Detecting Cuts in Sensor Networks*

Nisheeth Shrivastava Subhash Suri Csaba D. Tóth

Department of Computer Science,
University of California,
Santa Barbara, CA 93106, USA.
{nisheeth, suri}@cs.ucsb.edu

Department of Mathematics,
Room 2-336, MIT,
Cambridge, MA 02139, USA.
toth@math.mit.edu

***Abstract*— We propose a low overhead scheme for detecting a network partition or cut in a sensor network. Consider a network S of n sensors, modeled as points in a two-dimensional plane. An ε-*cut*, for any $0 < \varepsilon < 1$, is a linear separation of εn nodes in S from a distinguished node, the *base station*. We show that the base station can detect whenever an ε-cut occurs by monitoring the status of just $O(\frac{1}{\varepsilon})$ nodes in the network. Our scheme is deterministic and it is free of false positives: no reported cut has size smaller than $\frac{1}{2}\varepsilon n$. Besides this combinatorial result, we also propose efficient algorithms for finding the $O(\frac{1}{\varepsilon})$ nodes that should act as *sentinels*, and report on our simulation results, comparing the sentinel algorithm with two natural schemes based on sampling.**

I. Introduction

Wireless sensor networks (WSN) have emerged as an important new technology for instrumenting and observing the physical world. The basic building block of these networks is a tiny microprocessor integrated with one or more MEMS (micro-electromechanical system) sensors, actuators, and a wireless transceiver. These devices can be embedded or scattered in large quantities in a physical space, where they self-organize into an *ad hoc* multi-hop wireless network, allowing us to observe and monitor the world at an unprecedented spatial and temporal resolution. A rich variety of scientific, commercial, and military applications [7], [11], [25], [32] has been proposed for sensor networks, and many experimental prototypes are under development in academia and industry. Realizing the full potential of the sensor networks, however, requires solving several challenging research problems. Many of these challenges stem from two major limitations of the sensor nodes: low power and low bandwidth. Consequently, a number of proposals have been made for improving the data collection and information processing in sensor networks, including power-aware routing and scheduling [16], [24], [27], in-network aggregation [15], [23], [28], query processing [13], data storage management [12], etc.

In this paper, we address a different kind of challenge for sensor networks, which does not seem to have received adequate attention. *How to monitor the sensor network itself, and how to detect when the network has suffered a significant "cut"?* After all, if sensor networks are to act as our remote "eyes and ears," then we need to ensure that any significant failure (natural or adversarial) suffered by the network is promptly and efficiently detected. Tracking the operational health of the infrastructure is important in any communication network, but it is especially important in sensor networks due to their unique characteristics, and the need to perform this duty with very little overhead.

In our view, *power efficiency*, *scalability*, and absence of *false positives* are the three most important considerations for a scheme to detect network cuts. Because a sensor network's lifetime is largely determined by how well it conserves power, solutions where all sensors are continuously monitored are both inefficient and unscalable. Because sensor networks can vary in size from few hundred nodes to hundreds of thousands, it is also desirable to design schemes that are highly scalable, so that the task of cut detection does not end up consuming a large part of the network resources. Finally, because many sensor network applications envision unmanned and remote deployment, failure detection schemes that yield *false positive*, or false negatives, are highly undesirable. With this motivation, we now describe our problem setting.

A. Cuts in Sensor Networks

Consider a set S of n sensors, which are modeled as points in the two-dimensional plane. (More generally, we can assume that the sensors lie on a surface or terrain that is topologically equivalent to the plane.) An adversary can make a *linear cut* through the sensor network, disabling all the sensors on one side of the line; the base station is assumed to lie on the other (safe) side. Formally, given a line L, let L^- and L^+ denote the two half-planes defined by L, and let $L^-(S)$ and $L^+(S)$ denote the subset of sensors that lie in these half-planes. We will adopt the convention that the linear cut induced by L disables all the sensors in $L^-(S)$. Alternatively, the adversary can disrupt the communication so that sensors on one side of the line cannot communicate with sensors on the other side, including the base station. These two formulations are equivalent for our purpose. There are other natural forms of cuts, such as circular cuts, rectangular cuts, polygonal cuts. We will briefly discuss them in Section VI.

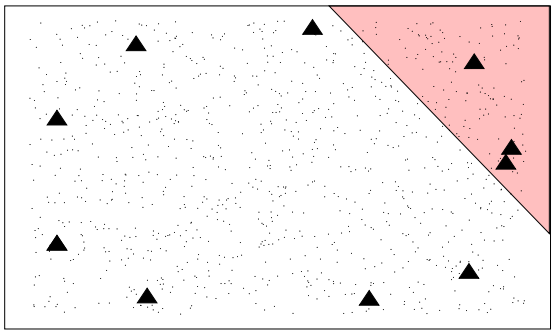

Fig. 1. 1000 sensor nodes, distributed uniformly at random, and a linear cut. The sentinel nodes for $\varepsilon = 0.05$ are shown as triangles.

We call a linear cut an ε-*cut* if at least ε fraction of the sensors are cut off, where $0 < \varepsilon < 1$ is a user-specified parameter. Formally, L is an ε-cut if $|L^-(S)| \geq \varepsilon|S|$. Our primary focus in this paper

* Work on this paper was supported in part by the NSF ITR grant IIS-0121562 and by the Institute for Collaborative Biotechnologies through grant DAAD19-03-D-0004 from the U.S. Army Research Office.

is to develop a low-overhead scheme for detecting ε-cuts in sensor networks.

Our scheme for detecting ε-cuts will choose a small subset of sensors, which act as *sentinels*. Each sentinel will communicate with the base station at a regular time interval. We assume that the base station is not attacked, and it always lies in the safe halfplane L^+. A communication failure from a sentinel is taken to mean that the sentinel has been cut off. Our problem now becomes: can one choose a small number of sensor nodes as sentinels so that (1) every ε-cut can be detected based solely on the live/dead status of sentinels, and (2) the algorithm does not report false positives. In Figure 1, we show a collection of 1000 sensor nodes, distributed uniformly at random, and its sentinel set for $\varepsilon = 0.05$.

Before describing our results, we first briefly discuss why we chose ε-cuts as our definition, why avoiding false positives is challenging, and why the detection scheme requires an approximation slack.

B. ε-Cuts

The ε-cuts are motivated both by practical and theoretical concerns. It makes practical sense to treat failures as significant only when a *fraction* of the network is cut. It may be tempting to ask for schemes that detect failure of a fixed (user-specified) *number* of sensors, regardless of the network size. However, no efficient and scalable solution is theoretically possible in this case, as the following simple example shows. Imagine n sensors arranged in a circle, and suppose we want to detect cuts of size m. Then, at least one sensor for every m consecutive sensors must be chosen as a sentinel, which scales very poorly with the network size.

C. False Positives

By monitoring sufficiently many randomly chosen sensors, one can detect all ε-cuts with high probability. For instance, a random sample of size $O(\frac{1}{\varepsilon}\log\frac{1}{\delta\varepsilon})$ is sufficient to catch any ε-cut with probability at least $1-\delta$ [21], [22]. The algorithm simply declares an ε-cut whenever at least one of the chosen sensors fails. Unfortunately, this simple scheme suffers from the false positives problem. Many cuts reported by this algorithm, however, are false positives, where the size of the failed network can be arbitrarily smaller than εn. Indeed, if one of the random samples happens to lie on the boundary of the sensor field, then it can cause an alarm even if a single sensor is cut off.

A more sophisticated form of sampling can effectively eliminate false positives, but at the expense of a very large number of sentinels. In particular, the concept of *ε-approximation* can be used to distinguish between all cuts larger than εn and those smaller than, say, $\frac{1}{2}\varepsilon n$. But an ε-approximation requires $\Theta(\frac{1}{\varepsilon^2}\log\frac{1}{\epsilon\delta})$ sentinel nodes. A simple calculation, including the actual constants involved, however, shows that even for modest values of $\varepsilon = 0.1$ and $\delta = 0.05$, the size of the sentinel set is at least $10,000$! Thus, random sampling based schemes are infeasible, due to false positives or due to unscalably large size.

D. The Need for Approximation

Finally, we point out that we need to allow some approximation slack between an ε-cut and a false positive. If we demand that all ε-cuts be reported, and *no cut of size smaller than εn be reported*, then no scalable solution exists. Consider, once again, the setting of n sensors arranged in a circle. We claim that in this case *every other sensor* must act as a sentinel. Otherwise, suppose that neither of the two consecutive sensors a and b is a sentinel. Now consider two cuts: one, where a is not cut but b and the $\varepsilon n-2$ predecessors of b are cut; and second where the cut includes a, b and the $\varepsilon n-2$ predecessors of b. These two cuts differ only in a but neither a nor b is a sentinel, so the algorithm cannot distinguish between the two cuts. But, by the strict ε-cut definition, the former is not a reportable cut, while the latter is. We adopt the standard convention and set a *lower bound* on the size of any cut reported. Specifically, our algorithm successfully will report every ε-cut and, at the same time, every reported cut comes with a guarantee: *at least $\varepsilon/2$ fraction of the network must have been cut.* The fraction $1/2$ is chosen to simplify our analysis, and it can be replaced by any user specified parameter.

E. Our Contribution

Our paper makes three contributions. First, we prove a combinatorial result: for any positive real number $\varepsilon < 1$, there exists a sentinel set of size $O(\frac{1}{\varepsilon})$, which can detect every ε-cut. Moreover, every cut reported as an ε-cut includes at least $\frac{1}{2}\varepsilon n$ failed sensors. A key point to note is that the size of the sentinel set depends only on the parameter ε, and *not on* n, the size of the sensor network. Thus, our construction is highly scalable. It is easy to see that the sentinel size of $O(\frac{1}{\varepsilon})$ is asymptotically optimal.

Second, we describe two efficient algorithms, a deterministic algorithm for constructing a minimal sentinel set and a faster randomized algorithm for computing a sentinel set of size $O(\frac{1}{\varepsilon})$. All our algorithms are centralized, because we envision the entire process taking place at a base station.

Finally, we implemented our scheme and ran simulations using a variety of synthetic sensor network models. In our simulations, we found that the size of sentinel set was always significantly smaller than the worst-case bound of Theorem 3.1. We also compared our sentinel algorithm with two variants of sampling-based schemes: random sampling, and radial sampling. Our experiments show that even for rather well-behaved sensor distributions, these methods produce a significant number of false positives and false negatives.

F. Related Work

The problem of network partition in sensor networks has been raised in several papers, but it appears not to have been investigated formally. In their survey paper [6], Chong and Kumar raise this problem with a security focus: sensor networks may operate in hostile environments and schemes to detect tampering should be built into the design. In [5], Cerpa and Estrin propose schemes for self-configuring sensor network topologies. They mention network partition as an important problem for which "complementary system mechanisms will be needed," but leave that as a future research direction. In [19], Lifton, Broxton and Paradiso consider a network disconnection problem, but with a very different focus: the nodes are *cooperative*. For instance, sensor nodes with low battery power communicate with the network to determine whether the network will be partitioned if they failed. This is also an experimental paper, with no formal algorithm analysis.

Our research is inspired by some recent work by Kleinberg et al. [17], [18] on detecting failures in a *wired* network. In Kleinberg's setting, the network is modeled as an undirected graph on n nodes; an adversary can destroy up to k edges (or vertices); and the detection algorithm installs a set of detection agents (equivalent to our sentinels) that engage in *pairwise* communication. They are interested in detecting when the graph is disconnected into two subsets, each of size at least εn. The main result in [17], [18] is that every graph has a (ε, k)-detection set (number of sentinels) of size $O(k^3\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}+\frac{1}{\varepsilon}\log\frac{1}{\delta})$, which can detect an ε-cut with probability $1-\delta$. There are three important differences between these results

and our results. One, because of the inherent geometric structure of sensor networks, linear or other geometric partitions are more natural than the k independent edge failures. Second, due to the geometric structure of our problem, just $O(\frac{1}{\varepsilon})$ sentinels suffice in our case, while in Kleinberg's case, they require a much larger set of monitoring nodes, as well as pairwise communication between those nodes. Finally, our scheme yields no *false positives*, while the algorithms proposed in [17], [18] suffer from false positives: every ε-cut is detected, but not every cut detected in an ε-cut. Indeed, they do not provide any lower bound on the size of the cut.

The network partition problem has connections with the theory of VC-dimension [31] and ε-nets [22]. Unfortunately, ε-nets provide only *1-sided* guarantees: $|R \cap N| > 0$ does not imply that $|R \cap S| \geq \varepsilon n$. Thus, ε-nets make poor sentinels: they raise too many false alarms.

An ε-approximation is a stronger form of sample. Given a set of n points S, the ε-approximation intersects any range in (roughly) the same proportion as it intersects S, and so it would make a nice sentinel set. Unfortunately, the ε-approximation usually requires too many points (sentinels): it has size $O(\frac{d}{\varepsilon^2}\log\frac{1}{\varepsilon\delta})$. As mentioned earlier, with constant factors included, even with modest values of $\varepsilon = 0.1$ and $\delta = 0.05$, the size of ε-approximation exceeds $10,000$!

In [4], the notion of *sensitive* ε-approximations is introduced as a generalization of ε-approximation. However, even sensitive approximations have $O(\frac{1}{\varepsilon^{4/3}}\log\frac{1}{\delta\varepsilon})$ size for linear cuts, whereas our sentinel sets have optimal $O(\frac{1}{\varepsilon})$ size. In addition, the deterministic construction of sensitive approximations is quite involved.

II. Geometric Preliminaries

The network topology and the communication protocol are not directly relevant to our result. We simply assume that the sensor network is connected and that every sensor is able to communicate with a *base station* through multi-hop routing, as long as a valid communication path exists. We also assume that the location of every sensor is available to the base station. A set S of n sensors scattered in a terrain is modeled as a set of n points in the plane (ignoring the altitude of each sensor). Our problem of monitoring the integrity of the sensor field is best studied in a geometric setting.

A. Sentinel Sets

We wish to detect if the sensor network has suffered a linear cut of size at least εn. We do so by monitoring a small subset of sensor nodes, called the *sentinel set* W. An adversary can introduce a linear cut, by disabling all sensors lying on the right side L^- of a line L. It is assumed that the base station lies on the safe side, L^+. We call a directed line L an ε-cut if its halfplane L^- contains at least ε fraction of all the sensors; formally, L is a ε-cut if $|L^-(S)| \geq \varepsilon n$.

We would like to point out that the base station has *no explicit information about the line* L. It only learns the *signature vector* $\sigma(W)$ that represents the alive or dead status of the sentinel sensors; that is, $\sigma(W)$ is a binary vector of length $|W|$. Our goal is to compute a sentinel set of *small* size that can detect every ε-cut correctly, but never reports a cut of size less than $c\varepsilon n$, for some constant $c < 1$. For ease of presentation, we choose $c = 1/2$ in this paper, but all our results generalize to any fixed value of c, $0 < c < 1$. With this motivation, we have the following definition.

Definition 2.1: Suppose S is a set of n sensors, and $\varepsilon > 0$ is a user-specified parameter. A subset of sensors W, where $W \subseteq S$, is called an *ε-sentinel set* if, for any linear cut L, we can decide whether L is an ε-cut or that L is a smaller than $\frac{\varepsilon}{2}$-cut, by observing only the signature $\sigma(W)$.

Thus, the signature of an ε-sentinel set will let us detect every cut of size at least εn. Furthermore, we would also know which cuts have size less than $\frac{\varepsilon}{2}$. In the gray area where the size of the cut is between $\frac{\varepsilon}{2}n$ and εn, the algorithm is free to go either way: report it or ignore it.

Because the sentinels are points (sensor locations) in the plane, there are precisely $m(m-1) = O(m^2)$ *combinatorially distinct* signatures $\sigma(W)$ that correspond to linear cuts. Because the base station cannot distinguish linear cuts that yield identical signatures, the family of lines corresponding to a specific signature $\sigma(W)$ cannot contain an ε-cut and a less-than $(\varepsilon/2)$-cut simultaneously. This insight suggests that the sentinel problem becomes a *separation* problem in a transformed space, which we describe next.

B. A Duality Transform

We use a point-line duality of the Euclidean plane. The dual of a point $p(a,b)$ is the line $p^* : y = ax - b$ and, conversely, the dual of a (non-vertical) line $L : y = ax - b$ is the point $L^* : (a,b)$. The vertical lines can be handled by using a slightly more involved *projective* duality. Instead, we use the simpler transform here, and assume that all sensor nodes have distinct x-coordinates. In this way, for every vertical line, there is a slightly perturbed non-vertical line with the same signature $\sigma(S)$. It can be easily checked that the duality transform inverts the *above-below relation*: if point p lies above (resp. below) line L, then the dual line p^* is below (resp. above) the dual point L^*. A similar transform is used in Liu et. al. [20] for tracking a linear shadow over a sensornet.

The duality transform maps our set S of n sensors into a set S^* of n lines. Conversely, a linear cut L is transformed into a point L^*. We point out that the *orientation* of L is lost in the duality. We assume throughout that the line L is oriented so that the right halfplane L^- lies *above* the line L. Thus, in the linear cut induced by L, all the sensors above L are cut off. A similar argument holds when the halfplane L^- lies below L. Thus, to cover both cases, we will consider cuts where either εn points lie below the cut, or εn points lie above the cut.

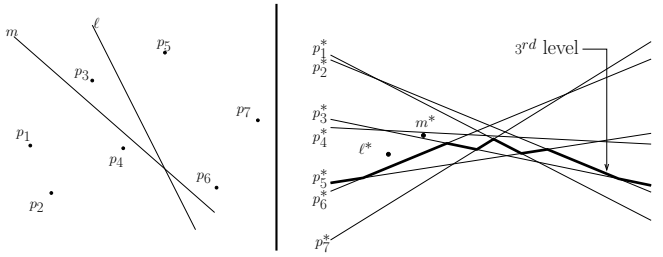

Fig. 2. A set of 7 points (left), and the corresponding dual arrangement (right). The lines ℓ and m dualize to points ℓ^* and m^*, respectively. The duality transform inverts the above-below relation; for instance, point p_5 lies above line ℓ, and its dual line p_5^* lies below the dual point ℓ^*. The 3-level of the arrangement is shown in bold.

C. Line Arrangements and Levels

The set of n lines S^* in the dual plane form a *line arrangement*, denoted $H(S^*)$. The arrangement is a dissection of the plane into convex polygons, some of which are unbounded. The *vertices* of the arrangement are the intersection points between pairs of lines; the *edges* of the arrangement are the line segments between two consecutive vertices on a line. An arrangement of n line has at most $n(n-1)/2$ vertices and at most $n(n+1)$ edges. For technical

simplification, we assume that no more than 2 lines pass through a vertex.

The set of edges in the arrangement that lie above exactly $k-1$ other lines form an x-monotone polygonal curve. This curve is called the *k-level* of the arrangement. (A point (a,b) is above k lines if the ray $\{(a,y) : y < b\}$ crosses exactly k lines of the set S^*.) The 1-level, for instance, is the *lower envelope* of the arrangement. A k-level bends at every vertex along its way. See Figure 2 for illustration.

Consider a sentinel set $W \subseteq S$, and some linear cut L. The signature $\sigma(W)$ of W with respect to L tells us which sensors are *below* the line L and which ones are *above* L. In the transformed plane, this tells us which lines of W^* pass above the dual point L^*, and which ones pass below L^*. In order for W to be an ε-sentinel, we should be able to decide *for any point* L^* whether at least εn lines of S^* pass below L^* or fewer than $\varepsilon n/2$ lines of S^* pass below L^*, based solely on the signature $\sigma(W)$. Thus, W is an ε-sentinel if for any point L^* we can tell if L^* lies above the (εn)-level or below the $(\varepsilon n/2)$-level of $H(S^*)$ based on the location of the point L^* *in the arrangement formed by* W^*. The important point here is that we want to determine the location of a point in the arrangement of S^*, but do so by looking only at the arrangement formed by the much smaller set W^*.

D. Minimum Link Separators in Arrangements

Given two disjoint simple polygonal curves, γ_1 and γ_2, in the plane, a *separator* ϱ is a polygonal curve that partitions the plane into two parts such that γ_1 and γ_2 lie on opposite sides of ϱ. A *minimum link separator* for γ_1 and γ_2 is such a separator with the minimum number of vertices (i.e., bends).

A minimum link separator ϱ between the εn and the $\varepsilon n/2$ levels of the arrangement $H(S^*)$ can efficiently distinguish ε-cuts from the less than $(\varepsilon/2)$-cuts. Specifically, if L^* lies below ρ then L is certainly not an ε-cut; and if L^* lies above ρ then L is surely an $(\varepsilon/2)$-cut. A minimum link separator, in general, is free to use any lines. However, in our setting, this separator will be used to form a sentinel set, and therefore we *must use only the lines of* S^* in the minimum link separator. (Indeed, the previously known algorithms for constructing separators did not require the separator be part of the arrangement [10], [21].) Therefore, in the following discussion, we define the *separator in an arrangement* H to be a polygonal curve that only uses the edges of the arrangement H.

III. Combinatorics of Sentinel Sets

In this section, we prove our first main result and show that there is a separator with $O(\frac{1}{\varepsilon})$ links. The sentinel set is formed by choosing the points whose dual lines contain these links. We next show that based on the signature $\sigma(W)$ of this set, we can determine in $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ time if there is a linear cut of size at least εn, or that the cut is smaller than $\varepsilon n/2$. Since we do not know the orientation of the cut, we make two sentinel sets: one for the separator of top levels and and the other for bottom levels. If the signature of any one of them indicates a cut of size at least εn, then we declare there is an ε-cut in the network.

A. Existence of a small sentinel set

The following theorem states our main combinatorial result.

Theorem 3.1: In an arrangement of n lines in the plane, there is always a separator of size $O(\frac{1}{\varepsilon})$ between levels εn and $\frac{1}{2}\varepsilon n$.

The proof of the theorem relies on a couple of technical lemmas about separators and levels. Consider an a-level and a b-level in the arrangement of n lines in the plane, where $0 \le a < b \le n$. For any integer k, where $a \le k \le b$, we define a specific x-monotone *zig-zag* separator, denoted $z(k)$, between the a-level and the b-level. Informally, the separator $z(k)$ starts with the leftmost segment of the k-level, follows that line until it runs into either the a-level or the b-level at a vertex, at which point it "reflects" and follows the other line determining that vertex. Thus, the path $z(k)$ zig-zags between the a-level and the b-level. We note that $z(k)$ is an x-monotone path, it only uses the lines of the arrangement $H(S^*)$, and all its "bends" are at convex vertices of the a-level or at reflex vertices of the b-level; indeed, it can also have several consecutive bends along the same level, on the reflex vertices of a-level and the convex vertices of b-level. See Figure 3 for an example.

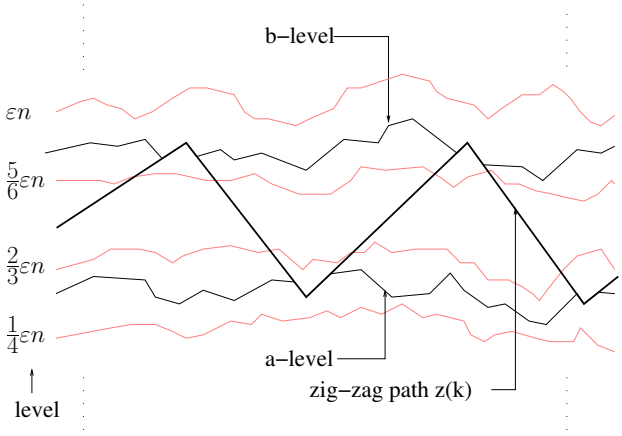

Fig. 3. A zig-zag path between two levels.

Altogether there are $(b-a+1)$ such paths between the a-level and the b-level, one for each value of k between a and b. The following lemma notes that these paths are pairwise edge-disjoint.

Lemma 3.2: The $(b-a+1)$ zig-zag separators between the a-level and the b-level are pairwise non-overlapping; that is, they can only intersect at vertices.

Proof: The leftmost edges of the zig-zag paths are pairwise disjoint by definition. If two zig-zag paths, say, $z(k)$ and $z(k')$ meet at a vertex v and they reach v from the left on different lines, then v cannot be a convex vertex of the a-level nor a reflex vertex of the b-level, and so both $z(k)$ and $z(k')$ pass through v without a bend. If v is a convex vertex of the a-level or a reflex vertex of the b-level, then only one zig-zag path can reach it from the left, and so only one path leaves it on the right. ∎

Thus, the total number of bends in all the $(b-a+1)$ zig-zag paths is upper-bounded by the number of convex vertices of the a-level and the reflex vertices of the b-level. We, therefore, have the following lemma.

Lemma 3.3: If the a-level and the b-level have x vertices in total, then there exists a zig-zag path between them of size at most

$$\frac{x}{b-a+1}.$$

So, how many vertices can a single level of the n-line arrangement have? Unfortunately, determining the asymptotic complexity of levels in line arrangement is a notoriously difficult problem in computational geometry [9]. The best known upper bound for the complexity of k level is $O(nk^{1/3})$ due to Dey [8], and the best lower bound is

$\Omega(n \cdot 2^{\sqrt{\log k}})$ due to G. Tóth [29]. On the other hand, the *average* size of the first k level is always linear, by a result of Alon and Győri [2]. In particular, they show that, for any k, where $1 \leq k \leq n$, the *total size* (number of vertices) of the levels 1 through k is nk. We use this result to show that there is a linear size level in the vicinity of an εn level and of an $(\varepsilon n/2)$-level.

Lemma 3.4: In an arrangement of n lines in the plane, there is always a level of size at most $6n$ between the levels $\frac{5}{6}\varepsilon n$ and εn. Similarly, there is always a level of size at most $4n$ between levels $\frac{2}{3}\varepsilon n$ and $\frac{1}{2}\varepsilon n$.

Proof: By the result of Alon and Győri [2], the total complexity of the first εn levels is at most εn^2. Clearly, this is also an upper bound on the total complexity of the $\frac{1}{6}\varepsilon n + 1$ levels between levels εn and $\frac{5}{6}\varepsilon n$. By the pigeon hole principle, at least one of these levels must have size at most $\varepsilon n^2/(\frac{1}{6}\varepsilon n+1) \leq 6n$. An analogous argument shows that there is a level of size at most $4n$ between levels $\frac{2}{3}\varepsilon n$ and $\frac{1}{2}\varepsilon n$. ∎

We can now complete the proof of Theorem 3.1.

Proof: [of Theorem 3.1] Consider an arrangement of n lines in the plane. Choose a and b such that $\frac{1}{2}\varepsilon n \leq a \leq \frac{2}{3}\varepsilon n \leq b \leq \varepsilon n$, and the size of the a level is at most $4n$ and the size of the b-level is at most $6n$; such a and b exist by the preceding lemma. The total size of these two levels is at most $10n$, and $(b - a + 1) \geq \frac{1}{6}\varepsilon n$. By Lemma 3.3, we conclude that there is a zig-zag path of size $O(\frac{1}{\varepsilon})$ between levels a and b. This zig-zag path is clearly a separator between the εn and the $\frac{1}{2}\varepsilon n$ levels. ∎

The constant factors in Theorem 3.1 are quite loose. Our primary goal is simply of prove the asymptotic result that sentinel sets of size $O(\frac{1}{\varepsilon})$ exist. Our simulations show that in practice the sentinels sets are significantly smaller than the worst-case bound would indicate.

B. Detecting ε-cuts from a signature

The εn sensors that are cut off may lie either below or above the line. We, therefore, compute two separators, one to detect separation of points below the cutting line, and the other to detect separation above the line. In order to avoid unnecessary replication, we describe our scheme for the lower separator, with the understanding that the complete construction involves a symmetric application of the algorithm for the other case as well.

We have shown that there is an $O(\frac{1}{\varepsilon})$ size separator between levels εn and $\frac{1}{2}\varepsilon n$ in the arrangement. We choose our sentinel set W to be the dual of these $O(\frac{1}{\varepsilon})$ lines. Let $w_1, w_2, \ldots, w_m$, where $m = O(1/\varepsilon)$, denote the points (sensors) forming the sentinel set. We now show how to use the signature $\sigma(W)$ to determine whether there is an ε-cut or the cut is below the $\frac{\varepsilon}{2}$-cut threshold. See Figure 4 for an illustration.

Given a separator of size $O(\frac{1}{\varepsilon})$ between levels εn and $\frac{1}{2}\varepsilon n$ in the arrangement $H(S^*)$, the set W^* of lines spanned by the separator edges (the dual lines of the sentinels), and a signature vector $\sigma(W)$, we can report ε-cuts as follows. If a sensor w_i is dead, then we know that the dual of any possible linear cut L must be *above* w_i^*. Otherwise, we know that the dual must be *below* w_i^*. In either case, one of the two *halfplanes* defined by w_i^* is where the dual point L^* could possibly be. We determine the common intersection C of all the m halfplanes, one for each w_i^*, which can be computed in $O(m \log m)$ time by a standard divide and conquer algorithm [26]. We then choose an arbitrary point $p \in C$ in this common intersection, and determine if p lies above or below the separator. This can be done in $O(\log m)$ time because the separator is x-monotone and it has m links: it is sufficient to find the line $w^* \in W^*$ (by binary search) that lies directly above or below p, and test against that. If p is above the separator, we report that there is an ε-cut. Otherwise, we report that no ε-cut exists. *Due to lack of space, we omit the detailed proof of correctness of this algorithm. Instead, we simply illustrate the construction on an example.*

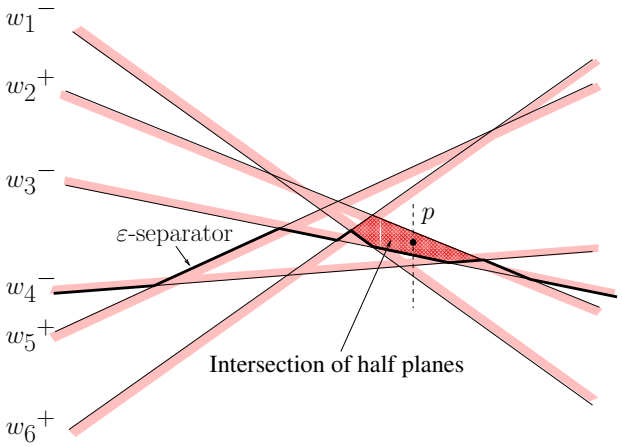

Fig. 4. The intersection of the half-planes determined by the sentinel lines is a cell of the arrangement.

Figure 4 shows an example with six sentinel nodes. Suppose that only w_2, w_5, and w_6 send signals to the base station, and so w_1, w_3, and w_4 are assumed to be cut off. The dual of any ε-cut must lie above the lines w_1^*, w_3^* and w_4^*, and it must lie below the lines below the lines w_2^*, w_5^*, and w_6^*. The common intersection of these halfplanes is shown as C. A point $p \in C$ lies above the ε-separator (drawn in bold line), and so we report an ε-cut.

IV. Computing a Sentinel Set

Our proof of Theorem 3.1 directly leads to a deterministic algorithm: we first compute the a and b levels, which are each of linear size, then find all the zig-zag separators, and pick the smallest one, which is guaranteed to have size $O(\frac{1}{\varepsilon})$ size. In the following, we present two improved schemes. an $O(n^2 \log n)$ time deterministic algorithm for computing a *minimum link separator*, and an $O(\frac{n}{\varepsilon} \log n)$ time *randomized* algorithm for computing an $O(\frac{1}{\varepsilon})$ size separator.

A. Finding a Minimum Link Separator

The algorithm implicit in our proof only uses the vertices of the a and b-levels. A minimum link separator, however, can use any lines and vertices *between* these two levels. Our new algorithm performs a plane-sweep over the arrangement $H(S^*)$, and for every edge e (between levels a and b) computes the minimum number of turns necessary to reach it from the left horizon boundary. In this way, we can compute the optimal separator between the levels a and b. The algorithm processes $O(n^2)$ vertices from left to right. Sorting the vertices by x-coordinates requires $O(n^2 \log n)$ time, which can be done either in advance or online by updating an event queue. At every vertex, only $O(1)$ work is done. We record all computation in $O(n^2)$ space in order to be able to back-track and output a minimum link separator when the line-sweep terminates. We compute an optimal size minimum link separator in $O(n^2 \log n)$ time and $O(n^2)$ space. Due to lack of space, we omit further details from this abstract.

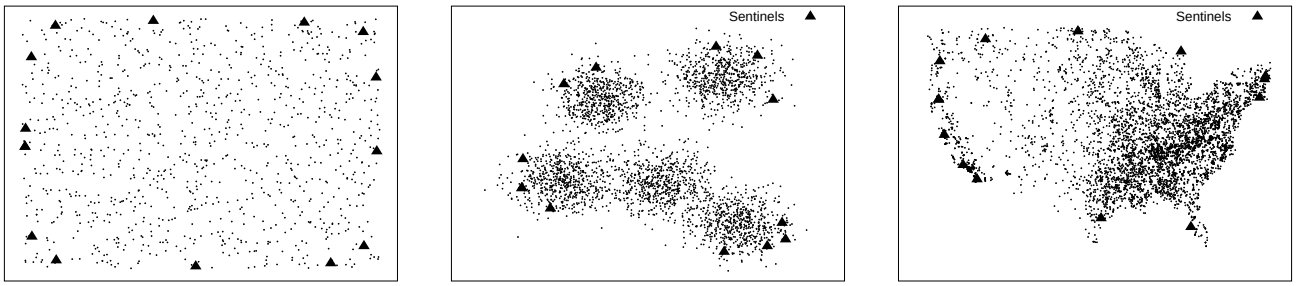

Fig. 5. The leftmost figure shows the uniform data for $n = 5000$; the middle figures shows the non-uniform data for $n = 5000$; the rightmost figure shows the US Census data (we show only a randomly sampled subset of points for clarity). In each case, we show a sentinel set for $\varepsilon = 0.01$ and $n = 5000$.

B. A Randomized Algorithm for Computing a Separator

The basic idea of our randomized algorithm goes back to our combinatorial proof presented in Section III. Every time we used the pigeon hole principle to find a below-average size level and a zig-zag path, we can use a *random* level and zig-zag path. The random choice will return a level or path of expected (average) size with constant probability.

In particular, we begin by choosing two integers uniformly at random $a \in [\frac{1}{2}\varepsilon n, \frac{2}{3}\varepsilon n]$ and $b \in [\frac{5}{6}\varepsilon n, \varepsilon n]$, and a random integer $k \in [a, b]$. We then compute the zig-zag path $z(k)$. We show that each segment of this path can be traced in $O(n \log n)$ time. Again, due to lack of space, we omit further details from this abstract. Since the zig-zag path $z(k)$ has expected complexity $O(\frac{1}{\varepsilon})$, the algorithm runs in expected time $O(\frac{n}{\varepsilon} \log n)$. It is important to note that although the size of sentinel has probabilistic guarantees, the resulting zig-zag path is always a separator between levels $\frac{1}{2}\varepsilon n$ and εn. In our experiments, we found that in practice the size of the ε-sentinel set is consistently smaller than the worst-case combinatorial bound.

V. Experimental Evaluation

In this section, we describe our simulations results that are intended to evaluate the scalability of our sentinel based scheme. We also performed experiments comparing our scheme with some simple sampling-based methods.

The geometric distribution of sensors is likely to vary widely from application to application. We, therefore, generated several random and non-random distributions of points in the plane to model a variety of sensor networks. We used three main data sets in our simulation: (1) uniform, (2) non-uniform, and (3) US census data. Figure 5 shows example distributions of these three sets. The uniform set contains n random points uniformly distributed in a square. The non-uniform set contains n points, equally divided among k clusters. Each cluster is centered at a random point, and the points in the cluster are generated using a Gaussian distribution. The last set is a US Tiger Census map, which includes locations of $14,000$ geographical features in the USA. We chose these locations as positions for the sensors.

Our first set of experiments show the scalability of the sentinel sets as a function of n and ε.

A. Scalability with Network Size

In order to isolate the effect of network size, we ran all experiments with a fixed value of $\varepsilon = .01$; the results are nearly identical with other values of ε. We generated networks of all three types (uniform, non-uniform, and tiger), with n varying from 1000 to 14000; for the census data, we randomly chose n points from the set for n smaller than $14K$. All the results shown in this experiment are averaged over 5 different seed values for the random number generator.

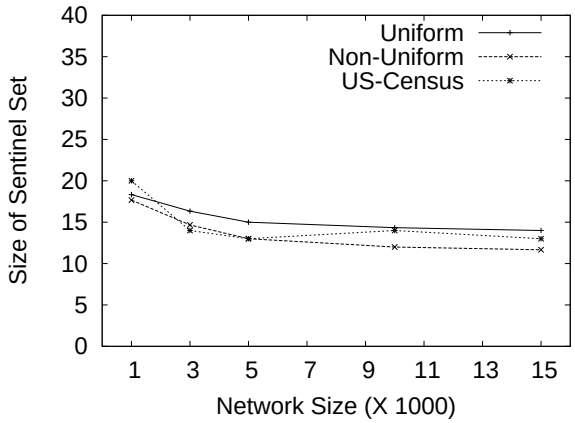

Fig. 6. The size of sentinel set vs. the network size.

As predicted by the theory, the results (Figure 6) show that the size of the sentinel set is independent of n. Moreover, the observed size of the sentinel set is *significantly* smaller than the worst-case bound of Theorem 3.1. In all cases, the sentinel set was smaller than even $1/\varepsilon$.

B. Scalability with ε

In this experiment, we evaluated the behavior of our scheme with different values of the cut threshold ε. These experiments were performed with networks of a fixed size $n = 5000$. We varied ε from 0.001 to 0.1, and the result is shown in Figure 7; again, all the results are averaged over 5 different seed values for the random number generator.

As expected, the size of the sentinel set increases as the value of ε decreases. Still, the size of the sentinel sets in all cases is significantly smaller than the worst-case bound of Theorem 3.1. Even for very small values of ε, say $\varepsilon = 0.001$, the algorithm generates sentinel sets of size less than 30.

C. Comparison with Other Schemes

We mentioned earlier that the problem of detecting network cuts has been raised in several papers, but no algorithms seem to have been proposed for it. Thus, we don't have any specific algorithm to compare with. (The ε-approximation is not a good scheme, because it requires a very large set of sentinels.)

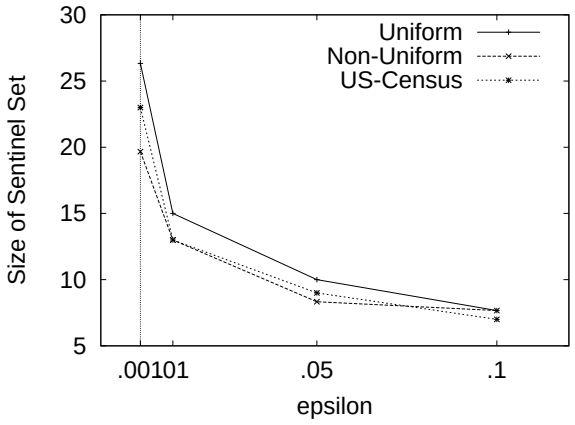

Fig. 7. The size of sentinel set vs. ε.

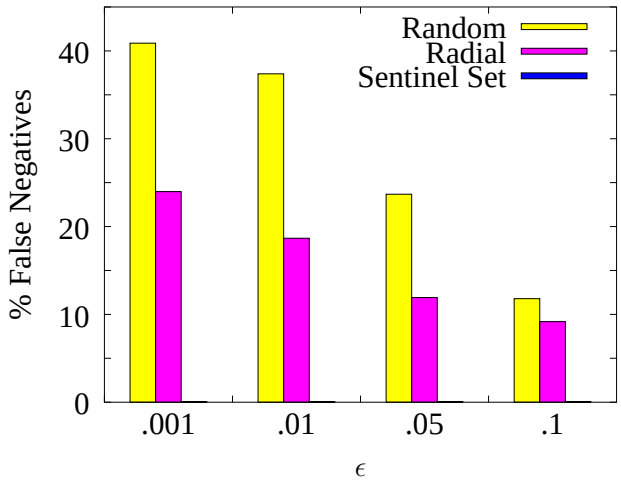

Fig. 8. False negatives: ε-cuts not detected by the random and radial sampling schemes. Our sentinel scheme has no false negatives.

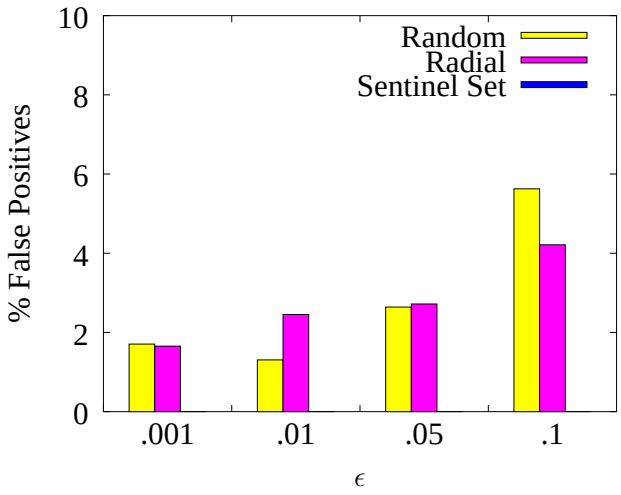

Fig. 9. False positives: the cuts reported by the random and the radial sampling that were below the approximation threshold $\frac{1}{2}\varepsilon n$. Our sentinel scheme has no false positives.

Instead, we implemented two natural heuristics to evaluate their effectiveness in comparison to our sentinel algorithm. Both schemes are based on sampling. In the first one, called *random sampling*, we choose a certain number of sensors uniformly at random and designate them as sentinels. Whenever any one of these chosen sentinels is cut off, the algorithm declares an ε-cut. In order to make a fair comparison with our sentinel scheme, we choose the same number of random nodes as the size of our sentinel set. A second scheme does *radial sampling*: we choose $|W|$ directions uniformly, and for each direction choose the εn-th extreme point, where $|W|$ is the size of the sentinel set for that instance.

For each test case, we first run our sentinel algorithm to compute the sentinel set W, and then use W samples for the random sampling and the radial sampling schemes. We evaluated the effectiveness of these schemes using *false negatives* and *false positives*. That is, how many ε-cuts are missed by these schemes (false negatives), and how many cuts reported as ε-cut are smaller than $\varepsilon n/2$. Our sentinel scheme yields no false positives or negatives, as guaranteed by theory.

We fixed $n = 5000$, and varied ε in the range $[0.001, 0.1]$. We show the results averaged over all the datasets, because we found that the characteristics of error is very similar across different sets. We simulated a large number of random linear cuts, and measured the false positives and negatives in each case.

For false negatives, we generated 250 cuts by randomly sampling between the εn and $2\varepsilon n$ levels of the arrangement. All these are true ε-cuts and, therefore, should be detected. Figure 8 shows that the sentinel scheme correctly detects all these cuts, but the random and radial sampling miss a significant fraction (between 10% and 40%) of these ε-cuts. We also ran experiments where cuts were chosen randomly between εn and $10\varepsilon n$ levels, and the results were essentially identical.

A further study (results not shown here) revealed that these incorrect decisions are also arbitrarily bad in terms of quality. With $\varepsilon = .01$, for instance, nearly 8% of the false negatives were in fact cuts where more than $7\varepsilon n$ of the sensors were cutoff; and some cuts of size up to $10\varepsilon n$ remained undetected.

For false negatives, we generated 250 cuts by randomly sampling between level 1 and level $\frac{1}{2}\varepsilon n$. These cuts are all below the approximation threshold, and so should not be reported. However, as Figure 9 shows the random and the radial sampling schemes misreport some of them as cuts.

In conclusion, even for relatively well-behaved distributions, the sampling schemes yield many false positives and negatives. Because the sentinel scheme chooses its sentinels carefully based on the distribution of points, in more irregular distributions, it can have significantly fewer sentinels than random sampling based methods. For instance, consider an example with three clusters of εn points each near the corners of an isosceles triangle; the remaining $n - 3\varepsilon n$ points lie in a cluster near the center of the triangle. Our scheme picks a constant number of sentinels from the 3 corners, which are sufficient to detect all ε-cuts. In random sampling, it will take $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon\delta})$ samples to ensure with probability $1-\delta$ that nodes from all three corner clusters are selected. Similar constructions are possible for radial sampling too.

VI. Discussion and Future Directions

We proposed a simple, low-overhead scheme for detecting cuts (partitions) in sensor networks. We show that linear ε-cuts can be detected by monitoring just $O(\frac{1}{\varepsilon})$ nodes of the network, which is asymptotically the best possible; a simple example of n sensors

arranged in a circle gives a matching lower bound. In practice, however, we expect even fewer than $\frac{1}{\varepsilon}$ sentinels, which is borne out by our simulation results.

An important feature of our algorithm is the lack of false positives or false negatives. Thus, every cut of size εn or larger is detected, and no cut is reported unless it includes at least $\frac{1}{2}\varepsilon n$ nodes.

One issue that we did not address is the noise or the inherent instability of individual sensor nodes. If one of the sentinel nodes dies naturally, it can mislead our algorithm. One possible way to deal with the inherent unreliability of individual sensors is to *use multiple copies of sentinel sets*. For instance, instead of one, we can compute k disjoint sentinel sets. Whenever an ε-cut occurs, each set independently will detect it. We can adopt the policy that we report an ε-cut only all k sets (or a majority of them) agree. Finding cut-detection schemes that are inherently fault-tolerant to multiple individual node failures is a challenging research problem, and a topic of our future work. We have implicitly assumed that the network cut does not destroy the communication path between any live sentinel and the base station. If a live sentinel's path does get disrupted, we can use any of the reactive *ad hoc* network protocols to discover and set up a new path to the base station.

In this paper, we have tried to minimize the communication cost for detecting linear cuts by using only a small number of sentinel nodes. Different sets of sentinels, however, may lead to different communication costs, and an important second-order optimization would take this effect into account. Another way to minimize the communication in the network would be to make the cut detection more decentralized. These are both very practical questions and natural directions for future work.

In this paper we have limited ourselves to *linear* cuts. This is an important and natural class of cuts, but a richer set of cuts would include *circular cuts*, *rectangular cuts*, and *polygonal cuts*. These classes, including the polygonal cuts as long as the polygons have only a constant number of sides, are ranges with a finite VC dimension. Therefore, the basic method of ε-approximation can be used to construct sentinel sets for each of these classes. Unfortunately, as we mentioned earlier, the worst-case bounds for ε-approximations and sensitive approximations are not very attractive. We plan to investigate if, using additional geometric insights, we can construct near-linear size sentinel sets for these more general forms of cuts.

ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers for their valuable comments and suggestions for future directions of research.

REFERENCES

[1] P. K. Agarwal, M. de Berg, J. Matousek, and O. Schwarzkopf. Constructing levels in arrangements and higher order Voronoi diagrams. *SIAM J. Comput.* **27**, 654–667, 1998.

[2] N. Alon and E. Győri. The number of small semispaces of a finite set of points in the plane. *J. Combin. Theory Ser. A* **41**, 154–157, 1986.

[3] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 2000.

[4] H. Brönnimann, B. Chazelle, and J. Matoušek. Product range spaces, sensitive sampling, and derandomization. *SIAM J. Comput.* **28**, 1552–1575, 1999.

[5] A. Cerpa and D. Estrin. ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies. *IEEE Transactions on Mobile Computing*, 3, 2004.

[6] C-Y. Chong and S. P. Kumar. Sensor Networks: Evolution, opportunities and challenges. *Proc. of the IEEE*, **91**, 1247–1256, 2003.

[7] D. Culler, D. Estrin and M. Srivastava. Overview of Sensor Networks. *IEEE Computer*, 41–49, 2004.

[8] T. K. Dey. Improved bounds for planar k-sets and related problems. *Discrete & Computational Geometry*, 373–382, 1998.

[9] P. Erdős, L. Lovász, A.Simmons, and E. G. Straus. Dissection graphs of planar point sets. In *A Survey of Combinatorial Theory*, 139–149, 1973.

[10] H. Edelsbrunner and E. Welzl. Constructing Belts in two-dimensional arrangements with applications. *SIAM J. Computing* **15**, 271–284, 1986.

[11] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. *Proc. of 5th Annual Mobicom*, ACM, 1999.

[12] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. *Proc. SenSys*, 2003.

[13] J. Gehrke and S. Madden. Query processing in sensor networks. *Pervasive Computing*, 2004.

[14] S. Har-Peled. Taking a walk in a planar arrangement. *SIAM J. Computing* **30**, 1341–1367, 2000.

[15] J. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond Average: Toward Sophisticated Sensing with Queries. *Proc. IPSN*, 2003.

[16] B. Hohlt, L. Doherty and E. Brewer. Flexible power scheduling for sensor networks. *Proc. IPSN*, 205–214, 2004.

[17] J. Kleinberg. Detecting network failure. *Internet Mathematics* **1**, 37–56, 2003.

[18] J. Kleinberg, M. Sandler, and A. Slivkins. Network failure detection and graph connectivity. *Proc. 15th Sympos. on Discrete Algorithms*, ACM Press, 76–85, 2004.

[19] J. Lifton, M. Broxton and J. Paradiso. An Implementation of Distributed Unconnected Graph Determination in Sensor Networks. *Project paper for MIT 6.978 course.*

[20] J. Liu, P. Cheung, F. Zhao and L. Guibas. A dual-space approach to tracking and sensor management in wireless sensor networks. *Proc. of International Workshop on Wireless Sensor Networks and Applications, 2002*

[21] J. Matoušek. Construction of ε-nets. *Proc. 5th Symp. Computational Geometry*, ACM Press, 1–10, 1989.

[22] J. Matoušek. ε-nets and computational geometry. *New Trends in Discrete and Computational Geometry*, Vol. 10, Algorithms and Combinatorics, 69–89, 1993.

[23] S. Nath, P. Gibbons, Z. Anderson and S. Seshan. Synopsis Diffusion for Robust Aggregation in Sensor Networks. *Proc. ACM SenSys*, 2004.

[24] J. Newsome and D. Song. GEM: Graph Embedding for Routing and Data Storage in Sensor Networks without Geographic information. *Proc. ACM SenSys*, 76–88, 2003.

[25] G. J. Pottie and W. J. Kaiser. Wireless integreated network sensors. *Communications of the ACM* **43**, 51–58, 2000.

[26] F. P. Preparata and M. I. Shamos. Computational Geometry: An Introduction, Springer-Verlag, New York, 1985.

[27] N. Sadagopan and B. Krishnamachari. Maximizing data extraction in energy-lmited sensor networks. *Proc. IEEE INFOCOM*, 2004.

[28] N. Shrivastava, C. Buragohain, D. Agrawal and S. Suri. Medians and Beyond: New Aggregation Techniques for Sensor Networks. *Proc. ACM SenSys*, 2004.

[29] G. Tóth. Sets of points with many halving lines. *Discrete and Computational Geometry* **26**, 187–194, 2001.

[30] G. T. Toussaint. A simple linear algorithm for intersecting convex polygons. *The visual computer* **1**, 118–123, 1985.

[31] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications* **16**, 264–280, 1971.

[32] F. Zhao, J. Liu, J. Liu, L. Guibas and J. Reich. Collaborative signal and information processing: An information directed approach. *Proc. of the IEEE*, 2003.