Distributed, Reliable Restoration Techniques using Wireless Sensor Devices

Yannis Drougas, Vana Kalogeraki*
Department of Computer Science and Engineering
University of California-Riverside, Riverside, CA 92521 USA
{drougas,vana}@cs.ucr.edu

Abstract

Wireless sensor networks are small, inexpensive and flexible computational platforms, that have found popular applications in various areas including environmental monitoring, health care and disaster recovery. One fundamental question is how to place the nodes in the network so that complete coverage of the monitored area is achieved. In this paper, we use techniques from discrepancy theory that accurately represent the uncovered area using just a few discrete points, to make sure that every point in the network is covered by at least k sensors, where k is calculated based on user reliability requirements. Our technique is fully distributed, deploying a low number of sensors, and minimizes the communication costs. Our experiments demonstrate that our technique is highly effective in achieving a reliable restoration of a given sensor network area.

1 Introduction

Embedded sensor devices have created tremendous opportunities for a wide variety of application settings. Large-scale wireless sensor network deployments have emerged in environmental and habitat monitoring, agriculture, health care, homeland security and disaster recovery missions [16, 18, 22]. Different from general computer systems, the deployment and management of sensor networks pose significant challenges, mainly due to the uncertain nature of the deployment process as well as the limitations of the sensors themselves. Wireless sensor devices are either placed manually at predetermined locations, or (in some environments where human intervention is not possible) the nodes have to be deployed randomly and they will remain unattended for extended time periods. Once deployed, sensors are prone to failures due to manufacturing defects, environmental conditions (such as fires) or battery depletion. In such circumstances, the data (e.g., sensors' reports) may become stale or get lost. Failed or damaged devices can introduce inconsistencies in determining the criticality of a situation and thus must be replaced to repair the network.

Failures in the sensor network can be addressed by deploying sensors in large numbers (to increase redundancy); and the network can therefore survive a few node failures. However, the computation, communication and deployment costs increase with the number of sensors. Furthermore, these assumptions only partially reflect the nature of real-world sensor network environments. In practice, failures are correlated (*i.e.,* geographically). Correlated failures can lead to reduced system reliability. Given the dynamic environment, the limited resource capabilities and the unreliable nature of the sensor devices, we are interested in developing efficient and practical *restoration* mechanisms that can be implemented in-network to identify repairs and restore a sensor network to ensure reliable coverage. The fundamental question we want to answer in this paper is "*Given an area to be monitored, and an initial set of embedded sensor devices, how can we determine the number of devices required to restore the network to ensure that each point of the area is covered with at least k sensors?*"

Previous research [9, 17, 24] has looked at the problem of (k=1)-coverage, where the question is how well a number of sensors fully cover an area. The problem of k-coverage is fundamentally different because we need to achieve the combined goal of k-coverage and deploying the minimum number of sensors. By requiring that each point of the area must be covered by at least k sensors rather than a single sensor node, we provide fault tolerance and also prolong the lifetime of the network. Recent solutions [23, 24, 25] have looked at the problem of configuring an already deployed network or propose complex placement methods that are computationally expensive to run on the sensors. Sensors are resource limited; thus, it is essential to minimize the computation overhead. Our objective is to implement a distributed, low-complexity, in-network solution that can be run in the distributed sensor network to compute the best possible placement of the nodes. We assume that new sensors can be deployed to the proposed locations by a human

*This research was supported by NSF Awards 0330481 and 0627191.

or a mobile robot. Our algorithm can be implemented on such mobile robots or on the sensor devices. We provide a fast and efficient way of estimating a non-covered region and determining the best locations for the new sensor nodes. Restoring k-coverage is desirable in many practical applications:

1. As an example, consider an *environmental monitoring application* for monitoring wild-fires. Millions of acres of land areas are destroyed due to forest fires every year. If temperature-sensing nodes could be deployed to fully cover these regions, early warnings from sensors can help preventing such infernos. Reliable restoration is important in these settings to identify and repair faulty sensors, and to filter spurious reports.

2. Another example is the case of *intruder detection*. The detection of an intruder in a surveillance sensor network often requires that the intruder should be detected by more than one sensor devices. The ability of the network to detect the intruder and the accuracy of the detection increases with the number of nodes monitoring the area. In this application, restoring k-coverage is essential in order to increase precision and accurately determine the exact position, speed and direction of the intruder. In [4] it is shown that such k-coverage also improves the accuracy of such methods.

3. k-coverage also increases the lifetime of the network. When k nodes are covering a point, we have the option of putting some of them to sleep or balance the workload among all k nodes. Thus, k-coverage leads to significant energy savings and increases the lifetime for the network.

Our Contribution: In this paper we propose DECOR (DEpendable COverage Restoration), a method to restore k-coverage in sensor networks. More specifically, given a user reliability requirement k for the degree of coverage, our goal is to find the minimum number of sensor devices and their location to restore k-coverage of an area partially monitored by a sensor network, so that all the points in the entire area are covered by k sensor nodes.

We propose a distributed, low-complexity, in-network mechanism that first determines uncovered regions in the sensor network field and then proposes the deployment of nodes to completely cover the area. It consists of the following novel components: (a) an efficient and accurate method for representing an uncovered sensor area using techniques from discrepancy theory, and (b) a distributed mechanism for identifying a small number of nodes required to cover the sensor area and their locations. By representing the uncovered area as a set of points, we can use efficient and simple algorithms for finding small sets of sensors to cover the uncovered areas. Our mechanism is entirely distributed and works by partitioning the sensor network into cells and run our algorithms locally at each cell, solving a disk covering problem. The goal is to cover k times each set of points on the plane by a set of disks. This problem is known to be NP-complete for $k = 1$ [15]. However, there exist various approximate solutions that run in polynomial time and have a bounded error ratio [3]. DECOR uses such an approximate method, to achieve k-coverage of the entire region. We illustrate that our approximation method can be efficiently implemented in a sensor network, while the running time of the algorithm is polynomial. The use of the above technique is also the reason why our method can be used both to initially deploy a sensor network and to resume coverage of partially covered areas that emerge as a result of node failures.

2 System Model

We consider a set of n embedded sensor devices deployed in a geographic area A. Examples of such devices are motes. We assume that sensors are static and homogeneous. Each sensor s_i has a *sensing radius* r_s and a *communication radius* r_c (shown in Figure 1). The sensing radius determines the coverage radius of the sensor. That is, sensor s_i can cover any point located within a disc area of radius r_s centered at sensor s_i. The communication radius r_c of a node determines the set of nodes reachable from s_i, called 1-hop neighbors of s_i. In a heterogeneous network deployment, the sensing and coverage radii of the sensors may vary, depending on the type of the sensors and on the deployment conditions. Our solution is designed to work under such a setting, since the only assumption we make is that the sensing radius is smaller than or equal to the communication radius, ($r_s \leq r_c$).

There may be different degrees of coverage of a particular point in a sensing field depending on the user reliability requirements. A point p of area A is said to be *covered* by a sensor node s_i if and only if p's distance from s_i is smaller than or equal to r_s. A point p is said to be *k-covered* if and only if p is *covered* by at least k sensors of the network. Alternatively, one can say that p is k-covered if it lies within the sensing radii of at least k sensor nodes. The area A is said to be k-covered by a network of sensors if every point of A is k-covered. Area coverage does not necessarily imply network connectivity. It has been shown that a necessary and sufficient condition to guarantee network connectivity when full coverage is achieved, is $r_c \geq 2 \cdot r_s$ [23, 24, 20]. Although our solution does not rely on such an assumption to guarantee coverage, if this condition is met, then our techniques also guarantees k-connectivity. That means, the network remains connected, even if $k-1$ sensors fail. This is a simple corollary of the k-coverage property.

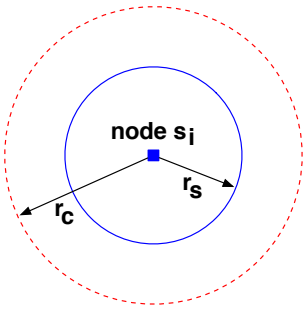

Figure 1. Sensing and communication radius.

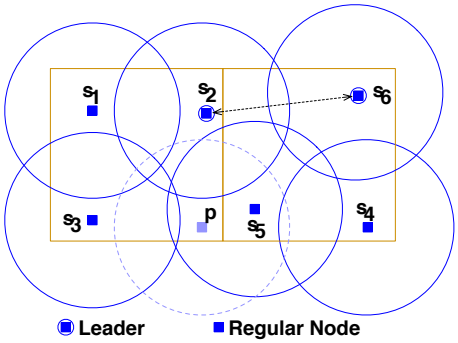

Figure 2. The grid-based model for $k = 1$.

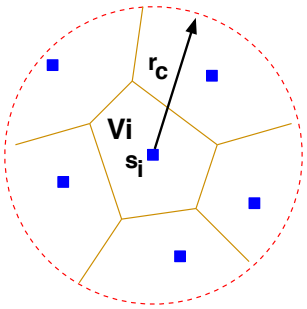

Figure 3. The local Voronoi cell of sensor s_i.

Let us assume an area A that needs to be k-covered. There can be two cases: either there is no current coverage of the area, or that the area is already partially covered by an already deployed sensor network. The latter case can occur because sensor nodes are inherent to failures; failures of some nodes can lead to loss of coverage of a region of the area. Failures are also the main reason that prevents us from using the simple solution of solving the coverage problem for $k = 1$ and then placing $k > 1$ nodes at each position indicated by the given solution: Besides the fact that placing multiple nodes on the same position is sometimes impossible due to practical reasons, nodes are more likely to suffer failures if a disaster happens in that specific area. In addition, such a solution would not be efficient in the case where we need to restore the coverage of a partially covered region. Next, we describe the node failure model we consider.

2.1 Failure Model

We focus on two types of failures: random node failures and geographical area failures. A number of conditions can cause sensor node failures: hardware failures, battery depletion, environmental conditions (such as fires, collapsed buildings, animal movement), or malicious activity. Although sensors are also susceptible to packet loss and link failures, monitoring each point with k sensor devices significantly decreases the probability of loosing critical data.

Random Node Failures: The nodes are failure-prone. We assume that multiple sensors can fail independently and concurrently and that all sensor nodes have the same failure probability q. Thus, the probability that a point will be covered by at least one sensor device, is computed as: $1 - q^k$, where k is the user reliability requirement.

Area Failures: During the lifetime of the network, the nodes covering an area may fail altogether. A reason for this can be a natural disaster (like an earthquake). In such a case, the level of coverage of the region where the disaster occurs is expected to dramatically drop, even to the point of no coverage. The degree of coverage drop and also the affected area depends on the size of the area where the failure occurred, the level of coverage k and also other factors such as the density of the nodes.

3 DECOR

In this section we describe DECOR, a DEpendable COverage Restoration approach, for solving the k-coverage restoration problem in sensor networks. Nodes running DECOR perform the following two steps: (1) Estimate the uncovered region, and (2) Identify a small number of sensors and their respective location in order to achieve k coverage. Finding the optimal k-coverage of the sensor area in a distributed manner, requires the exchange of large number of messages among the sensors, which makes it difficult to implement on resource constrained devices. Thus, we employ an approximate method to achieve k coverage of the entire region. Our approximation is shown to be efficient, while the running time of the algorithm is polynomial. The value of the parameter k can be tuned dynamically to achieve the desired level of coverage required by the user. We make an assumption that the sensor nodes are either GPS enabled or they are capable of finding out and reporting their respective positions to other nodes using an algorithm like the one proposed in [12].

3.1 Architecture

The given region is partitioned into local regions called cells. In each cell, sensors solve the k-coverage problem

locally. We consider two schemes for partitioning: *Grid based* (figure 2) and *Voronoi based* (figure 3).

In the *Grid based* scheme (Figure 2), the area is partitioned into fixed cells, with a single node (the leader) being responsible for each cell. We propose a hierarchical network organization in which a randomly elected leader will represent each local cell. We assume that each sensor knows its location, that communication is possible between any pair of sensors that lie in the same cell and that the leader at least knows the geographical boundaries of its cell. Each leader is responsible for identifying the uncovered regions in its cell, decide where to deploy new nodes and propagate its decision to the base station. We make the assumption that there is at least one sensor in each cell to act as a leader. This is without loss of generality, because if an entire cell is empty, we can use a regular positioning of sensors to cover it. A number of efficient, in-network algorithms have been proposed to solve the leader election problem [6, 11, 13]. The basic idea is to employ a random selection of leaders and a rotation mechanism for leadership selection so that the energy dissipation experienced by the leader in communicating with sensors gets spread across all nodes in the cell. We can use any of the aforementioned algorithms to elect leaders. The leader selection algorithm is performed periodically in the background.

In the *Voronoi based scheme* (Figure 3), each node s_i constructs it own cell. The cell of a node s_i is an approximation of its Voronoi cell. The Voronoi cell of s_i is a region, for which, the following holds:

Definition 1. *The local Voronoi cell V_i of a node s_i is an area, for each point p of which, the distance $d(p, s_i)$ from s_i to p is smaller than the distance $d(p, s_j)$ of p to any other node that has a direct link with s_i.*

An example of the local Voronoi cell V_i of a node s_i is shown in Figure 3. The node can determine its local Voronoi cell by considering location and coverage information exchanged with its neighbors. For each point p in its local Voronoi cell, a node *estimates* whether it is k-covered or not. Each time a new sensor node is placed, the placement may affect the size of the Voronoi cells of some neighboring nodes, in which case, the size of their cells will need to be updated to reflect the node addition.

3.2 Estimating the Uncovered Region

We tackle the problem of estimating an uncovered region by approximating the uncovered area using discrete points. Each node s_i maintains a set P_i of points that approximates the area of its cell. Instead of producing an actual description of the uncovered area, we produce an implicit description by finding a set of points that are not covered. This has the following advantages:

1. The algorithm is simple and can be executed locally. In order to determine the uncovered area, we need to find for each point if it is covered or not, which can be done easily locally.

2. The description of the uncovered area is also very simple: It consists of a set of discrete points.

However, the accuracy of the algorithm depends on how well the chosen set of points approximate the area. The problem of approximating a continuous measure such as the area with a discrete measure such as a set of points has been studied extensively in the area of Monte Carlo and Quasi-Monte Carlo integration. It has been shown that there exist sequences of 2-dimensional points that approximate the area much better than a random set of points of equal cardinality. Such point sets are characterized by low-discrepancy [21, 5]. For choosing a set of points to approximate the uncovered region, we propose to use the Halton and Hammersley generator which generates low discrepancy points for dimension d, of the order of $O(\frac{\log^d N}{N})$ and $O(\frac{\log^{d-1} N}{N})$ respectively, when a random set of points would have $O(\sqrt{\frac{\log \log N}{N}})$ discrepancy.

Detection of uncovered areas is done by eliminating the points from the point set in the node's cell, which lie in the coverage area of the previously placed sensor nodes so that the remaining points give us roughly a close estimation of all the uncovered regions in that cell. Whenever an uncovered region exists, nodes running DECOR deploy new nodes at the region's boundaries. This is because of the way DECOR works: Each node runs a greedy algorithm independently from other nodes, trying to place a new node in such a position, so that it will cover as many uncovered points as possible. As a result, the uncovered area decreases until full coverage is achieved (until all the points are covered). We would like to note that a node (either a leader node in the case of the Grid based scheme or a node in the Voronoi-based scheme) considers only the points in its cell when determining the uncovered region. The size of each cell is small. Thus, the run-time overhead can be kept low. DECOR is also able to detect uncovered areas resulting from node failures. Node failures can be detected using an algorithm like the ones proposed in [19].Neighboring nodes periodically exchange meta-information about their positions, with a period T_c. Once a node stops receiving such messages from one of its neighbors, this indicates that the neighbor has failed. The nodes do not need to be synchronized to ensure this functionality. Under the grid-based approach of DECOR, leaders are able to detect failure of leaders in neighboring cells (since leaders are able to communicate). In the case that the leader of a cell in the grid-based approach fails, one of the following will happen: (1) The remaining nodes in the cell will elect a new leader. (2)

If no nodes exist in the cell, the leader of a neighboring cell will place a new leader in the uncovered cell.

3.3 Identifying New Sensor Locations

DECOR uses a greedy method in order to determine the position of the new nodes to be inserted. In each iteration of the algorithm, a node s_i examines its assigned cell for uncovered points. The points that are examined are the points in P_i (the Halton / Hammersley points the cell is approximated with). If at least one uncovered point is found, the node inserts a new node s_j to one of the points in P_i. The point p where the new node s_j will be placed, is selected based on its *benefit*, $b_{j,p}$. The benefit $b_{j,p}$ of a sensor s_j at point p is a value that estimates the usefulness of placing a node in p. It is computed using the following formula:

$$b_{j,p} = \sum_{p':d(p',p)\leq r_s} \max\{(k - k_{p'}), 0\} \quad (1)$$

In the above formula, k represents the coverage requirement, while $k_{p'}$ represents the *current coverage of point* p'. In other words, $k_{p'}$ is the number of nodes that are currently covering point p'. By placing sensor s_j at point p, it will also cover all points p' that lie within distance r_s from p. Therefore, the difference $k - k_{p'}$ is a measure of the necessity to cover each point p'. The larger the coverage requirement k and the smaller the number of nodes $k_{p'}$ covering p', the more important it is to cover p' with a new node. On the other hand, the smaller the coverage requirement k and the larger the current coverage $k_{p'}$ of p', the smaller the necessity to cover p' with a new node. By taking into consideration $k - k_{p'}$, we try to cover first those points that are more urgent to be covered (since they are the least covered and the most possible to stop being covered on the event of a failure). The benefit $b_{i,p}$ of a node s_i placed at point p is therefore an indication of the value of placing s_i at p. The algorithm works incrementally. At each step, DECOR chooses to place a new node at the point with the maximum benefit. The algorithm continues until all the points of the area are covered by k nodes. This way we ensure that a minimal number of nodes will be used to k-cover the area. The DECOR deployment algorithm run by a node s_i is illustrated in Algorithm 1.

An advantage of our distributed approach is that we minimize the communication cost and the energy consumption, because we minimize the sensor to sensor communication. Under the grid based approach, a disadvantage is that a leader needs to have enough computational resources to compute the set of low discrepancy points for its entire cell and then run the coverage algorithm. By tuning the size of the cell, we can reduce the region that each leader is responsible for, so that fewer computational resources are required. Furthermore, using a leader rotation algorithm we can periodically assign the responsibility of a cell's coverage to a different node. Under the Voronoi based solution, the local Voronoi cell of each node s_i consists of a subset of the points within distance r_c from the node. The cell size of s_i decreases as more and more nodes are deployed, since points assigned to s_i are re-assigned to new nodes, that lie within distance r_c from s_i and with which, s_i can establish a communication link.

A node that runs the algorithm is either a leader in the grid-based approach or any regular node in the Voronoi approach.

Algorithm 1 The DECOR algorithm

Let P_i be the set of Halton points in the cell
  of node s_i with $k_p < k$
while there is a point $p \in P_i$
  select point $p' \in P_i$ such that benefit $b_{j,p'}$ is maximum
  place sensor s_j at point p'

An additional step is required in cases where we need to cover areas near the borders of the cells. A node may falsely detect that points in its cell are not covered, while these points may be covered by a sensor in a neighboring cell. DECOR addresses this problem by having nodes exchange information with their neighbors regarding placement of new nodes. In the grid based approach, a leader determines whether a new node is also covering part of the area of a neighboring cell. In such a case, before the insertion of the new node it informs the respective leader of the neighboring cell. This way, each leader is aware of points in its cell covered by nodes in neighboring cells. Note that the leader is still responsible for the coverage of these points. The location information about nodes in neighboring cells is only exploited by the leader in order to avoid over-coverage of parts of its cell. Under the Voronoi based approach, each node can by definition communicate with nodes whose distance is less than r_c. Since $r_s \leq r_c$, a node can accurately estimate the coverage of each of its points.

4 Experimental Evaluation

We performed a comprehensive set of experiments to evaluate DECOR. In our simulation, we deployed up to 200 sensor nodes on a 100×100 sensor network. The field was approximated with 2000 Halton points (an example of such a field is shown in Figure 4). We also experimented using a set of Hammersley points to approximate the field. The results were similar to the ones presented in this section and are omitted due to space limitations. The sensing radius of each node was $r_s = 4$ (the values of r_s and r_c are similar to the ones used in [25]). In the grid-based approach, we evaluated DECOR using two different types of cells: a small cell (5×5) and a big cell (10×10). Note that a

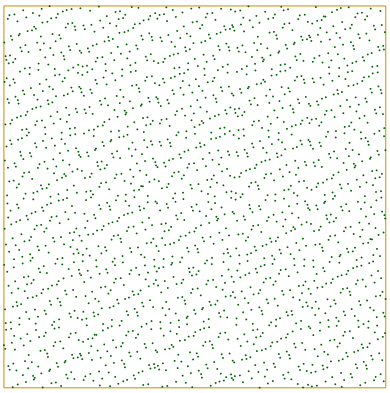

Figure 4. A field approximated with 2000 points.

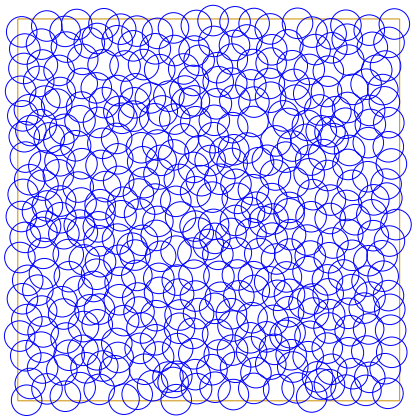

Figure 5. An example of the resulting DECOR deployment.

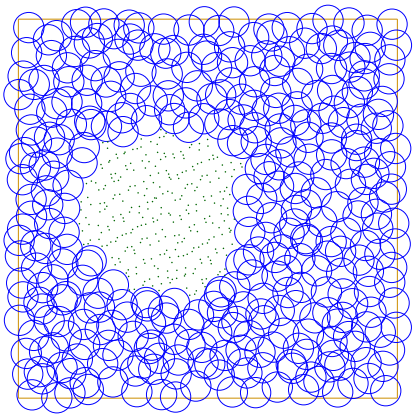

Figure 6. An uncovered area.

node with sensing radius $r_s = 4$ can almost entirely cover a 5×5 cell. This is not true for a 10×10 cell though. In the Voronoi approach, we chose two values for the communication radius: $r_c = 2 \cdot r_s = 8$ (small communication radius) and $r_c = 10 \cdot \sqrt{2} \approx 14$ (big communication radius). The large communication radius was selected in correspondence to the size of a cell under the grid-based based approach. Assuming that the cell size is 5×5, then, the maximum distance between two neighboring leaders is $10 \cdot \sqrt{2}$. Therefore, $r_c = 10 \cdot \sqrt{2}$ is the minimum required communication radius in order for the grid-based based approach to function without the need of any routing mechanism for the inter-leader communication. In all figures, the average of 5 runs, each one on a randomly generated field, are shown. Figure 5 shows an example of the resulting DECOR deployment. Figure 6 shows an example of an area failure which we used in our simulations. We evaluated our approach using the following metrics:

- The *total number of nodes* required to restore k-coverage of all the points in the area. Since Halton and Hammersley points accurately represent an area, this is actually the number of nodes required to cover 100% of the area k times.

- The number of nodes that are *redundant*. A node is considered to be redundant, if it does not contribute to the coverage of the area. By eliminating this node, we would still achieve k-coverage. Redundant nodes are identified at the end of the algorithm execution. The number of redundant nodes should be minimal.

- The coverage of the network achieved by our algorithm when failures occur. We have evaluated DECOR under both random node failures and area failures.

We have compared DECOR with a (1) *centralized greedy algorithm* that uses the same heuristic as DECOR to identify the locations of the new nodes but using a global view of the field. The centralized greedy algorithm is expected to result in a more efficient placement than DECOR. However, having global knowledge of the field is not possible in many cases. We also compared DECOR with a (2) *random placement algorithm* that places the nodes at random positions in the field until k coverage is achieved. DECOR requires no centralized authority and each node only needs minimal information about its neighborhood. We have evaluated DECOR under both Grid-based and Voronoi-based architectures. We have performed two sets of experiments. In the first set, we evaluated the deployment method of DECOR. In the second set of experiments we have evaluated how DECOR works under random sensor node and area failures.

4.1 Evaluation of Deployment Method

In the first experiment we evaluated DECOR under both Grid-based (small cell and big cell) and Voronoi based (small communication radius and big communication radius) architectures. In figures 7 and 8, our algorithm is evaluated in terms of the nodes required to achieve k coverage. Figure 7 shows the percentage of the points of the field that are k-covered for different number of sensors, for $k = 3$. Figure 8 shows the number of nodes required in order to achieve k-coverage of 100% of the monitored area for different values of k. In both cases, we can see that DECOR achieves k-coverage using a small number of nodes. DECOR tries to distribute nodes in the field as fairly as possible, first trying to cover the areas where coverage is too low. When there are not enough nodes, this strategy results in a lot of points being covered by at least one node,

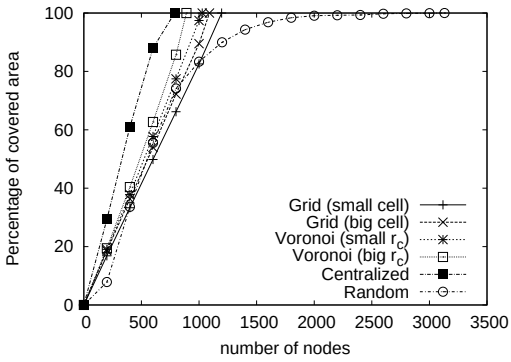

Figure 7. Coverage achieved with different number of sensors, for $k = 3$.

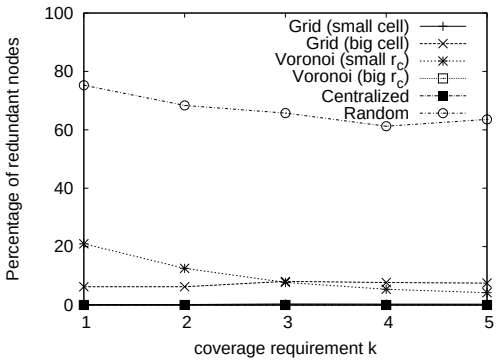

Figure 9. Percentage of redundant nodes vs. k.

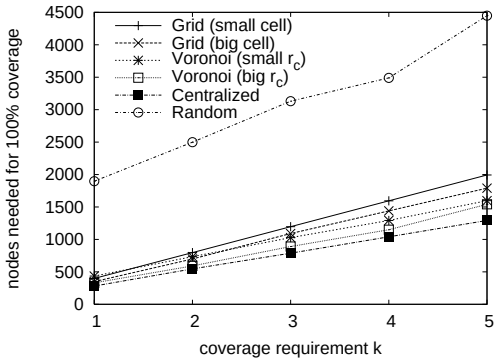

Figure 8. Number of nodes needed for k-coverage of the area vs. k.

ensuring the best possible coverage under the given conditions. On the other hand, when enough nodes are available, this strategy ensures that k-coverage will be reached for every point as fast as possible. As we can see in figures 7 and 8, the centralized greedy algorithm has better performance than any DECOR approach. This is expected, since DECOR is a distributed algorithm, that only requires local information on each node. However, as shown in figures 7 and 8, the performance of DECOR is similar to that of the centralized greedy algorithm, when it is appropriately configured. For example, for $k = 4$, the centralized approach is shown to achieve k-coverage of the entire field using 788 nodes. Under the Voronoi approach, DECOR can achieve the same coverage using as few as 891 nodes (about 13% more than the centralized algorithm, under the Voronoi approach). Under the grid-based approach with a 5×5 cell, the number of nodes required is 1196 nodes.

In the next experiment we measured the percentage of nodes that are redundant, shown in Figure 9. A redundant node is pure overhead: It is a node that does not cover any point that needs to be covered. Covering such a point with an additional node is not beneficial since the point is already k-covered. So, if the number of redundant nodes is high, a lot of resources are wasted. The figure shows that in the Voronoi based architecture, increasing the communication radius r_c results in substantial decrease in the number of redundant nodes. This is expected, since increasing r_c means that each node is informed for a larger area. Thus, more accurate data can be gathered for the coverage of the area. For the grid-based approach, increasing the cell size results into an increase of the number of redundant nodes. Although this seems counter-intuitive, it can be explained as the number of redundant nodes is analogous to the length of the sides of the cell. The random deployment is the most inefficient of all. It was shown to employ 1500 (when $k = 1$) to 3000 (when $k = 5$) redundant nodes. The centralized greedy solution on the other hand, resulted in no redundant nodes. This is expected, since the algorithm is centralized and employs global knowledge about the field. However, when configured appropriately, DECOR does not waste too many resources either. In the case of the Grid approach with a big cell and the Voronoi approach with a big communication radius, the nodes have enough information about the field, so that they place a few or no redundant nodes at all.

The message overhead of DECOR is shown in figure 10. For the Voronoi approach, the figure represents the number of messages sent by each node, since there is one node per cell. For the grid-based approach, the number of messages per cell is the number of messages sent by a leader of the cell. The number of messages sent by a node is an indication about the energy dissipation of a node. Under the grid-based approach, a leader is responsible for informing all its neighbors about any node that is placed in its region. Thus, the bigger the cell size, the more the messages that need to be sent by a leader. Similarly, under the Voronoi approach, the number of messages needed to be sent by a

node upon placement, is analogous to the communication radius r_c. In order to balance the load among many nodes in the cell (in the grid based approach), we assume the usage of a leader rotation algorithm, under which, every node in each cell periodically serves as the leader of the cell. In that case, the responsibility of sending update messages is shared among all the nodes in the cell. The average number of messages sent per node when a leader rotation algorithm is employed, was about 4 messages per node when the cell size was small and 2 messages per node when the cell size was big. Also, the number of messages was constant, independently of the coverage requirement k. This is expected, since the total number of nodes employed in the grid-based approach increases proportionally with k. Thus, the burden of message transmissions is shared between more nodes.

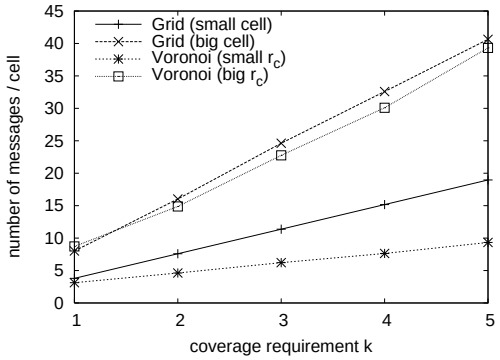

Figure 10. Message overhead of DECOR.

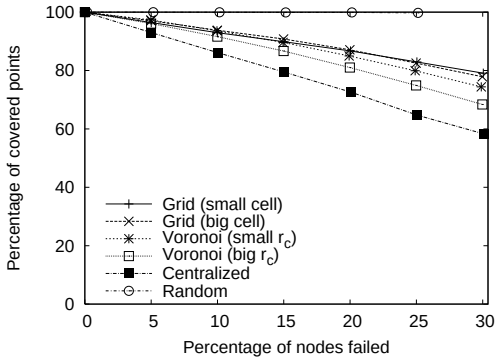

Figure 11. 3-coverage under random failures.

4.2 Evaluation under Failures

A benefit of k-coverage is that the sensor network can withstand node failures and keep the area monitored in the event of such failures. In this section we present our experimental results under nodes failures, that occur after the network is fully deployed. Unless otherwise stated, the node failures in this section are assumed to occur uniformly across the nodes of the network.

In the first experiment we observe the performance of the deployment algorithms for $k = 3$, when a randomly selected subset of up to 30% of the deployed nodes fail (Figure 11). A set of nodes deployed under the grid-based approach is shown to tolerate more failures than a set that was deployed under the Voronoi approach. Overall, DECOR adds the redundancy that is needed in order to provide better fault tolerance than centralized greedy algorithm. The random deployment algorithm is shown to tolerate more node failures. However, it uses about 4 times more nodes than any of the other methods and results in 10 to 20 times more redundant nodes. So, fault tolerance is too expensive under the random deployment. In Figure 12, the maximum percentage of random failures that can be tolerated in order to preserve coverage of at least 90% of the network, is presented. It is shown that, depending on k, DECOR can withstand failures of up to 75% of the deployed nodes and still cover 90% or more of the area with one or more nodes. Again, a larger value for k makes significant difference. In fact, 1-coverage of 90% can be achieved for $k \geq 2$, even when 30% of the nodes fail, while for $k \geq 3$, the performance of DECOR is equal to or better than the performance of the random deployment, despite the large number of redundant nodes employed by the latter.

Next, we performed experiments that feature the failure of the nodes in an entire region of the monitored area. We assume that a sudden event, like natural disaster destroyed all the nodes in a region covered by a disc of radius 24. This means that the disaster affected about 17% of the area. An example of such a phenomenon for $k = 1$ is shown in Figure 6. As a result, the affected area remains uncovered. Figure 13 shows to what extend is coverage maintained throughout the area. As expected, the percentage of k-covered points is the same for all deployment algorithms. In such a case, what matters is the quick restoration of coverage. To ensure detection of failures, nodes under DECOR exchange messages periodically. Figure 14 shows the number of nodes needed in order to recover coverage of the disaster area. The random placement needed from 1500 to 3000 nodes, being most inefficient. DECOR is shown to need 25% to 50% more nodes than the centralized algorithm. For $k = 5$, the centralized algorithm uses about 250 nodes to cover the uncovered area. The grid-based approach uses about 300 (small cell) and 270 (big cell) nodes. The Voronoi version of DECOR demonstrates better performance, using 270 (small r_c) and 250 (big r_c) nodes. In addition to that, one should consider that under the grid-based approach, extra overhead is incurred due to the fact that the leaders of the cells in the failure area have failed. So, the fraction of the cost of the grid-based approach to the cost of the Voronoi based approach in such a case is high.

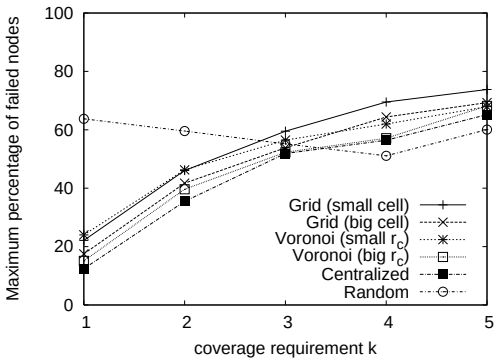

Figure 12. Maximum allowed failures for 1-coverage of 90% of the area.

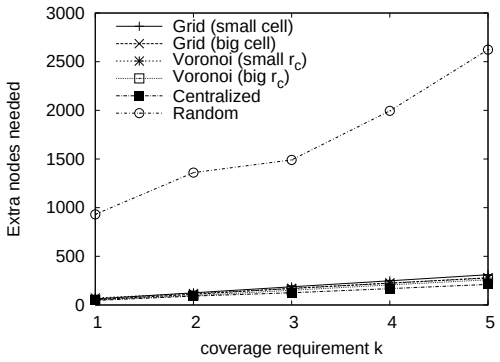

Figure 14. Number of nodes required to recover coverage of a failure area.

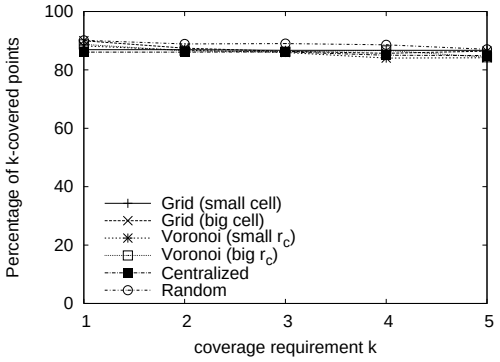

Figure 13. k-covered points after an area failure.

5 Related Work

Recent research has studied the problems of 1-coverage and k-coverage in wireless sensor networks. The authors in [8] formulate this problem as a decision problem and determine whether every point in the service area of the sensor network is covered by at least k pre-defined sensors. In [17], an algorithm is proposed for $k = 1$ coverage in which they use a centralized control server and nodes are connected using a gateway. Different from our work, the authors have assumed a simpler problem, the problem of efficient coverage of an area with base stations. Typically in a sensor network, the number of sensor nodes is significantly higher than the number of base stations. Also a base station has a broader coverage capacity compared to tiny sensor devices.

The coverage problem has also been discussed in [14]. The authors solve the problem of best-coverage path between any pair of sensor nodes using Delaunay triangulation and the Voronoi diagram. They assume a centralized control server, where nodes are connected using a gateway. Their approach may not be feasible for a large network where nodes are scattered over multiple hops. In [10], the authors propose a mechanism to determine the appropriate number of sensors to deploy that achieves k coverage of protected regions and in addition improves the lifetime of individual sensors. They consider three kinds of deployments for a sensor network on a unit square - an $\sqrt{n} \times \sqrt{n}$ grid, random uniform sensor distribution(for all n points), and Poisson distribution(with density n). Different from our work, they do not propose any placement algorithm for the sensor nodes.

The authors of [1] consider the problem of k-connectivity on a sensor network. They present an algorithm that solves this problem with provable performance guarantees. The problem of maintaining k independent paths between any two nodes of a sensor network is also an important problem. In [25], the authors consider the combined problem of k-coverage and k-connectivity. The solution they propose involves the computation of Voronoi diagrams from independent sensor nodes. Their method does not require any central authority. However, important implementation aspects, such as the computation of the required local Voronoi cells are not discussed. Furthermore, their approach requires a great amount of information exchange among the nodes, like connectivity and coverage data. The solution is rather complex to be implemented on computationally limited sensors and requires an excessive amount of communication among the nodes of the system. Our solution is much more appropriate to be applied in a sensor network environment.

The OGDC method [24] maintains both coverage and connectivity in wireless sensor networks, for the special case of $k = 1$. OGDC assumes that the transmission range of a node is much larger than the sensing range. The authors of PEAS [23] consider recovery from unpredictable node failures in wireless sensor networks. PEAS recovers

failures by using a randomized algorithm to wakeup sleeping nodes. However, PEAS only considers coverage for $k = 1$ and does not propose any placement algorithm.

An algorithm to configure an already deployed sensor network is presented in [20], offering both coverage and connectivity. However, it cannot be applied for the deployment of new nodes, as our approach. The authors in [2] propose an algorithm that extends the network lifetime and maximizes coverage with a user-defined number of sensors. This work examines the inverse of the problem addressed by DECOR. Additionally, the algorithm configures an already existing sensor network, rather than proposing the placement of new nodes, as DECOR does. In [7], H-SEND, a system to monitor and re-configure a sensor network in order to repair errors is presented. H-SEND addresses node failures due to software errors. However, it does not attempt to deploy new nodes in the event of node failures.

6 Conclusions

In this paper we have studied the problem of restoring the coverage of a geographical region in sensor networks using a minimum number of nodes. The coverage problem has many practical applications in environmental monitoring, surveillance and disaster recovery. Our mechanism, using techniques from discrepancy theory, computes the number of extra sensor nodes required to completely cover the given region. Our approach is unique in that it can be applied in a distributed manner by dividing the region into various cells and applying the algorithm in each of the local cells. We have demonstrated through simulations that our technique is effective in achieving coverage restoration of a given area.

References

[1] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus. Deploying sensor networks with guaranteed capacity and fault tolerance. In *MOBIHOC*, pages 309–319, Urbana-Champaign, IL, 2005.

[2] W. Choi and S. K. Das. A novel framework for energy-conserving data gathering in wireless sensor networks. In *IEEE INFOCOM*, pages 1985–1996, Miami, FL, 2005.

[3] M. Franceschetti, M. Cook, and J. Bruck. A geometric theorem for approximate disk covering algorithms. Technical report, Caltech, 2001.

[4] D. L. Hall and J. Llina. *Handbook of Multisensor Data Fusion*. CRC Press, 2001.

[5] J. Hammersley and Handscomb. *D.C. Monte Carlo Methods*. Wiley, New York, 1964.

[6] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient Communication Protocols for Wireless Microsensor Networks. In *HICSS*, page 8020, Maui, HI, 2000.

[7] D. Herbert, Y.-H. Lu, S. Bagchi, and Z. Li. Detection and repair of software errors in hierarchical sensor networks. In *SUTC*, pages 403–410, 2006.

[8] C.-F. Huang and Y.-C. Tseng. The coverage problem in a wireless sensor network. In *WSNA*, pages 115–121, 2003.

[9] N. Kumar, D. Gunopulos, and V. Kalogeraki. Sensor network coverage restoration. In *DCOSS*, pages 409–409, Marina del Rey, CA, 2005.

[10] S. Kumar, T. H. Lai, and J. Balogh. On k-coverage in a mostly sleeping sensor network. In *MobiCom*, pages 144–158, Philadelphia, PA, 2004.

[11] J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao. Distributed group management for track initiation and maintenance in target localization applications. In *IPSN*, pages 113–128, Palo Alto, CA, 2003.

[12] N. Malhotra, M. Krasniewski, C. Yang, S. Bagchi, and W. Chappell. Location estimation in ad hoc networks with directional antennas. In *ICDCS*, pages 633–642, Columbus, OH, 2005.

[13] N. Malpani, J. L. Welch, and N. Vaidya. Leader election algorithms for mobile ad hoc networks. In *DIALM*, pages 96–103, Boston, MA, 2000.

[14] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *INFOCOM*, pages 1380–1387, Anchorage, AK, 2001.

[15] M. P. R. Fowler and S. Tanimoto. Optimal packing and covering in the plane are np complete. *Inf. Proc. Letters*, 12(3):133–137, 1981.

[16] C. Sadler, P. Zhang, M. Martonosi, and S. Lyon. Hardware design experiences in zebranet. In *SenSys*, pages 227–238, Baltimore, MD, 2004.

[17] S. Slijepcevic and M. Potkonjak. Power Efficient Organization of Wireless Sensor Networks. In *ICC*, pages 472–476, Helsinki, Finland, 2001.

[18] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *SenSys*, pages 214–226, Baltimore, MD, 2004.

[19] A. T. Tai, K. S. Tso, and W. H. Sanders. Cluster-based failure detection service for large-scale ad hoc wireless network applications. In *DSN*, pages 805–814, Florence, Italy, 2004.

[20] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. D. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *SenSys*, pages 28–39, Los Angeles, CA, 2003.

[21] H. Woźniakowski. Average case complexity of multivariate integration. *Bull. Amer. Math. Soc. (N.S.)*, pages 185–194, 1991.

[22] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *SenSys*, pages 13–24, Baltimore, MD, 2004.

[23] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *ICDCS*, pages 28–37, Providence, RI, 2003.

[24] H. Zhang and J. C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *Int. Journal of Wireless Ad Hoc and Sensor Networks*, 1(1-2):89–124, 2005.

[25] Z. Zhou, S. Das, and H. Gupta. Fault tolerant connected sensor cover with variable sensing and transmission ranges. In *SECON*, Santa Clara, CA, 2005.