Redundancy and Coverage Detection in Sensor Networks

Journal:	*Transactions on Sensor Networks*
Manuscript ID:	TOSN-2005-0072
Manuscript Type:	Paper
Date Submitted by the Author:	04-Nov-2005
Complete List of Authors:	Carbunar, Bogdan; Purdue University, Computer Science Grama, Ananth; Purdue University, Computer Science Vitek, Jan; Purdue University, Computer Science Carbunar, Octavian; Purdue University, Computer Science
Computing Classification Systems :	category{C.2.1}{Network Architecture and Design}{Distributed networks}, category{C.2.1}{Network Architecture and Design}{Network topology}, category{C.2.4}{Distributed Systems}{Distributed applications}
Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.	
final.tar.gz	

Redundancy and Coverage Detection in Sensor Networks

BOGDAN CĂRBUNAR, ANANTH GRAMA, JAN VITEK
Purdue University, West Lafayette, IN, 47906
and
OCTAVIAN CĂRBUNAR
IFIN-NIPNE, Magurele, Romania

We study the problem of detecting and eliminating redundancy in a sensor network with a view to improving energy efficiency, while preserving the network's coverage. We also examine the impact of redundancy elimination on the related problem of coverage-boundary detection. We reduce both problems to the computation of Voronoi diagrams, prove and achieve lower bounds on the solution of these problems, and present efficient distributed algorithms for computing and maintaining solutions in cases of sensor failures or insertion of new sensors. We prove the correctness and termination properties of our distributed algorithms, and analytically characterize the time complexity and traffic generated by our algorithms. Using detailed simulations, we also quantify the impact of system parameters such as sensor density, transmission range, and failure rates on network traffic.

Categories and Subject Descriptors: C.2.1 [**Network Architecture and Design**]: Distributed networks; C.2.1 [**Network Architecture and Design**]: Network topology; C.2.4 [**Distributed Systems**]: Distributed applications

General Terms: Algorithms, Design

Additional Key Words and Phrases: Sensor networks, coverage, energy efficiency, redundancy elimination, coverage boundary

1. INTRODUCTION

The need for distributed sensing and monitoring of remote or inaccessible areas, along with the integration of sensed information into a variety of physical processes provide overarching motivations for sensor networks. The emphasis on low-cost, dense sensing comes with significant constraints on battery power, communication bandwidth, and compute power. For this reason, considerable work has focused on efficient techniques for extending the network's lifetime. This paper focuses on the problem of detecting and eliminating redundant sensors without affecting network coverage. We refer to this as the *coverage-preserving, energy-efficient redundancy elimination* problem (see Fig. 1 for an illustration). The technical challenge associated with this problem lies in accurate and efficient detection of redundant sensors, and the selection of the maximum number of redundant sensors that can be safely turned off simultaneously.

A problem closely related to the coverage preserving redundancy elimination problem requires detection of the boundary of the network's coverage. As illustrated in Fig. 1, we define the *coverage-boundary* of a network to include not only the sensors situated on the outer periphery of the network, but also the ones that define "holes" in the coverage (sensors B, C, $\cdots$, H). Coverage boundary is important for identifying gaps in coverage and for optimizing sensor placement. It is easy to see that a sensor is redundant if its removal does not impact the coverage boundary.

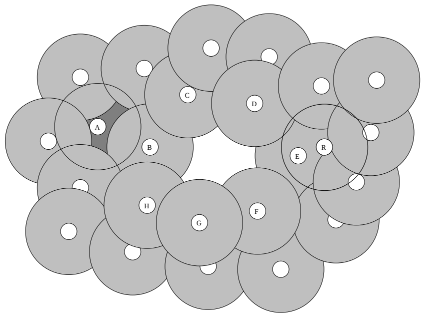


Fig. 1. Example of sensor network coverage – disks represent the coverage of the sensors situated at their center. Lightly shaded disks represent the coverage area of sensors situated on the boundary of the network. The dark areas belong to sensors that are not on the coverage boundary (e.g., sensor A). Sensor R is an example of a redundant sensor, since its coverage area is completely subsumed by other sensors.

While these problems are easily stated, several constraints contribute to the complexity of these problems. First, sensors must solve these problems in a distributed, efficient, and scalable manner. Second, solutions to these problems must be adaptive in nature, *i.e.*, when old sensors fail or new ones are deployed, the new solution must be rapidly computed from the previous solution. Finally, since RF interfaces have a limited transmission range, protocols must account for overheads of multi-hop routing. Since communication typically consumes more energy than computation, the protocols themselves need to be energy efficient.

The idea of turning off selected sensors in order to extend the lifetime of the network has been previously explored. Zhang and Hou [Zhang and Hou 2004] propose OGDC, a distributed algorithm, which works on the principle that an area is completely covered by a set of sensors if the crossing points generated by the sensing disks are covered. While OGDC's purpose is to minimize the number of sensors covering the crossing points, it declares as redundant only those sensors whose bitmapped sensing disks are completely covered by their transmission neighbors. Wang et al. [Wang et al. 2003] propose a protocol, CCP, based on a similar observation as OGDC. However, the complexity of determining the redundancy of a sensor is $O(N^3)$, where N is the number of sensors within twice the sensing range of that sensor. Huang and Tseng [Huang and Tseng 2003] propose an algorithm for determining the k-coverage of a region by a sensor network, based on the notion of k-perimeter-coverage. As with OGDC [Zhang and Hou 2004] and CCP [Wang et al. 2003] each sensor computes its k-coverage by communicating with all the sensors within twice its sensing range. The computational complexity of the method is $O(N \log N)$. However, in order for a sensor s to determine its redundancy, it has to ask all the sensors within twice its sensing range to reevaluate their perimeter coverage without s. Thus, for each sensor, the actual complexity of determining its redundancy is $O(N^2 \log N)$. Tian and Georganas [Tian and Georganas 2002] declare as redundant sensors whose transmission neighbors completely "sponsor" their sensing disks. Thus, the algorithm is completely localized. Ye et al. [Ye et al. 2003] propose a randomized algorithm for turning off sensors, whose primary goal however is not to maintain coverage.

In contrast to these existing results, we present a more efficient deterministic solution to the problem of accurately detecting redundant sensors and safely (from a coverage standpoint) turning them off. The complexity of our solution for each sensor $\mathtt{s}$ is $\mathtt{O}(\mathtt{N}\log\mathtt{N})$, where N is the number of Voronoi neighbors of $\mathtt{s}$. We also present efficient algorithms for maintaining the solution in cases of sensor failures and new sensor deployments. We show that the expected number of sensor affected is $\mathtt{O}(\log\mathtt{N})$. Moreover, since our solution relies on Voronoi diagrams, a sensor affected by a new or a failing sensor can recompute its redundancy in $\mathtt{O}(\log\mathtt{N})$ expected time [Devillers et al. 1992], whereas the solutions of [Zhang and Hou 2004; Wang et al. 2003] require $\mathtt{O}(\mathtt{N}^3)$ and the solution of [Huang and Tseng 2003] requires $\mathtt{O}(\mathtt{N}^2\log\mathtt{N})$.

While our work focuses on extending the coverage lifetime of wireless sensor networks, prior work [Zhang and Hou 2004; Wang et al. 2003] has explored the relationship between coverage and connectivity of sensor networks. Specifically, both [Zhang and Hou 2004] and [Wang et al. 2003] prove that if the transmission range of sensors equals or exceeds twice their sensing range, coverage implies connectivity. Moreover, the work of [Wang et al. 2003] extends this result for k-coverage and k-connectivity. The results presented in this paper, however, are not based on assumptions regarding the relationship between sensing and transmission ranges of sensors.

This paper makes the following specific contributions: in Section 4, we derive necessary and sufficient conditions for a sensor to be redundant and present an efficient distributed algorithm for the coverage-preserving, energy-efficient redundancy elimination problem. In Section 5, we present necessary and sufficient conditions for a sensor to be on the coverage boundary. We prove a lower bound of $\Omega(\mathtt{n}\log\mathtt{n})$ for any (serial) algorithm for the problem, where $\mathtt{n}$ is the total number of sensors in the network. We present a distributed algorithm for computing the coverage-boundary, whose serial counterpart has $\Omega(\mathtt{n}\log\mathtt{n})$ complexity. Both algorithms are based on the distributed and adaptive construction of Voronoi diagrams. In Section 6, we present efficient and scalable distributed algorithms for recomputing local Voronoi information in the event of sensor failures and deployment of new sensors. The algorithms are then used to efficiently maintain the redundancy and coverage-boundary information, without recomputing the entire solution. We show that our algorithms are efficient and prove their correctness and stability. In Section 7, we experimentally characterize the performance of our algorithms and conclusions are drawn in Section 8.

2. RELATED RESEARCH

The problem of coverage of a set of entities has been studied in a variety of contexts. Zhang and Hou [Zhang and Hou 2004] prove an interesting result that if the transmission range of sensors equals or exceeds twice the sensing range, coverage implies connectivity of the sensor network. Furthermore, they prove that, given a region $\mathtt{R}$ containing sensors, if each crossing point (intersection point of sensing disks of sensors) in $\mathtt{R}$ is covered by at least one other sensor in $\mathtt{R}$, then $\mathtt{R}$ is completely covered by the sensors. Based on this result, they propose OGDC, a distributed algorithm for selecting a subset of the sensors covering all crossing points. The selected sensors need to stay active, while the rest can be turned off. During the decision stage, each sensor can be in one of the following states: UNDECIDED, ON, and OFF. A sensor switches to OFF only when its sensing disk is completely covered by its neighbors. To verify this condition, the sensing disk is divided into small grids and a bitmap is used to indicate whether the center of each grid is covered by a neighbor of the sensor.

Wang et al. [Wang et al. 2003] prove a similar result regarding the connection between coverage and connectivity to [Zhang and Hou 2004] and extend the result to the case of k-coverage implying k-connectivity. Specifically, they prove that if the transmission range is larger or equal to the sensing range of sensors, k-coverage implies k-connectivity of the network and 2k-connectivity of the interior of the network. They also prove that if all crossing points in the deployment area are k-covered the area is k-covered. Based on this result, they propose a distributed algorithm for turning off redundant sensors. A sensor s_1 decides to be inactive if all the crossing points inside its sensing range are at least k-covered. The time complexity of this decision algorithm is $O(N^3)$, where N is the number of sensors within distance twice the sensing range of s_1. In comparison, the corresponding complexity of our algorithm is only $O(N \log N)$ (the time required to build the Voronoi diagram of the sensors placed within twice the sensing range of s_1).

Tian and Georganas [Tian and Georganas 2002] present an algorithm for detecting sensors whose coverage area is completely covered by other sensors. A sensor s_1 turns itself off only when each sector of its sensing disk is covered by one of the sensors located inside its sensing range. If s_2 is such a sensor, the algorithm conservatively considers only the sector generated by the intersection of the sensing disks of s_1 and s_2 to be “sponsored” by s_2. In contrast, our solution considers the entire “lune” of s_1, generated by the intersection of the sensing disks of s_1 and s_2, to be covered by s_2. Jiang and Dou [Jiang and Dou 2004] identified several shortcomings of [Tian and Georganas 2002] and provided a protocol for improving its performance.

Huang and Tseng [Huang and Tseng 2003] propose an algorithm for determining the k-coverage of a region by a sensor network. They prove that an area is k-covered if each sensor in the network is k-perimeter-covered, where a k-perimeter-covered sensor has each point on the perimeter of its sensing disk covered by at least k other sensors. Note that a 1-perimeter-covered sensor is equivalent to a non coverage-boundary sensor, as defined in Section 5. The solution for determining perimeter coverage requires each sensor to communicate with all the sensors within twice its sensing range N and its computational complexity is $O(N \log N)$. Their solution is then used to determine redundancy and schedule inactive periods for redundant sensors. However, to determine its redundancy, a sensor s has to ask all the sensors within twice its sensing range to reevaluate the coverage of their perimeter without s. Thus, a sensor has to run the perimeter coverage N times, making the complexity of the protocol $O(N^2 \log N)$.

The solutions we propose improve on the results of the above works by allowing a sensor to determine its redundancy and presence on the coverage-boundary, while communicating only with its Voronoi neighbors. If the number of Voronoi neighbors of a sensor is N, the computation complexity of our protocols is $O(N \log N)$. Moreover, we provide efficient algorithms that distributively maintain the redundancy and coverage-boundary information when existing sensors fail and new sensors are deployed. For each sensor failure or new sensor deployment, the expected number of sensors affected by the notifications sent in our algorithms is only $O(\log N)$. Moreover, the operation of updating the local information based on such a notification has an expected complexity of $O(\log N)$.

Ye et al. [Ye et al. 2003] present a randomized algorithm for extending the network lifetime by keeping only a subset of the sensors active at any given time. Initially each sensor sleeps for a duration distributed according to an exponential distribution function $\lambda e^{-\lambda t}$, where λ is the average rate of probing. When a sensor s_1 wakes up, it queries other sensors placed within its probing range, $\mathtt{R_p}$. If at least one other sensor answers, $\mathtt{s_1}$ turns back to sleep for a duration that is distributed according to an exponential distribution function. Otherwise, it becomes active for the remainder of its life. While the probing range is adjustable, enabling control of the sensor network's node density, the algorithm cannot guarantee complete coverage. The paper proves, however, that if the transmission range of sensors equals or exceeds $(1+\sqrt{5})\mathtt{R_p}$, and each square cell of size $\mathtt{R_p}$ contains at least one sensor, then the active network of PEAS is connected with high probability.

Slijepcevic and Potkonjak [Slijepcevic and Potkonjak 2001] introduce a centralized algorithm for finding the maximum number of disjoint subsets of sensors, or covers, where each cover completely covers the same area as the initial set of sensors. They prove the problem to be NP-complete. Their solution is based on building sets of monitored points into disjoint fields, such that each field contains a maximum number of points covered by the same sensors. At each stage, a cover is built in a greedy fashion by associating an objective function with sensors that cover a field. The objective function gives the likelihood of generating redundant covers. Initially, a critical field, one covered by the smallest number of sensors, is selected. Then, the sensor that covers the field and has the highest objective function is selected. Selected fields and sensors are marked accordingly and this process is repeated until a full cover is obtained.

Gupta et al. [Gupta et al. 2003] use the notion of field defined above to provide centralized and distributed algorithms for computing connected sensor covers for query regions. Initially, a sensor in the query region is selected. A candidate set of sensors, whose sensing disks intersect the sensing disks of selected sensors is built. For each candidate sensor, a path from it to one selected sensor is constructed. The candidate sensor, along with its path, which cover the maximum number of uncovered fields is selected. In the proposed distributed version of this algorithm, this computation is performed by the last selected sensor. Furthermore, the search for the candidate sensor is done within a radius of $\mathtt{2r}$, where $\mathtt{r}$ is the maximum distance between any two sensors whose sensing disks overlap.

Carle and Simplot-Ryl [Carle and Simplot-Ryl 2004] extend the protocol of [Dai and Wu 2003], providing connected dominating sets of nodes in ad-hoc networks, to construct area-dominating sets of sensor networks. In this protocol, sensors listen for a timeout period for advertisements containing decisions of their neighbors before deciding their own state. When the timeout expires, a sensor checks that the subgraph of the neighbors that have sent advertisements is connected and that those neighbors completely cover its sensing disk. If this is the case, the sensor sends a positive advertisement to its neighbors. Otherwise, it has the option to transmit a negative advertisement or not.

Meguerdichian et al. [Meguerdichian et al. 2001] define the coverage using the best covered and least covered paths between two sensors in the network as metrics. Haas [Haas 1997] presents algorithms for optimizing coverage under constraints on message path length. Shakkottai et al. [Shakkottai et al. 2003] study the coverage of a unit square by a given number of sensors, under the assumption that sensor failures will affect the coverage. We refer readers to [Cardei and Wu 2004] for an extensive survey of coverage problems in sensor networks.

The problem of sensor coverage has also received considerable attention in robotics (please see [Choset 2001] for a survey). Given a bounded domain, the problem requires a robot equipped with a sensor to build a complete map of the environment without any initial knowledge. This requires the robot to pass through specified points of the unknown

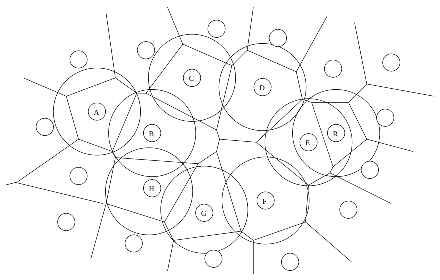


Fig. 2. Voronoi diagram of the sensors in Fig. 1. The circles represent the coverage disks of the labeled sensors. Note that only for A and R their coverage area completely covers their Voronoi cell. In the next section we will show that this is not a coincidence.

region. The notion of a *hierarchical generalized Voronoi graph* is used to incrementally construct the map using only line of sight data.

Distributed computation of Voronoi diagrams is addressed in [Stojmenovic 1999] in the context of routing in ad-hoc networks and in [Hu 1993] in the context of topology control of ad-hoc networks. In this approach, a sensor builds a Voronoi diagram of itself and its neighbors. While these protocols are localized, they provide an approximation of the actual Voronoi cells of sensors and can produce false-negatives for the problems addressed in this paper.

3. OVERVIEW OF VORONOI TESSELLATIONS

Given a set $\mathtt{S}$ of $\mathtt{n}$ sites $\mathtt{s}_1, \mathtt{s}_2, .., \mathtt{s}_\mathtt{n}$ in a plane, their Voronoi diagram is defined as the subdivision of the plane into $\mathtt{n}$ cells, one for each site, with the property that any point in the cell corresponding to a site is closer to that site than to any other site. Formally, the Voronoi cell corresponding to site $\mathtt{s}_\mathtt{i}$ is defined as

$$\mathtt{cell}_{\mathtt{vd}}(\mathtt{s}_\mathtt{i}) = \bigcap_{\mathtt{j}=1,\mathtt{j}\neq\mathtt{i}}^{\mathtt{n}} \{\mathtt{x} | \mathtt{dist}(\mathtt{s}_\mathtt{i}, \mathtt{x}) \leq \mathtt{dist}(\mathtt{s}_\mathtt{j}, \mathtt{x})\}$$

We use the notation $\mathtt{dist}(\mathtt{p}, \mathtt{q})$ to denote the Euclidean distance between two points p and q. Two Voronoi cells meet along a *Voronoi edge*, and three Voronoi cells meet at a *Voronoi vertex*. We call a site a *neighbor* of another site if the Voronoi cells of the two sites share an edge. We use these two terms interchangeably. Fig. 2 illustrates the Voronoi diagram of the network in Fig. 1.

A Delaunay triangulation of a set $\mathtt{S}$ of sites is defined as the unique triangulation of $\mathtt{S}$ such that no point in $\mathtt{S}$ is inside the circumcircle of any triangle of the triangulation. A Delaunay triangulation is the dual of the Voronoi diagram of $\mathtt{S}$, in the sense that two sites are vertices of the same Delaunay triangle iff they are Voronoi neighbors. We formally define the Delaunay distance as follows:

DEFINITION 3.1. *The Delaunay distance between two sites is the minimum number of hops on the Delaunay graph between the two sites.*

For example, in Fig. 2, sensors A and R are at Delaunay distance 4.

3.0.0.1 *Multiplicative Weighted Voronoi Diagram.*. A Multiplicative Weighted Voronoi Diagram (MWVD) is defined in a manner similar to a Voronoi diagram, with the addition

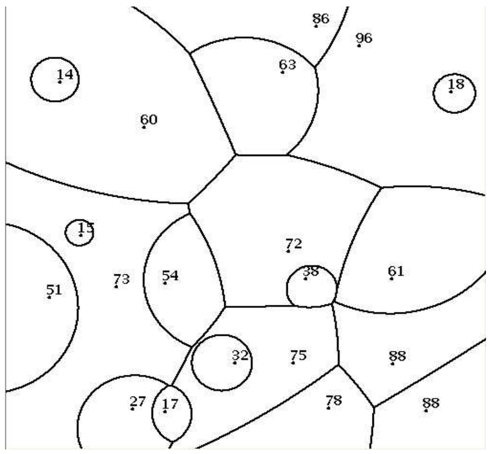


Fig. 3. Multiplicative weighted Voronoi diagram (MWVD) of 20 sensors. Each sensor is represented by a point, and its weight, determining the size of its cell.

of weights at each of the $\mathtt{n}$ sites. In the definition of the classical Voronoi diagram, the sites have equal weights. The MWVD replaces the Euclidean distance used by the Voronoi diagram with a new distance $\mathtt{d_{mv}}$ defined by

$$\mathtt{d_{mv}}(\mathtt{s_i}, \mathtt{x}) = \frac{\mathtt{dist}(\mathtt{s_i}, \mathtt{x})}{\mathtt{w_i}} \tag{1}$$

In this definition, $\mathtt{s_i}$ corresponds to one of the $\mathtt{n}$ sites in the plane, $\mathtt{w_i}$ is a weight associated with it, $\mathtt{dist}$ is the Euclidean distance function, and $\mathtt{x}$ corresponds to any point in the plane. The Multiplicative Weighted Voronoi cell of each site is formally defined as follows:

$$\mathtt{cell_{mwvd}}(\mathtt{s_i}) = \bigcap_{\mathtt{j}=1,\mathtt{j}\neq\mathtt{i}}^{\mathtt{n}} \{\mathtt{x} | \mathtt{d_{mv}}(\mathtt{s_i}, \mathtt{x}) \leq \mathtt{d_{mv}}(\mathtt{s_j}, \mathtt{x})\}$$

4. ENERGY-EFFICIENT COVERAGE

In this section we formalize the coverage-preserving energy efficient redundant sensor elimination problem, and provide a solution, RSE, based on Voronoi tessellations. We make the assumption that each sensor knows its two dimensional location. This is a reasonable assumption since, in the absence of this information, the coverage-boundary and the redundancy information cannot be uniquely or correctly determined (i.e., from topological information alone). While initially we assume that all the sensors in the network have the same sensing range, in Section 4.1 we extend our results to heterogeneous sensor networks, using *multiplicative weighted Voronoi diagrams*.

DEFINITION 4.1. *The* coverage of a sensor $\mathtt{s}$ *with planar coordinates* $(\mathtt{x}, \mathtt{y})$ *and sensing range* $\mathtt{r}$ *is a disk with center* $(\mathtt{x}, \mathtt{y})$ *and radius* $\mathtt{r}$. *We call this disk the coverage or sensing disk, and call its border the coverage or sensing circumcircle, denoted by* $\mathcal{C}(\mathtt{s})$. *We say that a point* $\mathtt{p}$ *is covered by a sensor* $\mathtt{s}$ *if* $\mathtt{dist}(\mathtt{s}, \mathtt{p}) \leq \mathtt{r}$.

The coverage of a network is the union of the coverage disks of all the sensors in the network. Formally, we have:

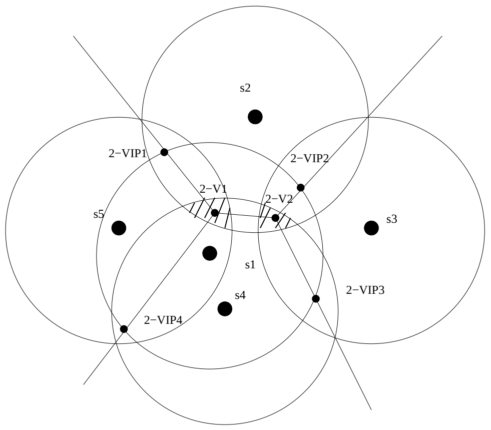


Fig. 4. Example redundant sensor, $\mathtt{s_1}$. Points $2 - \mathtt{V_1}$ and $2 - \mathtt{V_2}$ are 2-VVs of $\mathtt{s_1}$, and $2 - \mathtt{VIP_{1..4}}$ are 2-Voronoi Intersection Points of $\mathtt{s_1}$. Note that $2 - \mathtt{V_{1,2}}$ and $2 - \mathtt{VIP_{1..4}}$ are all covered by at least two of the Voronoi neighbors of $\mathtt{s_1}$.

DEFINITION 4.2. *The* coverage of a network *is the area $\mathcal{A}$ with the property that for any point $\mathtt{p} \in \mathcal{A}$, there exists at least one sensor $\mathtt{s}$ in the network such that $\mathtt{p}$ is covered by the coverage disk of $\mathtt{s}$.*

The definition of a redundant sensor follows naturally:

DEFINITION 4.3. *A sensor is said to be redundant if its sensing area is* completely *covered by other sensors.*

We define the 2-Voronoi diagram of a sensor in the following manner:

DEFINITION 4.4. *The 2-Voronoi diagram of a sensor $\mathtt{s}$ is the Voronoi diagram of the Voronoi neighbors of $\mathtt{s}$, when $\mathtt{s}$ is excluded. The 2-Voronoi Vertices (2-VV) of a sensor $\mathtt{s}$ are the Voronoi vertices of the 2-Voronoi diagram of $\mathtt{s}$. A 2-Voronoi Intersection Point (2-VIP) of $\mathtt{s}$ is the intersection between an edge of the 2-Voronoi diagram and the coverage circumcircle of $\mathtt{s}$. A 2-Voronoi edge (2-VE) of $\mathtt{s}$ is either a Voronoi edge between 2-Viv's of $\mathtt{s}$, or a Voronoi edge between a 2-VV and a 2-VIP of $\mathtt{s}$.*

Fig. 4 illustrates an example of a redundant sensor. In this example, sensor $\mathtt{s_1}$ has five 2-VEs, one between two 2-VVs, $2 - \mathtt{V_1}$ and $2 - \mathtt{V_2}$, and the rest between a 2-VV and a 2-VIP of $\mathtt{s_1}$.

In the following, we use $\mathtt{N_s}$ to denote the set of sensors that are the Voronoi neighbors of a sensor $\mathtt{s}$. The following lemma shows an important property of the coverage of Voronoi neighbors.

LEMMA 4.1. *For any sensor $\mathtt{s}$ in the network, the sensors in $\mathtt{N_s}$ are the ones closest to $\mathtt{s}$, out of all the sensors in the network. More precisely, if a sensor that is not in $\mathtt{N_s}$ covers a point $\mathtt{p}$ inside the coverage disk of $\mathtt{s}$, then $\mathtt{p}$ is also covered by at least one sensor in $\mathtt{N_s}$.*

PROOF. Let us consider the case in which sensor $\mathtt{s_1}$ has neighbors $\mathtt{s_2}$, and $\mathtt{s_3}$ whose coverage areas intersect each other and also the coverage area of $\mathtt{s_1}$ (Fig 5). We assume that all the sensors have the same coverage range, $\mathtt{r}$ (recall that this assumption can be relaxed with the use of MWVDs as opposed to Voronoi diagrams). Let $\mathtt{v}$ be

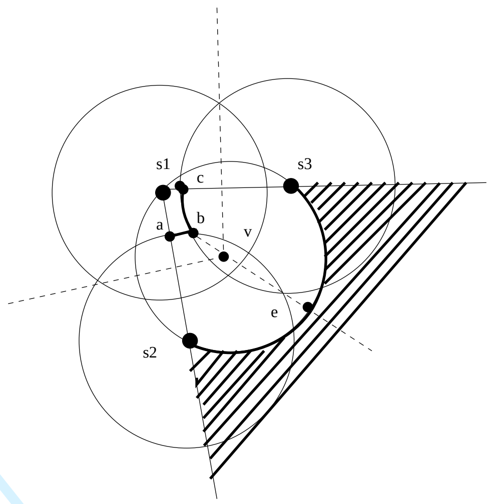


Fig. 5. Proof of Lemma 4.1. If sensors $\mathtt{s_1}$, $\mathtt{s_2}$ and $\mathtt{s_3}$ are mutual Voronoi neighbors, another sensor can only be placed in the hashed area. Otherwise, that sensor would be a Voronoi neighbor of $\mathtt{s_1}$.

the Voronoi vertex generated by the three sensors. Fig. 5 also shows the circumcircle of sensors $\mathtt{s_1}$, $\mathtt{s_2}$ and $\mathtt{s_3}$, centered at $\mathtt{v}$, containing no other sensor. Let $\mathtt{e}$ be the intersection point between this circle and the Voronoi edge generated by $\mathtt{s_2}$ and $\mathtt{s_3}$. The only area where another sensor, that is not a Voronoi neighbor of $\mathtt{s_1}$, can be placed, is the hashed area. Observe that $\mathtt{dist}(\mathtt{b},\mathtt{e}) = \mathtt{dist}(\mathtt{b},\mathtt{v}) + \mathtt{dist}(\mathtt{v},\mathtt{e})$. Therefore, due to triangle inequality, $\mathtt{dist}(\mathtt{b},\mathtt{e}) = \mathtt{dist}(\mathtt{b},\mathtt{v}) + \mathtt{dist}(\mathtt{s_3},\mathtt{v}) > \mathtt{dist}(\mathtt{s_3},\mathtt{b}) = \mathtt{r}$. Also, $\mathtt{dist}(\mathtt{s_2},\mathtt{a}) = \mathtt{dist}(\mathtt{s_2},\mathtt{b}) = \mathtt{r}$. It is easy to prove then, that the distance between any point on the arc $\widehat{\mathtt{ab}}$ and a point on the arc $\widehat{\mathtt{s_2e}}$ is greater than or equal to $\mathtt{r}$. The arcs are emphasized in Fig. 5. Similarly, the distance between any point on the arc $\widehat{\mathtt{bc}}$ and any point on the arc $\widehat{\mathtt{s_3e}}$ is greater than or equal to $\mathtt{r}$. Hence, any sensor placed in the hashed area covers less of $\mathtt{s_1}$'s coverage area than $\mathtt{s_2}$ and $\mathtt{s_3}$. The cases where the coverage areas of $\mathtt{s_2}$ and $\mathtt{s_3}$ do not intersect, or do not intersect the coverage area of $\mathtt{s_1}$ can be similarly proved. □

The following theorem, the main result of this section, translates the problem of finding a redundant sensor to a local examination of the sensor's Voronoi neighbors.

THEOREM 4.1. *A sensor* $\mathtt{s}$ *is redundant if and only if all the 2-VVs and 2-VIPs of* $\mathtt{s}$ *are covered by the Voronoi neighbors of* $\mathtt{s}$.

PROOF. **If** a sensor $\mathtt{s}$ is redundant, then all its 2-VVs and 2-VIPs are covered by the Voronoi neighbors of $\mathtt{s}$. This is illustrated in Fig. 4, which shows an example of a redundant sensor, $\mathtt{s_1}$. Since the coverage area of $\mathtt{s_1}$ is completely covered by other sensors, using Lemma 4.1, we infer that it is completely covered by the Voronoi neighbors of $\mathtt{s_1}$. Furthermore, since the coverage area of a sensor is a circle, any three neighbors of $\mathtt{s_1}$ that are mutual neighbors when $\mathtt{s_1}$ is excluded, will cover a common area. Fig. 4 shows the common areas of Voronoi neighbors $\mathtt{s_2}$, $\mathtt{s_4}$, and $\mathtt{s_5}$, and $\mathtt{s_2}$, $\mathtt{s_3}$, and $\mathtt{s_4}$, respectively, as the hashed areas. The common area of such three-neighbor sets contains the Voronoi vertex generated by them. This Voronoi vertex is a 2-VV of $\mathtt{s_1}$, and is therefore covered by three

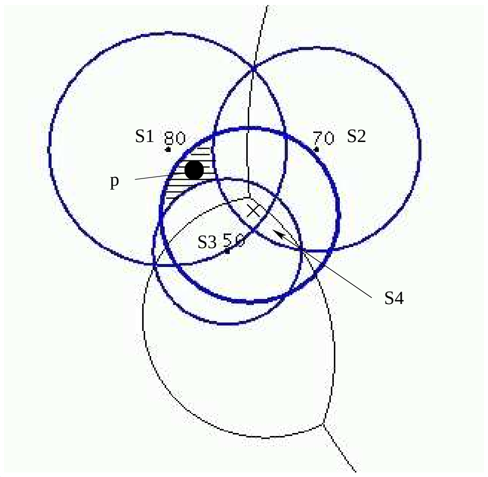


Fig. 6. Coverage of Voronoi neighbors with varying sensing ranges. The numbers associated with each of the three sensors, 80,70,50, represent the corresponding sensing radii of the sensors.

Voronoi neighbors of s_1. In a similar fashion it can be proved that each 2-VIP of s_1 is covered by at least two of the Voronoi neighbors of s_1.

Only if all the 2-VVs and 2-VIPs of a sensor are covered by the sensor's Voronoi neighbors, the sensor is redundant. Fig. 4 illustrates the proof. The 2-VVs, 2-VEs, and 2-VIPs of s_1 define a partition of the coverage area of sensor s_1, consisting of four regions. Each region of the partition is associated with a Voronoi neighbor of s_1. Since the 2-VVs and 2-VIPs of s_1 are covered by the Voronoi neighbors of s_1, following the definition of Voronoi diagrams (Section 3), each Voronoi neighbor of s_1 covers the 2-Voronoi vertices, 2-VIPs, and 2-VEs that it generates. Thus, the region of the partition associated with a Voronoi neighbor of s_1 is completely covered by that neighbor, making s_1 redundant. □

4.1 Heterogeneous Sensor Networks

Since several types of sensors may be employed in wireless sensor networks, it is unreasonable to assume equal sensing ranges. In this subsection we show that the above results hold even for wireless sensor networks with different sensing ranges. For this, we use Multiplicative Weighted Voronoi Diagrams (see Section 3). If the weight w_i of a sensor s_i is replaced with its sensing range, r_i, the distance function of a point x relative to s_i, from Equation 1, becomes: $d_{mv}(s_i, x) = dist(s_i, x)/r_i$. With this we can prove that a variant of Lemma 4.1 is valid for heterogeneous networks.

LEMMA 4.2. *There exists no sensor that covers more of the coverage disk of a sensor* s *than the Multiplicative Weighted Voronoi neighbors of* s.

PROOF. Fig. 6 shows three sensors, s_1, s_2 and s_3, their corresponding weights and sensing ranges and their MWVD. Assume that there is another sensor s_4 that is not a Voronoi neighbor of s_1 but covers more of its coverage disk than s_2 and s_3. Let p be such a point, covered only by s_1 and s_4 (placed in the hashed area in Fig. 6). Since $d(p, s_1)/r_1 < 1$, $d(p, s_4)/r_4 < 1$, $d(p, s_2)/r_2 > 1$ and $d(p, s_3)/r_3 > 1$, p will be in s_1 or s_4's Multiplicative Weighted Voronoi cell and not in the cell of s_2 or s_3. Thus, s_1 and

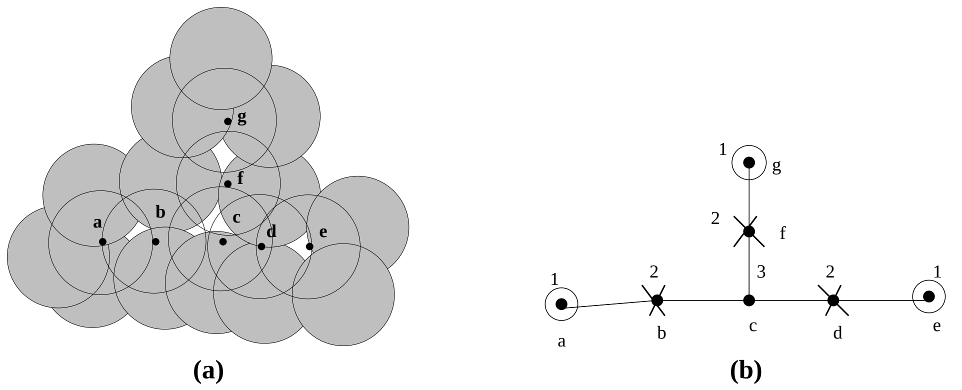


Fig. 7. (a) Example of a sensor network with blind points. Sensors $a, .., g$ are all redundant. However, if all of them are turned off simultaneously, the areas colored white are left uncovered. (b) Redundant graph of the network in (a) – the numbers associated with the nodes represent their degree in the redundant graph. The circled nodes represent winners in the fi rst round, and the crossed nodes represent their direct neighbors, losers. Sensor c is not a loser in the fi rst round since none of its neighbors is a winner, but it is a winner in the second round.

s_4 must share a Voronoi edge and are Multiplicative Weighted Voronoi neighbors. □

With Lemma 4.2, Theorem 4.1 applies directly to heterogeneous sensors and its proof follows the previous one.

4.2 Distributed detection of redundant sensors

In RSE, sensors locally decide their redundancy, based on the position of their Voronoi neighbors. The computation of the associated 2-Voronoi diagram of a sensor takes $O(N \log N)$ time, where N is the number of Voronoi neighbors of the sensor. A sensor can then verify in $O(N)$ time the coverage of each of its 2-VVs and 2-VIPs, by the Voronoi neighbors that generated it. This is because the number of Voronoi vertices of a Voronoi diagram is on the order of the number of sites of the Voronoi diagram and each verification takes constant time. Thus, the total complexity of locally determining redundancy is $O(N \log N)$.

4.3 Blind Points

If two redundant sensors that are also Voronoi neighbors, decide to turn off simultaneously, an area between them may be left uncovered. Such an area is called a blind point [Tian and Georganas 2002]. Fig. 7(a) shows an example of blind points created when *all* the redundant sensors simultaneously turn off. We need to find the maximum number of redundant sensors that can be turned off without generating blind points.

One solution to this problem, proposed by Tian and Georganas [Tian and Georganas 2002], uses a random back-off scheme. For RSE, we propose an alternate solution, based on a modification of a distributed approximation of the maximal independent set (MIS) problem [Luby 1985]. Let $G_R = (V_R, E_R)$ be the redundancy graph of the network, where V_R is the set of redundant sensors. There is an edge $e \in E_R$ between two redundant sensors if and only if they are Voronoi neighbors. Then, the blind point problem is equivalent to one of finding the maximum independent set of the redundancy graph, G_R.

The selection algorithm employed in RSE, similar to the one proposed by Luby [Luby 1985], proceeds in rounds. In each round, a redundant sensor sends to its redundant

Voronoi neighbors, a message containing its identity and remaining energy level. When a redundant sensor collects all such messages from its redundant Voronoi neighbors, it compares its energy level with the values received. A sensor that has the smallest energy level is a winner. Subsequently, winners send to all their redundant Voronoi neighbors, a message specifying their state. A redundant sensor that receives such a message from one of its redundant Voronoi neighbors becomes a loser. At the end of each round, the winners are turned off. Winners and losers of a round do not participate in subsequent rounds. An analysis of the expected number of rounds ($O(\log n)$ where n is the number of initial participants) and a proof of termination of a similar algorithm can be found in [Luby 1985].

Fig. 7(b) shows the redundancy graph of the sensor network from Fig. 7(a), and shows a trace of the selection algorithm. After the first round, sensors a, e, and g are winners and are turned off, and sensors b, d, and f are losers. In the second round, c is the unique participant, and a winner.

4.4 Management of Redundant Sensors

A winner in the above protocol can be safely turned off, since none of its redundant Voronoi neighbors are turned off. This is a special case of a sensor failure. In Section 6.3 we provide an algorithm for efficiently and correctly updating the local Voronoi information of sensors affected by sensor failures. The algorithm can be easily adapted to this situation, the only difference being that the affected sensors can be notified of the "failure", and do not have to discover it themselves.

Before turning off, a winning sensor chooses a random sleep time. When its sleep time expires, the sensor wakes up and checks the presence of its former Voronoi neighbors. If one (or more) of them have failed during its sleep, the reawakened sensor recomputes its redundancy information and remains active if no longer redundant. The process through which such a sensor notifies other sensors in order to update their Voronoi information is called *join* and is described in detail in Section 10.

5. PLANAR COVERAGE BOUNDARY

A problem closely related to the coverage-preserving, energy-efficient redundancy elimination problem is one of finding planar coverage boundary. It is easy to see that a sensor is redundant iff its removal does not alter the boundary of the network. In this section we formally define the coverage-boundary problem and provide an efficient distributed solution. Using Definition 4.1, we say that a sensor is on the boundary of the coverage of the network iff the circumcircle of its sensing disk is not entirely covered by the coverage disks of all the other sensors in the network.

DEFINITION 5.1. *A sensor s is said to be* on the boundary *of the coverage of a network if there exists a point p on $\mathcal{C}(s)$ such that p is not covered by the coverage disk of any other sensor in the network. However, if two sensors have the same position, we do not consider that any point on the circumcircle of one of the sensors is covered by the disk of the other sensor.*

With these definitions in place, we formally describe the *coverage-boundary* problem:

THE COVERAGE-BOUNDARY PROBLEM 1. *Given a set of n sensors in the plane, each with a sensing range r, find all the sensors that are on the boundary of the coverage.*

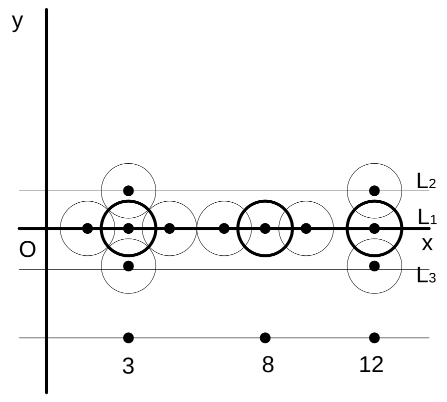


Fig. 8. Example for the proof of the lower bound for the coverage-boundary problem. We show the transformation from two sets $\mathtt{S}_1 = \{8, 3\}$ and $\mathtt{S}_2 = \{3, 12\}$ to an instance of the coverage-boundary problem in the Cartesian plane. The lowest horizontal line shows the coordinates on the x axis of the main circles. The main circles are represented using thicker arcs. Note that the circumcircle of the main circle corresponding to value 3 is completely covered by the secondary circles on $\mathtt{L}_1$, $\mathtt{L}_2$ and $\mathtt{L}_3$, whereas the circumcircles of the main circles for 8 and 12 are not completely covered by the secondary circles.

The following theorem establishes a lower bound for any solution of the coverage-boundary problem.

THEOREM 5.1. *The coverage-boundary problem has a* $\Omega(\mathtt{n}\log\mathtt{n})$ *lower bound.*

PROOF. The proof is based on a linear-time transformation from the set equality problem to the coverage boundary problem. The set equality problem is stated as follows: Given two sets S_1 and S_2 of real numbers, both of size n, determine if the two sets are equal. The problem is known to have a $\Omega(\mathtt{n}\log\mathtt{n})$ lower bound.

The transformation works in the following manner. Consider three horizontal lines in the Cartesian plane. The first line, $\mathtt{L}_1$, is the x axis, $\mathtt{L}_2$ and $\mathtt{L}_3$ are lines parallel to $\mathtt{L}_1$ going through the points $(0, \sqrt{2})$ and $(0, -\sqrt{2})$, respectively (Figure 8). For each element $\mathtt{e}_1$ from the set $\mathtt{S}_1$, insert three circles with their centers situated on $\mathtt{L}_1$, in the following manner. The main circle has the center at $(\mathtt{e}_1, 0)$ and radius 1, and the two secondary circles have centers at $(\mathtt{e}_1 - \sqrt{2}, 0)$ and $(\mathtt{e}_1 + \sqrt{2}, 0)$, respectively, both with a radius of 1. Similarly, for each element $\mathtt{e}_2$ from the set $\mathtt{S}_2$ add three circles in the plane. The main circle is again on $\mathtt{L}_1$, centered at $(\mathtt{e}_2, 0)$ and radius one. The secondary circles have their centers on $\mathtt{L}_2$ and $\mathtt{L}_3$, $(\mathtt{e}_2, \sqrt{2})$ and $(\mathtt{e}_2, -\sqrt{2})$, respectively, both with radius 1.

If two elements in the sets $\mathtt{S}_1$ and $\mathtt{S}_2$ are equal, the circumcircle of the main circles generated by the elements is completely covered by the disks of the secondary circles. Note that the two main circles generated by these elements do not cover each other, as stated in Definition 5.1. It is easy to see that the two sets $\mathtt{S}_1$ and $\mathtt{S}_2$ are equal if and only if the circumcircles of the main circles generated in the Cartesian plane are completely covered by the disks of the secondary circles. That is, $\mathtt{S}_1$ and $\mathtt{S}_2$ are equal iff the result of solving the coverage-boundary problem finds *all* the main circles as not being on the coverage-boundary. The transformation takes time $\mathtt{O}(\mathtt{n})$ since for every element in the two sets, a constant number of circles (three) are added in the Cartesian plane. This proves that the coverage-boundary problem has a $\Omega(\mathtt{n}\log\mathtt{n})$ lower bound. □

Our solution of this problem is based on the following theorem.

THEOREM 5.2. *A sensor* $\mathtt{s}$ *is on the boundary of the network if and only if the Voronoi cell of* $\mathtt{s}$ *is not completely covered by its sensing range.*

PROOF. **If** sensor $\mathtt{s_i}$ is on the boundary, the coverage disk of $\mathtt{s_i}$ does not entirely cover the Voronoi cell of $\mathtt{s_i}$. We only consider the case where $\mathtt{s_i}$'s Voronoi cell is bounded, since otherwise its cell is clearly not covered. Following Definition 5.1, let us take a point $\mathtt{x}$ on $\mathtt{s_i}$'s coverage circumcircle, such that $\mathtt{x}$ is not covered by the disk of any other sensor (Fig. 9(b)). Since $\mathtt{s_i}$'s cell is bounded, $\overline{\mathtt{d_i x}}$ intersects one of the Voronoi edges of $\mathtt{s_i}$'s Voronoi cell. Let that Voronoi edge be $\overline{\mathtt{v_1 v_2}}$, generated by $\mathtt{s_i}$ and $\mathtt{s_j}$, and the intersection point be $\mathtt{y}$. Point $\mathtt{y}$ cannot be inside $\mathtt{s_i}$'s coverage disk, since $\mathtt{x}$ would be covered by the coverage disk of sensor $\mathtt{s_j}$, contradicting Definition 5.1 (Fig. 9(a)). Point $\mathtt{y}$ is then outside $\mathtt{s_i}$'s coverage disk. Since $\mathtt{y}$ belongs to $\mathtt{s_i}$'s Voronoi cell, there exists a point in $\mathtt{s_i}$'s Voronoi cell not covered by $\mathtt{s_i}$'s coverage disk.

Only if the coverage disk of sensor $\mathtt{s_i}$ does not entirely cover its Voronoi cell, then $\mathtt{s_i}$ is on the boundary of the network. To prove this, let us consider a point $\mathtt{y}$ situated on a Voronoi edge belonging to $\mathtt{s_i}$'s cell, such that $\mathtt{y}$ is not covered by $\mathtt{s_i}$'s sensing disk (Fig. 9(b)). Let $\mathtt{x}$ be the intersection of $\overline{\mathtt{d_i y}}$ and $\mathcal{C}(\mathtt{s_i})$. Since the Voronoi cell of sensor $\mathtt{s_i}$ is convex, $\mathtt{x}$ is inside $\mathtt{s_i}$'s Voronoi cell. Then $\mathtt{r} = \mathtt{dist}(\mathtt{s_i}, \mathtt{x}) < \mathtt{dist}(\mathtt{s_j}, \mathtt{x}), \forall \mathtt{j} \neq \mathtt{i}$, hence $\mathtt{x}$ is not covered by any other sensor. According to Definition 5.1, $\mathtt{s_i}$ is on the boundary. □

The coverage-boundary sensors enjoy a special relationship with the redundant sensors. More precisely, a redundant sensor is *not* a boundary sensor, since by Definition 5.1, its circumcircle is also completely covered by other sensors. The following theorem describes another important property of redundant sensors.

THEOREM 5.3. *The temporary inactivation of a redundant sensor will not switch the boundary state of a sensor from non coverage-boundary to coverage-boundary.*

PROOF. The sensors that are affected by the inactivation of a redundant sensor, $\mathtt{r}$, are those whose coverage area intersects the coverage area of $\mathtt{r}$. According to Definition 5.1, a sensor is not on the coverage-boundary of the network if the circumcircle of its coverage

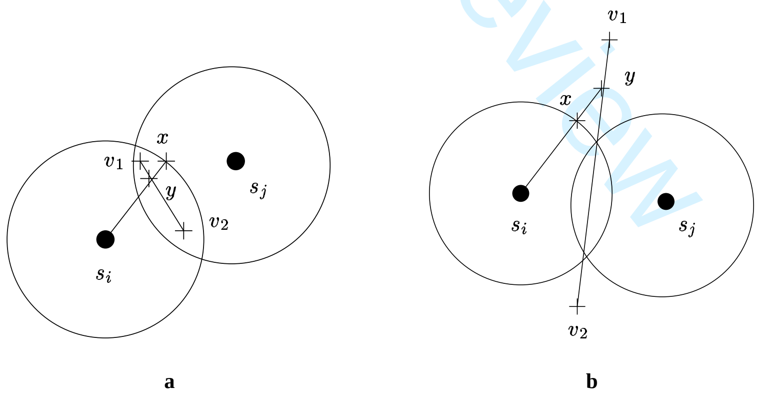


Fig. 9. Proof of Theorem 5.2. (a) Example of the case in which $\overline{\mathtt{s_i x}}$ intersects $\overline{\mathtt{v_1 v_2}}$ inside $\mathtt{s_i}$'s coverage disk. (b) The intersection point is outside $\mathtt{s_i}$'s coverage disk.

area is completely covered by other sensors. Let us say that sensor $\mathtt{s}$ is affected by the inactivation of sensor $\mathtt{r}$, and $\mathtt{s}$ is not on the coverage-boundary of the network before $\mathtt{r}$ is turned off. The coverage circumcircle of $\mathtt{s}$ is then divided into two arcs: one that is covered by $\mathtt{r}$ and one that is not. The one that is not covered by $\mathtt{r}$ is clearly covered by other sensors. Since $\mathtt{r}$ is redundant and the coverage areas of sensors are circles (convex regions) the arc of $\mathtt{s}$'s coverage circumcircle that is covered by $\mathtt{r}$ is also covered by other sensors. □

5.1 Heterogeneous Sensor Networks

Theorem 5.2 can be extended to the case in which sensors have different sensing ranges by using Multiplicative Weighted Voronoi Diagrams (MWVD). As in Section 4.1, we replace, in Equation 1, the weight $\mathtt{w_i}$ of a sensor $\mathtt{s_i}$ with its sensing range, $\mathtt{r_i}$. The distance function of a point $\mathtt{x}$ relative to $\mathtt{s_i}$ becomes $\mathtt{d_{mv}}(\mathtt{s_i}, \mathtt{x}) = \mathtt{dist}(\mathtt{s_i}, \mathtt{x})/\mathtt{r_i}$

To see why Theorem 5.2 still holds for the MWVD cells, consider the case in which the sensing range of sensor $\mathtt{s_i}$ does not entirely cover its MWVD cell, and the cell is bounded. If there is a Voronoi arc that is not covered, take a point $\mathtt{y}$ on the uncovered part of the arc. Let $\mathtt{x}$ be the intersection between $\overline{\mathtt{s_i y}}$ and $\mathcal{C}(\mathtt{s_i})$. Since $\mathtt{s_i}$'s Voronoi cell is convex, $\mathtt{x}$ is inside $\mathtt{s_i}$'s Voronoi cell. Then, by the definition of the Multiplicative Weighted Voronoi cell, $\mathtt{dist}(\mathtt{s_i}, \mathtt{x})/\mathtt{r_i} < \mathtt{dist}(\mathtt{s_j}, \mathtt{x})/\mathtt{r_j}$, $\forall \mathtt{j} \neq \mathtt{i}$. Since $\mathtt{dist}(\mathtt{s_i}, \mathtt{x}) = \mathtt{r_i}$, we conclude that $\mathtt{dist}(\mathtt{s_j}, \mathtt{x}) > \mathtt{r_j}$. Thus, $\mathtt{x}$ is not covered by the disk of any other sensor.

5.2 Distributed Computation of Boundary Sensors

Based only on the position information of its Voronoi neighbors and of itself, any sensor can decide its presence on the network coverage-boundary in the following manner. First, generate the Voronoi diagram of a set of sites consisting only of itself and its Voronoi neighbors; this step takes $\mathtt{O}(\mathtt{N} \log \mathtt{N})$ time, where $\mathtt{N}$ is the number of Voronoi neighbors. Then, check if its distance to each of the Voronoi vertices generated is less than the sensing range; this step takes $\mathtt{O}(\mathtt{N})$ time, since the number of Voronoi vertices of a Voronoi diagram is on the order of the number of generating sites. Thus, the total complexity of this operation is $\mathtt{O}(\mathtt{N} \log \mathtt{N})$.

6. DISTRIBUTED COMPUTATION AND MAINTENANCE OF VORONOI CELLS

The resource and scalability constraints of a sensor network make the existence of a centralized entity, that would compute the global Voronoi overlay, an unreasonable assumption. Instead, every sensor must keep enough data to allow for the local computation of the desired information. Our goal is to permit each sensor to correctly determine its Voronoi cell and the identity of its Voronoi neighbors. This information is sufficient to autonomously decide its own presence on the boundary, according to Theorem 5.2, or its redundancy, according to Theorem 4.1.

6.1 Initial Distributed Computation of Voronoi Generators

Initially, each sensor knows only its own location, and stores it in a local repository. Each step of the algorithm requires every sensor to send a message containing the information kept in its repository to all its network neighbors, i.e., sensors that lie within its transmission range. Upon receiving this message from a neighbor, a sensor adds the information received about new sensors to its local repository. It then recomputes its Voronoi cell with the information contained in the repository. At the end of every step, each sensor checks to see if the updates received brought it new information. The algorithm continues for a sensor until no new information is received from all its network neighbors. Upon termination, each sensor discards all the information from the local repository, with the exception of its Voronoi neighbors and its network neighborhood information.

After $\mathtt{k}$ steps, where $\mathtt{k}$ is the diameter of the network, every sensor learns of every other sensor in the network. Therefore the algorithm terminates in $\mathtt{O}(\mathtt{k})$ time. At each step, every sensor broadcasts exactly one message, hence the total number of messages is $\mathtt{O}(\mathtt{kn})$, where $\mathtt{n}$ is the number of sensors in the network. The number of steps in the distributed algorithm is optimal. This is because two Voronoi neighbors may be separated in the network neighbor graph by $\mathtt{k}$ links. The construction for this is simply a set of sensors organized in a ring in which each sensor only sees two other sensors in the network. In this case, the diameter $\mathtt{k}$ is $\mathtt{n}/2$ and two sensors that are Voronoi neighbors may be $\mathtt{n}/2$ hops away. Our experiments in Section 7.1 show that for uniform distributions of sensors, the actual number of steps required by a sensor to collect enough information to accurately build its Voronoi cell is much smaller than the network's diameter.

6.2 Deployment of New Sensors

The deployment of new sensors has the potential to change the set of redundant sensors and the coverage-boundary of the network. In both cases, this boils down to updating the local Voronoi information of the affected sensors. Fig. 10 illustrates an example in which a new sensor $\mathtt{ns}$ joins an existing network. Only sensors whose Voronoi cell is affected by the presence of the new sensor have to be notified about the new sensor. It is well known that the circumcircle of a Delaunay triangle contains no other site (sensor). We call such a circle a Delaunay circle. Hence, only sensors that generate a Delaunay triangle whose Delaunay circumcircle contains the new sensor, must be notified. We say that such a triangle is *in conflict* with the new sensor, and the sensors that generate the triangle are called *notifiables.* All the sensors in Fig. 10 are notifiable with regard to $\mathtt{ns}$.

Before proceeding with the description of the join algorithm, we present two useful lemmas, direct consequences of Lemmas 4.3 and 4.4 from [Devillers et al. 1992].

LEMMA 6.1. *Given a randomly deployed sensor network of size* $\mathtt{n}$, *the expected number of sensors affected by the random deployment of a new sensor is* $\mathtt{O}(\log \mathtt{n})$.

LEMMA 6.2. *The expected number of Voronoi neighbors of a sensor in a randomly deployed sensor network is constant.*

The Join Algorithm. Whenever a new sensor is deployed, it sends a beacon message and a sensor that receives this beacon is said to *detect* the new sensor. Thus, the new sensor can be detected by another sensor only if the radio transmission ranges of the two sensors exceeds the distance between them. A sensor that detects the presence of a new sensor needs to notify other sensors affected by its presence. Since more than one sensor can detect the new sensor, multiple, redundant notifications might be sent in the network. To avoid this, we choose only one sensor, called *introducer* ($\mathtt{S}$), to perform the notification. We choose $\mathtt{S}$ to be the sensor whose Voronoi cell contains the position of the newly deployed sensor. In Fig. 10(a), the introducer of $\mathtt{ns}$ is $\mathtt{s}_1$. If the new sensor is connected to the network, according to the definition of a Voronoi cell (Section 3), our choice guarantees that the introducer will detect the new sensor.

The introducer initiates the notification process, and during the process, each notified sensor uses only local information to determine its behavior. Each sensor has a list of

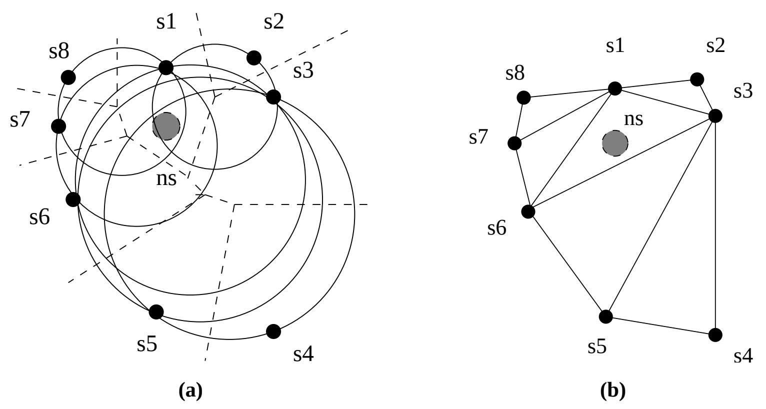


Fig. 10. Example of new sensor deployment: the gray circle, `nd`, represents the new sensor. (a) The Voronoi diagram of the old sensors. The circles denote circumcircles of Delaunay triangles. (b) The Delaunay triangulation of the old sensors.

incident Delaunay triangles, and the introducer, S, traverses its list of Delaunay triangles, identifies the ones in conflict with the new sensor, and isolates the notifiables among its Voronoi neighbors. Then, sensor S sorts its notifiables based on its Euclidean distance to them. Each sensor uses three colors to differentiate the notifiables in its local view. Initially, all the notifiable Voronoi neighbors are $white$. Upon completion of the coloring algorithm, they will be either $gray$ or $black$, and the sensor, in this case S, will only contact the black ones. A gray colored notifiable denotes a sensor that can be easier notified by one of the black notifiables, thus its notification should be postponed.

The coloring algorithm proceeds as follows. First, S takes its closest white notifiable, marks it black, and removes it from the white list. Then, it traverses the white list in clockwise order starting from the newly removed notifiable. If a white notifiable, W, is part of a local Delaunay triangle that has a gray or black vertex, G, such that $\mathtt{dist}(\mathtt{S},\mathtt{W}) > \mathtt{dist}(\mathtt{G},\mathtt{W})$, it marks W gray and removes it from the white list. Since W is closer to G than to S, it can be easier notified by G than by S. If, at the end of the traversal, a notifiable has been marked gray, the traversal is repeated, until no more white notifiables are marked gray. Then, if the white list is not empty, the closest white notifiable is again removed and marked black and the above process is repeated. In the example from Fig. 10, sensor $\mathtt{s}_1$ first colors sensor $\mathtt{s}_2$ black and in the subsequent traversal colors sensor $\mathtt{s}_3$ gray. It then colors sensor $\mathtt{s}_8$ black, and in the first traversal colors sensor $\mathtt{s}_7$ gray. A second traversal colors sensor $\mathtt{s}_6$ gray.

On completion of the coloring phase, S sends a notification to each black notifiable. For such a notifiable B, the message contains the identity and position of the new sensor, the identity of the notifier, in this case S, and the identities of all the gray notifiables in the local view of S, that B needs to notify. A sensor N that receives a new sensor notification, ignores the notification if it has already received it from another Voronoi neighbor. Otherwise, N creates its own white list of notifiable Voronoi neighbors. N removes from this list all the gray sensors contained in the notification, and marks them black, since it has to notify them. It also removes from the white list the notifier (the sensor from whom it has received this notification) and marks it gray, since it should not notify it. N then repeats the coloring phase described above, and sends to each black notifiable a notification with the same structure as the one that itself has received.

Example. Figure 10 illustrates an example of the notification process. Here, $\mathtt{ns}$ is a newly deployed sensor. As explained above, sensor $\mathtt{s}_1$ sends a message only to sensors $\mathtt{s}_2$ and $\mathtt{s}_8$. When sensor $\mathtt{s}_3$ receives the notification from sensor $\mathtt{s}_2$, it first constructs its list of notifiables, consisting of sensors $\mathtt{s}_2$, $\mathtt{s}_4$, $\mathtt{s}_5$, $\mathtt{s}_6$, and $\mathtt{s}_1$. It then colors sensor $\mathtt{s}_2$ gray, since sensor $\mathtt{s}_2$ is the one that sent it the notification. It then traverses the list of notifiables. In the first traversal it colors sensor $\mathtt{s}_1$ gray. We note that this is natural, since we already know that sensor $\mathtt{s}_2$ is closer to sensor $\mathtt{s}_1$ than sensor $\mathtt{s}_3$. In the next traversal, sensor $\mathtt{s}_3$ colors sensor $\mathtt{s}_6$ gray, since $\mathtt{dist}(\mathtt{s}_3, \mathtt{s}_6) > \mathtt{dist}(\mathtt{s}_1, \mathtt{s}_6)$. The next traversal colors sensor $\mathtt{s}_5$ gray, and the last colors sensor $\mathtt{s}_4$ gray. When sensor $\mathtt{s}_6$ receives the notification from sensor $\mathtt{s}_7$, which received it from sensor $\mathtt{s}_8$, it constructs its list of notifiables, $\mathtt{s}_7$, $\mathtt{s}_1$, $\mathtt{s}_3$, $\mathtt{s}_5$. It marks sensor $\mathtt{s}_7$ gray and propagates the gray color to sensors $\mathtt{s}_1$ and $\mathtt{s}_3$. It then marks sensor $\mathtt{s}_5$ black and notifies it. The notification similarly reaches sensor $\mathtt{s}_4$.

The local coloring and propagation of gray is meant to reduce the number of redundant messages. Sensor $\mathtt{s}_3$ will not notify sensors $\mathtt{s}_6$, $\mathtt{s}_5$ and $\mathtt{s}_4$, since sensor $\mathtt{s}_8$ will notify sensor $\mathtt{s}_7$, which in turn will notify sensor $\mathtt{s}_6$, and so on. This method will not eliminate all redundant notifications, but by locally reducing the distance between a notifier and a notifiable, this method has the advantage of reducing the number of actual messages that need to be sent for a notification. This is because the chance of two Voronoi neighbors of being in each other's radio transmission range increases when their Euclidean distance decreases.

We now present several properties of the join algorithm.

CORRECTNESS 1. *Upon completion of the algorithm, all the notifiable sensors have been notified.*

PROOF. By induction on the Delaunay distance between a notifiable and the introducer. Fig. 11 illustrates the proof, where $\mathtt{ns}$ represents the new sensor. The basis is simple, since all notifiables at distance 1 from the introducer are clearly notified. For the induction step, we assume that all notifiables at distance $\mathtt{l} - 1$ are notified. Let $\mathtt{s}_1$ be a notifiable at Delaunay distance $\mathtt{l}$ from the introducer. Being a notifiable, $\mathtt{s}_1$ has at least a triangle in conflict with the new sensor, $\mathtt{ns}$. Since in terms of the Euclidean distance $\mathtt{s}_2$ and $\mathtt{s}_3$ are $\mathtt{s}_1$'s closest Voronoi neighbors to $\mathtt{ns}$, the triangle $\Delta\mathtt{s}_1\mathtt{s}_2\mathtt{s}_3$ is such a conflict triangle. Sensor $\mathtt{s}_1$ is at Delaunay distance $\mathtt{l}$ from the introducer, $\mathtt{s}_\mathtt{i}$, therefore it has at least one Voronoi neighbor at Delaunay distance $\mathtt{l} - 1$ from $\mathtt{s}_\mathtt{i}$. Since $\mathtt{s}_\mathtt{i}$ is the closest sensor to $\mathtt{ns}$, no other Voronoi neighbor of $\mathtt{s}_1$ is at a smaller Delaunay distance from $\mathtt{s}_\mathtt{i}$ than $\mathtt{s}_2$ and $\mathtt{s}_3$. Hence, at least one of $\mathtt{s}_2$ or $\mathtt{s}_3$ are at Delaunay distance $\mathtt{l} - 1$ from $\mathtt{s}_\mathtt{i}$. If both are, since both are notifiable, they will both be notified. At least the one closer to $\mathtt{s}_1$ will then notify it, since both will detect the triangle $\Delta\mathtt{s}_1\mathtt{s}_2\mathtt{s}_3$ to be in conflict with $\mathtt{ns}$. If only one of $\mathtt{s}_2$ or $\mathtt{s}_3$ is at distance $\mathtt{l} - 1$, then that one will notify $\mathtt{s}_1$. □

TERMINATION 1. *The algorithm will terminate, i.e., notifications will not be sent indefinitely.*

PROOF. Each notifiable sensor can receive notifications only from its Voronoi neighbors that are also notifiable. Only notifiable sensors receive notifications. A notifiable sensor propagates a notification only to notifiable Voronoi neighbors, and only when receiving for the first time a notification concerning a new sensor. Since the number of expected

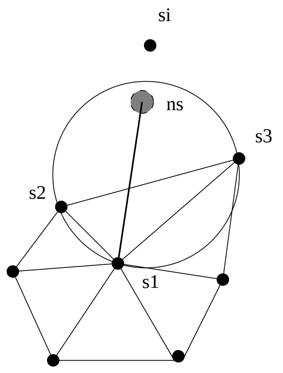


Fig. 11. Illustration of the correctness proof for the join algorithm

notifiables is bounded (Lemma 6.1), and each propagates a notification only once, the algorithm will terminate. □

COMPLEXITY 1. *The expected number of notifications for a join is* $O(\log n)$*, where* n *is the number of sensors already in the network.*

PROOF. For a new sensor, the expected total number of notifiables is $O(\log n)$ (Lemma 6.1), and each notifiable receives notifications only from its Voronoi neighbors that are also notifiables. Since the expected number of Voronoi neighbors of a sensor is constant (Lemma 6.2), the expected number of notifications is $O(\log n)$. □

We make the observation that using the construction provided in [Devillers et al. 1992], a sensor affected by a join can recompute its local Voronoi digram in $O(\log N)$ expected time.

6.3 Sensor Failures

Sensor failures, like the deployment of new sensors, also require the modification of local Voronoi neighbors of affected sensors. The following lemma identifies the sensors affected by a single failure, and limits the size of their set of candidate replacement Voronoi neighbors.

LEMMA 6.3. *A single sensor failure affects the local Voronoi information belonging only to the Voronoi neighbors of the failed sensor. Furthermore, the set of candidates for the position of new Voronoi neighbors of each affected sensor consists solely of the Voronoi neighbors of the failed sensor.*

PROOF. The direct Voronoi neighbors of the failing sensor are clearly affected, since one of their Voronoi neighbors disappears. To see why other sensors are not affected, consider the following: let f_2 be a failed sensor and s_6 be a sensor that is not a Voronoi neighbor of f_2 (Fig. 12). Since f_2 is not a Voronoi neighbor of sensor s_6, by the definition of Voronoi diagrams, there are no sensors in the interior of any of the Delaunay circles generated by sensor s_6. Hence sensor s_6 will not acquire new Voronoi neighbors. The same argument can be used to prove that the affected sensors will have to consider as candidates for new Voronoi neighbors, only the set of affected sensors. For example, sensor f_3 must only consider sensors s_1, s_4, s_5, f_1, s_8 to replace the position left by f_2. It need not consider sensor s_6, since s_6's Voronoi neighbors are not changed by f_2's failure. □

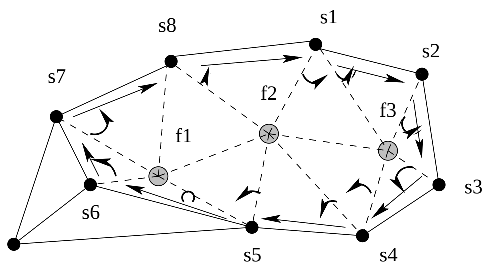


Fig. 12. Example of network where sensors $\mathtt{f}_1$, $\mathtt{f}_2$ and $\mathtt{f}_3$ fail simultaneously. The round arrows show the local decisions made by intermediate sensors, and the straight arrows show the trajectory of the DISC messages.

Departure Algorithm. We now present the actions taken when one or more sensors fail, possibly simultaneously. Each sensor S periodically sends a beacon to its Voronoi neighbors. For each Voronoi neighbor, S has a timeout value that is set whenever a beacon is received from the respective neighbor. The timeout value is an upper bound on the period of the beacon plus the round-trip time for that neighbor. If the timeout for a neighbor F expires before the respective beacon is received, S declares F failed and initiates a protocol to discover its new neighbors. For this, it creates a message of type DISC, containing the identity of S, a sequence number maintained by S and two lists. The sequence number is incremented each time a failure is detected by S. The first list, called the *failed list*, contains identities of sensors that are assumed to have failed, and initially contains only F. The second one, called the *affected list*, is empty, but eventually collects the identities of the sensors affected by the failure of the sensors in the failed list. Sensor S sends the DISC message to its first Voronoi neighbor in counterclockwise order from F.

A sensor N that receives a DISC message, is a Voronoi neighbor of F. Consequently, it adds its identifier to the affected list of the received DISC message. Sensor N then looks at the last item in the failed list, say G. Sensor N finds its first counterclockwise Voronoi neighbor starting from G, say H. If sensor H is locally considered by sensor N to be failed, and is not already in the failed list of the received DISC message, sensor N adds H to the failed list and repeats this process for the next counterclockwise Voronoi neighbor starting from H. The modified DISC message is forwarded to the first Voronoi neighbor of N, in counterclockwise order from G, that has not failed.

This procedure, similar to walking in a labyrinth with the left hand touching the wall, finds the smallest perimeter enclosing a cluster of simultaneously failed sensors, and reaches *all* the sensors affected by the cluster's failure. When the initiator S receives the DISC message that it initiated, it recomputes its Voronoi diagram and Delaunay triangulation using its local information and the information in the affected list received with the DISC message.

Example. Fig. 12 shows an example of a network in which three sensors, $\mathtt{f}_1, \mathtt{f}_2, \mathtt{f}_3$, fail simultaneously. Let us say that $\mathtt{s}_7$ detects that $\mathtt{f}_1$ has failed. $\mathtt{s}_7$ creates a DISC message containing $\mathtt{f}_1$ in the failed list, and sends it to $\mathtt{s}_8$. Before sending to $\mathtt{f}_2$, $\mathtt{s}_8$ detects that $\mathtt{f}_2$ has also failed, and adds $\mathtt{f}_2$ to the failed list. $\mathtt{s}_8$ then forwards the DISC message to $\mathtt{s}_1$. Similarly, $\mathtt{s}_1$ adds $\mathtt{f}_3$ to the failed list and forwards the message to $\mathtt{s}_2$. When $\mathtt{s}_4$ receives this DISC message, it detects that $\mathtt{f}_2$ and $\mathtt{f}_3$ are already in the failed list, so it forwards the message only to $\mathtt{s}_5$. All the intermediate sensors add their identities and positions to the affected list, so when $\mathtt{s}_7$ receives the DISC message that it has initiated, it is able to correctly recompute its Voronoi neighbors and incident Delaunay triangles.

Note. Each neighbor of a failed sensor must detect the absence of the failed sensor and generate a DISC message. The total number of messages sent is therefore on the order of the square of the number of Voronoi neighbors of the failed sensors. The number of messages could be linear, if the ring of affected sensors would have a leader. The leader could be chosen by each sensor, during its lifetime, to be its closest Voronoi neighbor. This sensor, called monitor, would then watch over its target's presence. In case of failure detection, the monitor would send two circular DISC messages, the first one collecting the information of all the affected sensors, and the second one propagating this list to all of them. However, the simultaneous failure of a sensor and its monitor would break the protocol, since not all the affected sensors would be notified.

We now prove several properties of the departure algorithm. Let $\mathtt{G_F} = (\mathtt{V_F}, \mathtt{E_F})$ be the failure graph, where $\mathtt{V_F}$ is the set of failed sensors, and $\mathtt{e} \in \mathtt{E_F}$ is an edge between two sensors $\mathtt{f_1}$ and $\mathtt{f_2}$ in $\mathtt{V_F}$ if $\mathtt{f_1}$ and $\mathtt{f_2}$ are Voronoi neighbors. For every connected component $\mathtt{C}$ in $\mathtt{G_F}$, let $\mathtt{A_C}$ be the set of Voronoi neighbors of the sensors in $\mathtt{C}$, that are not themselves in $\mathtt{C}$. That is, $\mathtt{A_C}$ is the set of sensors affected by the failure of the sensors in the connected component $\mathtt{C}$. In Fig 12, $\mathtt{f_1}$, $\mathtt{f_2}$ and $\mathtt{f_3}$ form a connected component $\mathtt{C}$ of the failure graph, and $\mathtt{A_C} = \{\mathtt{s_1}, .., \mathtt{s_8}\}$. With these definitions, a generalization of Lemma 6.3 can be easily proved.

LEMMA 6.4. *A connected component of the failure graph,* $\mathtt{C}$*, affects only the Voronoi diagram of its Voronoi neighbors,* $\mathtt{A_C}$*. Moreover, the set of candidates for the new Voronoi neighbors for each affected sensor in* $\mathtt{A_C}$ *is contained in* $\mathtt{A_C}$*.*

We prove several properties of the departure protocol.

CORRECTNESS 2. *On completion of the departure algorithm, the sensors affected by the failures will correctly compute their new Voronoi neighbors.*

PROOF. A sensor $\mathtt{S}$ that is affected by the failure of a sensor $\mathtt{F}$, part of a connected component $\mathtt{C}$ of the failure graph $\mathtt{G_F}$, will send a DISC message that will traverse all the sensors in $\mathtt{A_C}$. The message will collect information about all the traversed sensors. Using Lemma 6.4, we conclude that when $\mathtt{S}$ receives the DISC message that it initiated, it will have all the information necessary for computing the new Voronoi neighbors. □

TERMINATION 2. *The departure algorithm terminates.*

PROOF. The number of sensors that send DISC messages is equal to the number of sensors affected by the sensor failures. Each DISC message will traverse only sensors that are affected by failures. □

COMPLEXITY 2. *The total number of messages necessary to recompute the Voronoi neighbors of the sensors in the network, due to* $|V_F|$ *failures is* $O(|V_F|^2)$*.*

PROOF. The number of messages is upper bounded by the square of the number of sensors affected by the failures. The number of sensors affected by $|V_F|$ failures is $O(|V_F|)$, using Lemma 6.2. □

We make the observation that using a result of [Devillers et al. 1992], the expected cost of updating the local Voronoi diagram of a sensor affected by a sensor failure is $\mathtt{O}(\log \log \mathtt{N})$.

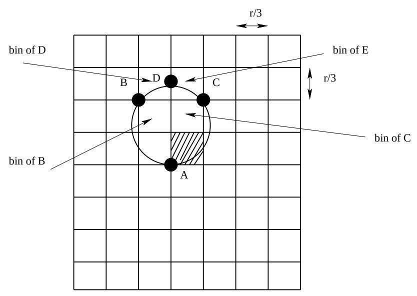


Fig. 13. Proof of Theorem 6.1 – divide the area in $r/3 \times r/3$ bins. Each bin contains at least one sensor. The hashed bin contains sensor A.

6.4 Routing to Voronoi Generators

We have assumed, until now, that the notification of the sensors affected by failures or new sensor deployments is done using direct messages between Voronoi neighbors. In other words, we have assumed that routing is done along the edges of the Delaunay triangulation. However, two Voronoi neighbors may not be within each other's transmission range, requiring a routing protocol. LAR [Ko and Vaidya 1998], DREAM [Basagni et al. 1998], and GPSR [Karp and Kung 2000] are examples of location based routing protocols that can be used for routing between non-adjacent Voronoi neighbors. However, in the following theorem, based on [Muthukrishnan and Pandurangan 2003], we provide a lower bound on the radio transmission range of sensors, that ensures that sensors that are Voronoi neighbors are likely to be within each other's transmission range.

THEOREM 6.1. *Let n be the number of sensors randomly placed in a square of area S, each sensor having a radio transmission range r. Then there exists a constant $c > 9$ such that if $r \geq \sqrt{\frac{cS\log n}{n}}$, then any two sensors s_1 and s_2 that are Voronoi neighbors are almost surely within each other's transmission range (i.e., $\Pr(\mathtt{dist}(s_1, s_2) \leq r) \rightarrow 1$ when $n \rightarrow \infty$).*

PROOF. We divide the square of area S into square bins of size $r/3$. There are $\frac{9S}{r^2}$ bins. Similar to [Muthukrishnan and Pandurangan 2003], the probability that a bin is empty after throwing n balls(sensors) in the initial square is

$$(1 - \frac{r^2}{9S})^n \leq e^{-nr^2/9S} \leq e^{-(c/9)\log n} = n^{-c/9}. \tag{2}$$

The second inequality was obtained by replacing r with $\sqrt{\frac{cS\log n}{n}}$. The expected number of empty bins is then

$$\frac{9S}{r^2}n^{-c/9} = \frac{9}{c\log n}n^{1-c/9} \leq n^{1-c/9} \tag{3}$$

The second inequality holds when $c \geq 9$. Thus, the expected number of empty bins tends to 0 when $n \rightarrow \infty$.

If every bin contains at least one sensor, it is easy to see that each sensor will be in the transmission range of all the sensors in the adjacent bins (the largest distance between two sensors in adjacent bins is $r\sqrt{5}/3 \leq r$) . To see how far two Voronoi neighbors can be, Fig. 13 shows the Delaunay circle with the largest diameter that a sensor can form with the sensors in two adjacent bins. The circle "bites" into the bins of sensors D and E. In order for D or E to be Voronoi neighbors of A, they need at least to be inside the Delaunay circle of A, B, and C, which has a diameter of $5r/6 < r$. Thus, D and E are covered by A. Thus, as the number of sensors increases a sensor tends to be in the transmission range of all its Voronoi neighbors. □

Given the transmission range of the sensors, this theorem can be easily used to find the number of sensors that need to be randomly deployed in a given square area, in order to provide, with high probability, direct connectivity between Voronoi neighbors.

Channel contention. Transmission collisions, hidden and exposed terminal problems, can be avoided using the slotted randomized philosophy of MACA, based on the transmission of Ready-To-Send (RTS) and Clear-To-Send (CTS) messages. Prior to data transmissions, neighboring nodes exchange RTS/CTS messages, sent at random intervals. Other nodes hearing RTS or CTS packets inhibit their transmission for the duration specified in the overheard packets. Randomization is used to avoid collisions between RTS packets of sending nodes. Data transmissions are then followed by acknowledgments, ensuring the correct receival of data.

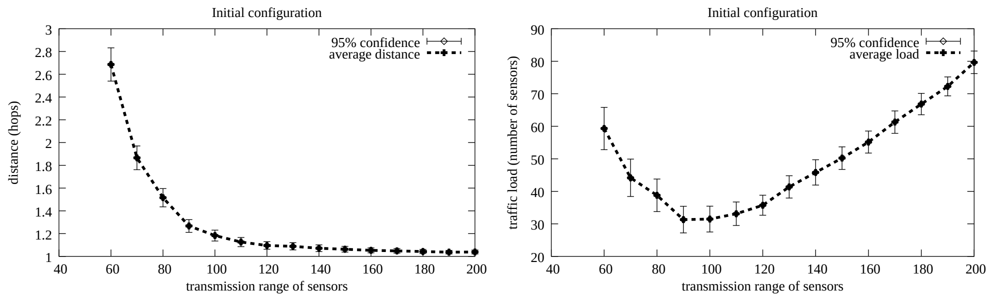
Delays. Sensors running our redundant sensor elimination and coverage-boundary detection algorithms base their decisions on the state of their Voronoi neighbors. At various stages of the protocols, sensors need the knowledge of the redundancy or coverage-boundary status of their Voronoi neighbors or their decision to stay active or sleep. Message loss becomes then an important issue.

During the initial computation of Voronoi cells (see Section 6.1), message broadcast is the predominant operation. This makes the use of acknowledgments an unscalable solution for message loss problems. However, sensors can use their knowledge of the identities of their one-hop neighbors in the following fashion. During step i of the initial computation, each sensor s maintains a list L_s of the neighbors from which it has received their step i broadcast packets, containing their network view. For this, each broadcast packet needs to contain a sequence number denoting the current step of its sender. After sending its ith broadcast packet, s starts a timer and waits for a fixed interval to receive step i broadcast packets from all its neighbors. When the timer expires, s contacts all the neighbors that are not in L_s, in a point-to-point protocol, to exchange step i information. The point-to-point protocol uses acknowledgments and retransmissions to solve message loss problems. Lack of answers for a predefined number of retries is interpreted as a failure.

During join and departure operations, message unicast is the predominant operation. Thus, acknowledgments and retransmissions can be similarly used to handle message loss problems.

7. SIMULATION RESULTS

In this section, we support our analytical results with a detailed simulation study for quantifying the overhead and correctness of our algorithms. In Section 7.1 we evaluate the number of message exchanges needed to compute the initial Voronoi cells of sensors. In Section 7.2 we compare our coverage preserving redundancy elimination algorithm, RSE, with two existing algorithms, PEAS [Ye et al. 2003] and TG [Tian and Georganas 2002]. In



(a) (b)

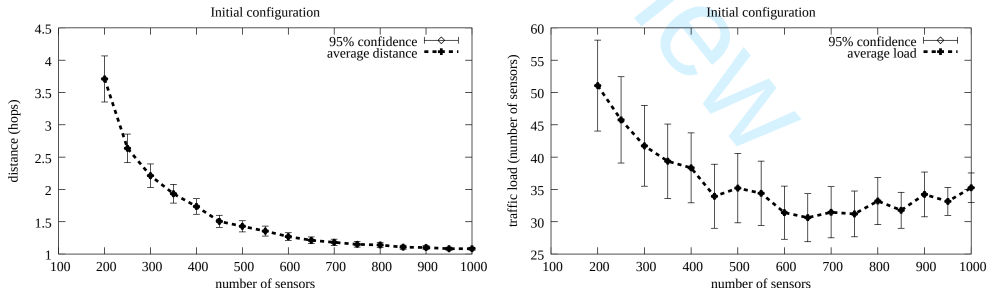
Fig. 14. The initial construction of the Voronoi diagram.

Section 7.3 we measure the communication traffic generated by join and leave operations. In Section 7.4 we investigate the ability of RSE to extend the coverage and connectivity lifetime of sensor networks.

7.1 Initial Construction of Voronoi Cells

The initial distributed computation of the Voronoi cells of sensors presented in Section 6.1 assumes a theoretical worst case where Voronoi neighbors may be k hops away, where k is the diameter of the network. In a worst case scenario, sensors have to learn of the entire network in order to accurately detect their Voronoi neighbors. In this subsection we measure the distance in number of hops between Voronoi neighbors and the traffic load imposed on the network by the initial computation of Voronoi cells, in realistic distributions of sensors.

In the first experiment we experience with transmission ranges between 60m and 200m for 700 sensors randomly distributed in a $1000 \times 1000m^2$ square area. Fig. 14(a) shows the average number of steps and the corresponding 95% confidence interval of the heart-beat algorithm presented in Section 6.1 required by a sensor to collect information about all its Voronoi neighbors. Recall that in the lth step of the heart-beat algorithm a sensor collects information about sensors that are l hops away. Therefore, Fig. 14(a) presents the distance in number of hops between a sensor and its farthest Voronoi neighbor. The average and confidence intervals are computed over all the sensors in the network. The distance in hops



(a) (b)

Fig. 15. The initial construction of the Voronoi diagram.

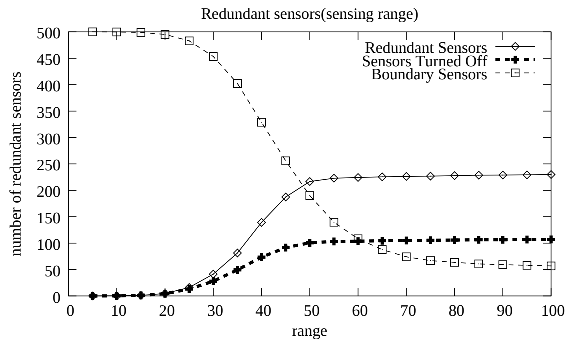


Fig. 16. Evolution of the number of boundary sensors, redundant sensors and sensors that can be turned off, for 500 sensors, when their sensing range increases from 5 to 100m.

between farthest Voronoi neighbors quickly decreases from almost 3 to very close to 1. For a transmission range of 90m, a sensor needs to contact sensors on average 1.2 hops away.

Fig. 14(b) shows the corresponding load in terms of the number of sensors whose identity and location information is collected by a sensor in order to accurately compute its Voronoi diagram. After an initial decrease to 31 for a transmission range of 90m, later the load experiences an increase to 80 for a transmission range of 200m. This is because larger transmission ranges imply larger sensor degrees (more sensors within the communication range). However, even in this case sensors have to collect only up to 11% of the total information collected when the number of hops is equal to the diameter of the network.

In the second experiment we place between 200 and 1000 sensors with a transmission range of 100m in the $10^6 m^2$ area. Fig. 15(a) shows the average and confidence interval of the distance in number of hops between a sensor and its farthest Voronoi neighbor. For higher concentrations of sensors, the distance in hops between Voronoi neighbors decreases, leading to fewer hops required to collect information about the Voronoi neighbors. For 450 sensors the average number of hops is under 1.5. The diameter of the network is however between 21 at 200 sensors and 16 at 1000 sensors. Thus, the heart-beat algorithm of Section 6.1 can safely stop at five times less steps than the network diameter.

Fig. 15(b) shows the corresponding average number of sensors whose information is collected in the heart-beat process by a sensor. While the load experiences an initial decrease with higher sensor densities, it later (starting at around 700 sensors) exhibits a mild increase. This is because higher sensor densities decrease the number of steps of the heart-beat algorithm (see Fig. 15(a)) but also increase the number of sensors placed in each other's communication range (sensor degree). For example, for 1000 sensors the Voronoi neighbors of a sensor are very likely 1 hop away but the average number of transmission neighbors is between 32 and 37. However, this represents 4% of the information collected by the heart-beat protocol when the number of steps employed is the diameter of the network.

7.2 Detecting Coverage-Boundary and Redundant Sensors

In the first experiment we investigate the performance of RSE under three metrics: the number of coverage-boundary sensors, the total number of locally detected redundant sensors, and the total number of sensors that can be simultaneously turned off. For all the experiments in this section, we randomly generate 10 different sensor network configurations, and present only the average results. We measure the evolution of wireless sensor networks of 500 sensors, whose sensing range is increased from 5 to 100m. Fig. 16 shows the results. When the range is smaller than 20, all the metrics have extreme values. However, for larger values, the number of boundary sensors decreases drastically to almost 10% of the total number of sensors. The number of simultaneous turn-offs however, saturates quickly at 20%, when the sensing range is around 50. Consequently, a relatively small sensing range is enough to detect most of the possible simultaneous sensor turn-offs.

The following experiment compares the performance of RSE with the PEAS [Ye et al. 2003] and TG [Tian and Georganas 2002] algorithms. The sensors running RSE, PEAS and TG have a sensing range of 50m and we increase the total number of sensors from 150 to 500. As specified in [Tian and Georganas 2002], the sensors running TG have the ability of knowing the status (active or sleeping) of their neighbors, before making their decision. Similar to the experiments of [Ye et al. 2003], for PEAS we set the probing range of sensors to be one third of their sensing range. We evaluate the performance of two variants of RSE, one used in conjunction with Luby's [Luby 1985] MIS approximation algorithm and one where sensors know the state of their neighbors before deciding their own state.

Fig. 17 shows the performance of RSE and TG, compared with the total number of redundant sensors. As expected, the total number of redundant sensors is always larger than the number of possible simultaneous turn-offs, both for RSE and TG. In addition, both variants of RSE consistently outperform PEAS and TG, by being able to simultaneously turn off more sensors. Note that both RSE and TG preserve the coverage of the network, however, TG makes more conservative decisions. This is because sensors running TG decide their redundancy on a subset of their transmission neighbors and the sponsoring algorithm is only an approximation. In our experiments PEAS is able to turn off more sensors than TG, however, it does not ensure complete coverage. By using the Voronoi neighbors of sensors to decide their redundancy, RSE accurately detects *all* the redundant sensors. However, RSE used in conjunction with Luby's [Luby 1985] approximation algorithm, discovers a subset of the inactive sensors discovered when sensors use the state of their neighbors in their decision.

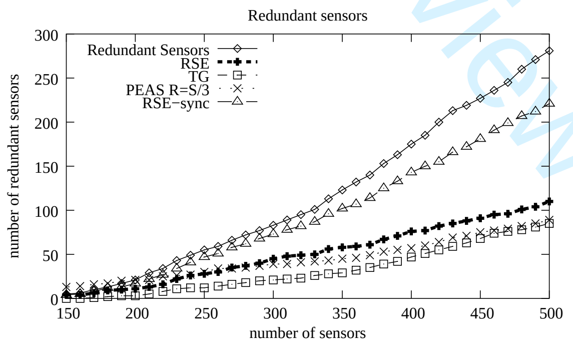
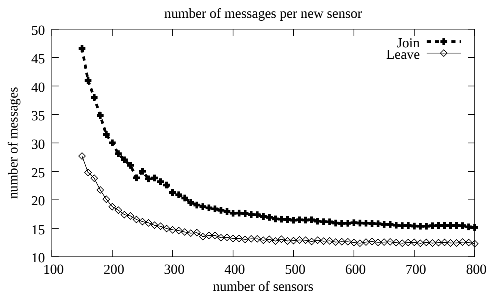
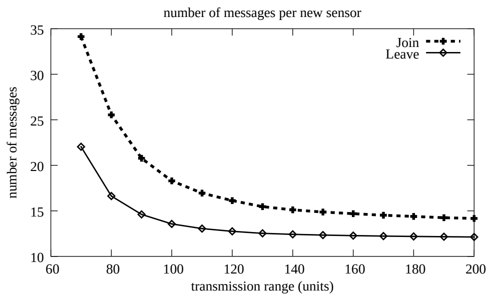


Fig. 17. Node scalability comparison between RSE, PEAS and TG. The number of sensors with a sensing range of 50m grows from 150 to 500.



(a)



(b)

Fig. 18. Plots showing the total number of messages required for the deployment of new sensors.

7.3 Network Load

We investigate the traffic generated in the network by insertions of new sensors and failure of existing ones. The messages generated are necessary to recompute the local Voronoi diagrams, and with it, the coverage-boundary and the redundant sensor information. As before, we assume that all the sensors have similar capabilities, which for these experiments consist of identical radio transmission ranges. Note that the experiments in Section 7.2 only consider the sensing range.

Each time a new sensor `ns` joins the network, new paths need to be generated between the new sensor and its new Voronoi neighbors. If the new sensor is not in the transmission range of one of its Voronoi neighbors, s_1, the path from `ns` to s_1 is chosen to be the concatenation of the path between `ns` and the sensor closest to it, with the path between its closest neighbor and s_1, since these paths already exist. Even though this is not necessarily the shortest path between `ns` and s_1, our measurements showed that the communication overhead introduced by using this path is very small. Our experiments measure not only the number of messages required to notify all the sensors affected by the new sensor, but also the number of messages necessary for the affected sensors to contact the new sensor.

In the first experiment, we initially place between 150 and 800 sensors in a region of size $1000 \times 1000m^2$. All the sensors have the same radio transmission range, of 115m. To investigate the performance of the join algorithm, for each value of the number of sensors initially deployed, we generate 10 random network configurations, and for each configuration we insert a new sensor at 150 random positions. Similarly, for sensor failures, for each of the 10 random networks we randomly select 150 sensors to fail. We present only the average values over 1500 measurements.

Fig. 18(a) shows the results of this experiment. The average number of messages transmitted for each join/leave operation, decreases abruptly with the increase in sensor density, and saturates at around 400 sensors. Even though a higher sensor density implies a larger number of Voronoi neighbors per sensor, increasing the number of sensors that need to be notified, it also implies shorter distances between Voronoi neighbors, thus fewer routing messages. Therefore, denser networks simplify the task of locally updating the Voronoi and coverage information.

In the second experiment, we place 500 sensors in the same square and increase the transmission range from 50 to 200. Similar to the previous experiment, the values reported

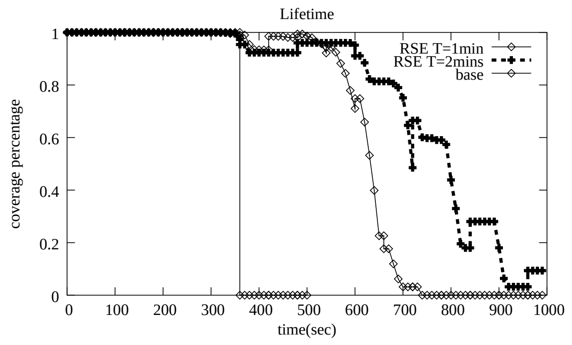


Fig. 19. Coverage lifetime of a sensor network of 500 nodes.

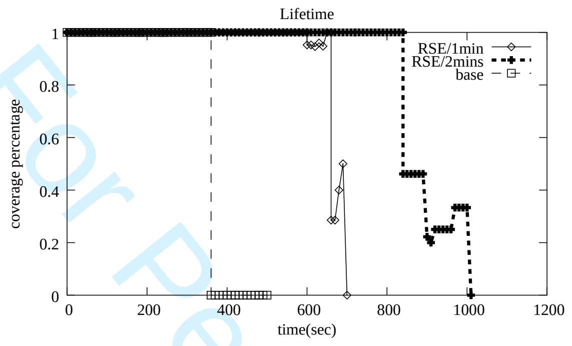


Fig. 20. Evolution of connectivity of a sensor network of 500 nodes.

are averaged over 1500 measurements. Fig. 18(b) presents the results. The average number of messages required per update decreases quickly and saturates at a transmission range of around 120. Energy-wise, there exists a trade-off between the transmission range employed and the total number of messages required per update. Since the energy required per transmission increases super-linearly with the distance, shorter transmission ranges are preferred.

7.4 Network Lifetime

In this section we investigate the ability of RSE to extend the coverage and connectivity lifetime of sensor networks. We randomly place 500 nodes with a sensing range of 50m, in a $500 \times 500m^2$ network deployment area. The battery of each node has an initial charge of 300 Joule. Following the energy model used in [Wang et al. 2003], we set the power consumption of transmit (Tx) to 1400mW, receive to 1000mW, idle to 830mW and sleep to 130mW. The ratio between the transmission range and sensing range of nodes is 2.25. The coverage of the network is evaluated every 10 seconds.

Sleeping nodes periodically re-activate (see Section 4.4. In order to balance the energy consumption over the network we investigate RSE for two global wake up period values, of 60 and 120 seconds. When nodes are re-activated, the redundancy information is recomputed by each node. From the conflicting redundant nodes, able to leave blind points (see Section 4.3), the nodes with the lowest battery power are elected to sleep. This policy is chosen in order to balance the power consumption over the network, effectively extending the coverage lifetime of the network.

Fig. 19 shows the evolution of the network's coverage in time, both for RSE and a base case where all nodes are continuously active. In the base case all nodes run out of battery after 360s, when the coverage of the network drops from 100% to 0. However, in the case of RSE, the coverage is maintained above 90% until 560s for a node re-activation period of 60s and until 610s for a period of 120s.

We evaluate the ability of RSE to extend the connectivity of a sensor network by placing a central collection point in the top leftmost corner of the $500 \times 500m^2$ network deployment area. We measure connectivity as the percentage of nodes able to route to the collection point. Fig. 20 compares RSE with the base case where nodes are continuously active. While in the base case the entire network fails at 360s, using RSE with a re-activation period of 60s enables sensors to maintain a connectivity level above 90% for 660s. For a 120s re-activation period, a similar connectivity is maintained for 840s. In conclusion, the use of RSE significantly extends the lifetime of the network, both in terms of coverage and connectivity.

8. CONCLUSIONS

In this paper, we have studied the problem of coverage-preserving, energy-efficient redundancy elimination for extending a network's lifetime, and the related coverage-boundary problem. We have reduced both problems to the computation of Voronoi diagrams and showed how to solve them using only local information. We have provided distributed and localized algorithms that allow sensors to update their view of the solution in cases of sensor failures and new sensor deployments. We have proved the correctness and termination properties of these algorithms. Our simulations show that the algorithms are efficient and scale well with the number of sensors.

9. ACKNOWLEDGMENTS

The authors would like to acknowledge the support of National Science Foundation through grants CCF 0325227, CCF 0444285, and CMS 0443148. The authors would also like to thank the anonymous reviewers for their helpful comments in revising the paper.

REFERENCES

BASAGNI, S., CHLAMTAC, I., SYROTIUK, V. R., AND WOODWARD, B. A. 1998. A distance routing effect algorithm for mobility (DREAM). In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. ACM Press, New York, NY, USA, 76–84.

CARDEI, M. AND WU, J. 2004. *Coverage in Wireless Sensor Networks*. Handbook of Sensor Networks. CRC Press.

CARLE, J. AND SIMPLOT-RYL, D. 2004. Energy effi cient area monitoring for sensor networks. *IEEE Computer 37,* 2 (February), 40–46.

CHOSET, H. 2001. Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artifi cial Intelligence 31*, 113–126.

DAI, F. AND WU, J. 2003. Distributed dominant pruning in ad hoc wireless networks. In *ICC '03: Proceedings of IEEE International Conference on Communications*.

DEVILLERS, O., MEISER, S., AND TEILLAUD, M. 1992. Fully dynamic delaunay triangulation in logarithmic expected time per operation. *Comput. Geom. Theory Appl. 2,* 2, 55–80.

GUPTA, H., DAS, S. R., AND GU, Q. 2003. Connected sensor cover: self-organization of sensor networks for effi cient query execution. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. ACM Press, 189–200.

HAAS, Z. May 1997. On the relaying capability of the reconfigurable wireless networks. In *VTC: Proceedings of 47th IEEE Vehicular Technology Conference*. Vol. 2. 1148–1152.

HU, L. 1993. Topology control for multihop packet radio networks. *IEEE Transactions on Communications 41*, 10 (October), 1474–1481.

HUANG, C.-F. AND TSENG, Y.-C. 2003. The coverage problem in a wireless sensor network. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. ACM Press, 115–121.

JIANG, J. AND DOU, W. 2004. A coverage-preserving density control algorithm for wireless sensor networks. In *Proceedings of ADHOC-NOW*.

KARP, B. AND KUNG, H. T. 2000. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM Press, New York, NY, USA, 243–254.

KO, Y.-B. AND VAIDYA, N. H. 1998. Location-aided routing (LAR) in mobile ad hoc networks. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. ACM Press, New York, NY, USA, 66–75.

LUBY, M. 1985. A simple parallel algorithm for the maximal independent set problem. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*. ACM Press, 1–10.

MEGUERDICHIAN, S., KOUSHANFAR, F., QU, G., AND POTKONJAK, M. 2001. Exposure in wireless adhoc sensor networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM Press, New York, NY, USA, 139–150.

MUTHUKRISHNAN, S. AND PANDURANGAN, G. 2003. The bin-covering technique for thresholding random geometric graph properties. Tech. Rep. 2003-39, DIMACS. November.

SHAKKOTTAI, S., SRIKANT, R., AND SHROFF, N. 2003. Unreliable sensor grids: Coverage, connectivity and diameter. In *Proceedings of IEEE INFOCOM*. Vol. 2. 1073–1083.

SLIJEPCEVIC, S. AND POTKONJAK, M. 2001. Power efficient organization of wireless sensor networks. In *ICC '01: Proceedings of IEEE International Conference on Communications*.

STOJMENOVIC, I. 1999. Voronoi diagram and convex hull based geocasting and routing in wireless networks. Tech. Rep. TR-99-11, University of Ottawa. December.

TIAN, D. AND GEORGANAS, N. D. 2002. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. ACM Press, 32–41.

WANG, X., XING, G., ZHANG, Y., LU, C., PLESS, R., AND GILL, C. 2003. Integrated coverage and connectivity configuration in wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM Press, 28–39.

YE, F., ZHONG, G., LU, S., AND ZHANG, L. 2003. PEAS:a robust energy conserving protocol for long-lived sensor networks. In *ICDCS '03: Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems*.

ZHANG, H. AND HOU, J. 2004. Maintaining coverage and connectivity in large sensor networks. In *International Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless and Peer-to-Peer Networks*.