and applications, and cognitive-radio networks. He is a member of the ACM.

Marwan Krunz is a professor of ECE at the University of Arizona and the co-director for Connection One, a joint NSF/state/industry IUCRC cooperative center that focuses on RF and wireless communication systems and networks. He received the Ph.D. degree in EE from Michigan State University in 1995. He joined the University of Arizona in January 1997, after a brief postdoctoral stint at the University of Maryland. He previously held visiting research positions at INRIA, HP Labs, University of Paris VI, and US West (now Qwest) Advanced Technologies. Currently, he is a visiting researcher at the University of Carlos III, Madrid, Spain, where he holds a Chair of Excellence ("Ctedra de Excelencia") position. Dr. Krunz's research interests lie in the fields of computer networking and wireless communications, with recent focus on cognitive radios and SDRs; distributed radio resource management in wireless networks; channel access and protocol design; MIMO and smart-antenna systems; UWB-based personal area networks; energy management and clustering in sensor networks; media streaming; and fault monitoring/detection in optical networks. He has published more than 160 journal articles and refereed conference papers, and is a co-inventor on three US patents. M. Krunz is a recipient of the NSF CAREER Award (1998). He currently serves on the editorial boards for the IEEE Transactions on Mobile Computing and the Computer Communications Journal. He previously served on the editorial board for the IEEE/ACM Transactions on Networking (2001-2008). He served as a TPC chair for various conferences, including INFOCOM 2004, SECON 2005, and WoWMoM 2006. He has served and continues to serve on the executive and technical program committees of many international conferences and on the panels of several NSF directorates.

Srinivasan Ramasubramanian is an associate professor in electrical and computer engineering at the University of Arizona. He received the B.E. (Hons.) degree in Electrical and Electronics Engineering from Birla Institute of Technology and Science (BITS), Pilani, India, in 1997, and the Ph.D. degree in Computer Engineering from Iowa State University, Ames, in 2002. He is a co-developer of the Hierarchical Modeling and Analysis Package (HIMAP), a reliability modeling and analysis tool, which is currently being used at Boeing, Honeywell, and several other companies and universities. His research interests include architectures and algorithms for optical and wireless networks, multipath routing, fault tolerance, system modeling, and performance analysis. He has served as the TPC Co-Chair of BROADNETS 2005, ICCCN 2008, and ICC 2010 conferences and LANMAN 2010 Workshop. He has served on the editorial board of the Springer Wireless Networks Journal from 2005 to 2009.