Wireless Netw
DOI 10.1007/s11276-007-0021-1

Localized algorithms for coverage boundary detection in wireless sensor networks

Chi Zhang · Yanchao Zhang · Yuguang Fang

© Springer Science + Business Media, LLC 2007

Abstract Connected coverage, which reflects how well a target field is monitored under the base station, is the most important performance metric used to measure the quality of surveillance that wireless sensor networks (WSNs) can provide. To facilitate the measurement of this metric, we propose two novel algorithms for individual sensor nodes to identify whether they are on the coverage boundary, i.e., the boundary of a coverage hole or network partition. Our algorithms are based on two novel computational geometric techniques called localized Voronoi and neighbor embracing polygons. Compared to previous work, our algorithms can be applied to WSNs of arbitrary topologies. The algorithms are fully distributed in the sense that only the minimal position information of one-hop neighbors and a limited number of simple local computations are needed, and thus are of high scalability and energy efficiency. We show the correctness and efficiency of our algorithms by theoretical proofs and extensive simulations.

Keywords Wireless sensor networks (WSNs) · Connected coverage · Computational geometry · Localized algorithm

C. Zhang (✉) · Y. Fang
Department of Electrical and Computer Engineering,
University of Florida, Gainesville, FL 32611
e-mail: zhangchi@ufl.edu

Y. Fang
e-mail: fang@ece.ufl.edu

Y. Zhang
Department of Electrical and Computer Engineering,
New Jersey Institute of Technology, Newark, NJ 07102
e-mail: yczhang@njit.edu

1 Introduction

Wireless sensor networks (WSNs) are ideal candidates for monitoring the physical space and enabling a variety of applications such as battlefield surveillance, environmental monitoring and biological detection. In such a network, a large number of sensor nodes are deployed over a geographic area (called the *region of interest* or ROI) for the purpose of monitoring certain events (e.g., emergence of the enemy's tanks). Typically, each sensor node has a very limited sensing range within which it is able to perform sensing operations. The sensed data will be transmitted to a *base station* (BS) over a multi-hop wireless path. The BS collects data from all connected nodes, concludes the activities in the ROI, and serves as a bridge to connect the WSN with outside users [2, 19].

As a consequence of this special network architecture, from the user's point of view, a position in the ROI is really under the surveillance of the WSN if and only if this position is within the sensing range of at least one of the sensor nodes connected to the BS. We define the collection of all these positions in the ROI as the *connected coverage*, or *coverage* in short, and argue in this paper that the continuous monitoring of the connected coverage is a must be functionality for all mission-critical WSNs to provide, regardless of their specific applications or focus.

First of all, connected coverage is the most important performance metrics used to measure the quality of service a WSN can provide in a certain time, and should be an inseparable complementarity of the report about the observed events in the ROI. For example, in the battlefield surveillance scenarios, the report from the BS that "none of the enemy's tanks have been observed in the ROI" is misleading

if it is not reinforced with the description of the current connected coverage. As sensors running out of energy, or being physically destroyed by natural or intended attacks, there is an inevitable devolution of the WSN characterized by the shrink of connected coverage or the growth of coverage hole in the ROI, and the WSN should continuously self-monitor the change of its coverage performance.

Secondly, the information of the connected coverage can also be used to facilitate many basic operations of WSNs. Some important ones are listed as follows:

Routing. If all the coverage boundaries can be identified beforehand, routing in a WSN can be very efficient, especially geographic routing [11]. The reason is that overlooking coverage boundaries may cause problems in communications, as routing along shortest paths tends to put an increased load on boundary nodes, thus quickly exhausting their energy supply and growing the coverage hole.

Topology control. In a densely deployed WSN, it is often suggested to allow sensor nodes to alternatively sleep to conserve energy while meeting the coverage requirement [21]. If a sensor node can self-identify its position on a coverage boundary, it can automatically tune its strategy to wake up neighboring nodes to fill in the coverage hole. Furthermore, in a WSN with both static and mobile sensors [32], identifying coverage boundaries among randomly deployed static nodes would help determine movement strategies of mobile sensors to improve connected coverage.

Self-diagnosis of network health. Self-diagnosing the health status of a WSN can keep sensor nodes aware of the probability of system failures, and help to launch many other network management activities. The current status of connected coverage is a very important input for such self-diagnosis. Also note that, instead of having all nodes to send their positions or health conditions to the sink, we only need a few nodes on coverage boundaries to do so. By doing so, we can not only reduce competitions for the wireless channel, but also save precious energy.

Re-deploying or repairing WSNs. For mission-critical applications, it may be necessary to repair or even re-deploy the WSN when the coverage performance is unsatisfactory. The details of coverage information can help decide when and how to perform the network repair or re-deployment. For example, we can know where the best places are for adding new nodes to reduce or eliminate the coverage holes and how many new nodes are needed.

In this paper, we develop two novel algorithms for coverage boundary detection in WSNs. In particular, we propose two novel computational geometric techniques, called *localized Voronoi polygon* (LVP) and *neighbor embracing polygon* (NEP), based on which two complementary algorithms are designed. The LVP-based algorithm requires both the directional information (the orientation of each neighbor) and the distance information (the distance to each neighbor), and theoretically can detect all the boundary nodes no matter how the nodes are distributed. By contrast, the NEP-based algorithm merely needs directional information, but can only find the local (or global) convex points of the coverage boundary. As compared to previous proposals, both algorithms can be applied to WSNs of arbitrary topologies. They are also truly distributed and localized by merely needing one-hop neighbors' information and a few simple local computations, and thus are of high scalability and energy efficiency. We show the correctness and efficiency of our algorithms by theoretical proofs and extensive experimental results.

The remainder of the paper is organized as follows. Section 2 provides the network model, problem definition and a concise overview of the existing proposals for coverage boundary detection. In Sections 3 and 4, we present the LVP-based and NEP-based algorithms and prove their correctness, respectively. Section 5 evaluates the performance of our algorithms by theoretical analysis and simulation results, and this paper is finally concluded in Section 6.

2 Preliminaries

In this section, we first give the notation, assumptions and the network model used in the paper, and then present the formal problem statement. The existing proposals for the coverage boundary detection will be summarized briefly at last.

2.1 Notation, assumption and network model

We use the following notation throughout the paper:

- $\|u - v\|$ or $\|uv\|$: the Euclidean distance between two points u and v, where $u, v \in \mathbb{R}^2$.
- ∂A: the topological boundary of a set $A \subset \mathbb{R}^2$.
- A^{C}: the complement of set $A \subset \mathbb{R}^2$, i.e. $A^{\mathsf{C}} = \mathbb{R}^2 - A$.
- $\overline{uv}$: the line segment from point u to v where $u, v \in \mathbb{R}^2$.
- n: the total number of sensor nodes in the network, or *network size*.
- s_i: the position of node i for $1 \leq i \leq n, i \in \mathbb{N}$.
- r_c: the communication range of sensor nodes.
- r_s: the sensing range of sensor nodes.
- $Disk\,(u, r)$: the closed disk of radius r and centered at point u. Let $\mathbf{0}$ indicate the origin and we have

$$Disk_{\mathbf{0}} = Disk(\mathbf{0}, r_s) = \{v : \|v - \mathbf{0}\| \leq r_s, v \in \mathbb{R}^2\}.$$

We define two operations on subsets of the Euclidean space:

- *Translation*: $A_u = A + u = \{v + u : v \in A\}$ for $u \in \mathbb{R}^2$ and $A \subset \mathbb{R}^2$.
- *Minkowski-addition*: $A \oplus B = \{u + v : u \in A, v \in B\}$ for $A, B \subset \mathbb{R}^2$. Obviously $A_u = A \oplus \{u\}$.

Throughout this paper, we assume that any two sensor nodes can directly communicate via bi-directional wireless links if their Euclidean distance is not greater than r_c, the *communication range*; and a position in the plane can be perfectly monitored (or covered) by a sensor node if their Euclidean distance is not greater than r_s, the *sensing range*. Although we use a simplified "disk model" here, our schemes are applicable to more general and practical scenarios. The impact of the disk model on the performance of our schemes is discussed in Appendix A. Similar to [3, 21, 33], we also assume that sensor nodes are homogeneous in the sense that r_c and r_s are the same for all nodes, and keep constant in each node's lifetime.

Instead of considering all the possible combinations of r_c and r_s, we focus on the case of $r_c = 2r_s$ in this paper. There are two reasons for doing so. First, as pointed out in [35], the specification of $r_c \geq 2r_s$ holds for most commercially available sensors such as Berkeley Motes and Pyroelectric infrared sensors. Second, as shown in Appendix B, for arbitrary spatial distributions of sensor nodes, $r_c \geq 2r_s$ is the sufficient and necessary condition for the existence of localized boundary node detection algorithms.[1] Therefore, we set $r_c = 2r_s$ to reduce communication energy consumption and interference. However, it should be noted that our algorithms are still applicable to the scenarios of $r_c > 2r_s$ without any changes.

For simplicity, we assume that the ROI is a 2-D square planar field hereafter. Our results, however, can be easily extended to 2-D or 3-D ROIs of arbitrary shapes. For $l > 0$, let A_l denote the square ROI of side length l centered at the origin, i.e., $A_l = [-l/2, l/2]^2$, and ∂A_l be the *border* of A_l. We examine a large-scale WSN consisting of hundreds or even thousands of stationary sensor nodes,[2] and denote the sensor nodes deployed in the ROI as $V = \{s_1, \ldots, s_i, \ldots, s_n\}$.

2.2 Formal definition of the problem

We now formally define the *connected coverage boundary detection* (CCBD) problem addressed in this paper. We start with a few definitions.

Definition 1. A connected set of nodes is said to be a *maximally connected set*, or a *cluster*, if adding any other node to the set will break the connectedness property. We write $Clust(s_i)$ for the cluster containing node s_i.

Based on the sensing model, the *sensing disk* of node s_i is given by $Disk(s_i, r_s) = Disk_{\mathbf{0}} + s_i$. Then the coverage corresponding to a cluster can be defined as follows:

Definition 2. We define the set of all points in A_l that are within radius r_s from any node of $Clust(s_i)$ as the set covered by $Clust(s_i)$. This set is denoted by

$$Cover(s_i) = \left(\bigcup_{u \in Clust(s_i)} (u + Disk_{\mathbf{0}}) \right) \cap A_l. \tag{1}$$

Definition 3. We define the *boundary nodes* of $Clust(s_i)$ as those whose minimum distances to $\partial Cover(s_i)$ are equal to r_s, and denote them by

$$BN(s_i) = \{u \in Clust(s_i) : \min \|u - v\| = r_s \text{ for } v \in \partial Cover(s_i)\};. \tag{2}$$

Accordingly, *interior nodes* is defined by

$$IN(s_i) = \{u \in Clust(s_i) : u \notin BN(s_i)\}. \tag{3}$$

We denote the position of the base station as BS. Note that the cluster $Clust(s_i)$ is connected with the BS if and only if $BS \in Clust(s_i) \oplus Disk(\mathbf{0}, r_c)$. Therefore, the *connected coverage* with BS, which means the total area of the ROI under the surveillance of BS due to contributions from each sensor node connected to BS, can be formally defined as

$$Cover(BS) = \bigcup_{1 \leq i \leq n} \{Cover(s_i) : BS \cap (Clust(s_i) \oplus Disk(\mathbf{0}, r_c)) \neq \emptyset\}. \tag{4}$$

Note that $Cover(BS)$ is uniquely decided by its boundary $\partial Cover(BS)$. Assume there are two different clusters $Clust(s_i)$ and $Clust(s_k)$ connected with BS, from Definition 2, we have $Cover(s_i) \cap Cover(s_k) = \emptyset$. Therefore,

$$\partial Cover(BS) = \left\{ \bigcup \partial Cover(s_i) : Clust(s_i) \text{ is connected with } BS \right\}, \tag{5}$$

which means CCBD problem can be simplified as finding the coverage boundary of each cluster, i.e., $\partial Cover(s_i)$. Since the minimum information required to describe $\partial Cover(s_i)$ is r_s and $BN(s_i)$, the CCBD problem is equivalent to finding the

[1] The formal definition of "boundary nodes" and "localized algorithms" will be given in Sections 2.2 and Appendix B, respectively.

[2] Stationary nodes here do not imply that the topology of the WSN is static. Instead, the WSN may have highly dynamic topology changes due to nodes failures, new nodes additions or nodes switching their states between active and sleeping modes to save energy. One advantage of our schemes lies in the efficiency to handle topology changes in WSNs (cf. Section 3.4).

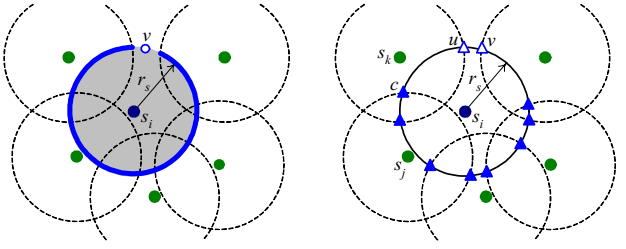
set $BN(s_i)$. Note that the CCBD problem formulated above can be easily generalized to the cases with multiple BSs or mobile BSs.

2.3 State of the art

The task of CCBD will be trivial if we do it in a centralized way and the exact locations of all nodes are available. For example, a single node has access to locations of all functional sensors (an "image" of the sensor distribution). In this scenario, traditional ways of edge detection in image processing are applicable. However, due to the energy constraints, this scenario is impractical for most WSNs. Distributed solutions to the CCBD problem have already been proposed in [11, 13, 16, 17, 32, 35]. In what follows, we further classify these approaches according to the boundary node identification methods they adopted.

2.3.1 Perimeter-based approaches

The first localized boundary node detection algorithm is proposed in [17], which is based on the information about the coverage of the perimeter of each node's sensing disk. It can be shown that node s_i is a boundary node if and only if there exists at least one point $v \in \partial Disk(s_i, r_s)$ which is not covered by any $s_j \in Neig(s_i)$ (cf. Fig. 1(a)). Based on this criterion, an algorithm with the complexity $O(k \log k)$ is designed in [17] to locally check whether one node is a boundary node, where k is the number of neighbors. A crossing-coverage checking approach proposed in [16, 35] simplifies the previous perimeter-coverage checking approach by just checking some special points called crossings on the perimeter. A *crossing* is defined as an intersection point of two perimeters of sensing disks. A node s_i is a boundary node if and only if there exists at least one crossing $v \in \partial Disk(s_i, r_s) \cap \partial Disk(s_j, r_s)$ which is not covered by any other $s_k \in Neig(s_i) - \{s_j\}$. Figure 1(b) shows an example where c is a crossing determined by two perimeters



(a) (b)

Fig. 1 Perimeter-based boundary node detection approaches. (a) Perimeter-coverage checking approach proposed in [17]. The solid curve represents the portion of perimeter of sensing disk covered by neighbor nodes. (b) Crossing-coverage checking approach proposed in [16, 35]. Solid and open triangles represent covered and uncovered crossings, respectively

$\partial Disk(s_i, r_s)$ and $\partial Disk(s_j, r_s)$, which is covered by the third sensing disk of node s_k. The problem of perimeter-based approaches is that each node needs to check positions and status of all of its neighbors, which is inefficient when the sensor nodes are densely deployed (cf. Section 5) so that every time when a node dies, all its neighbors need to check the coverage of their perimeters or crossings again.

2.3.2 Polygon-based approaches

In [11, 13, 32], *Voronoi diagram* (VD) is used for boundary node detection. Briefly speaking, the VD of a node set V, is the partition of the Euclidean space into polygons, called *Voronoi polygons* (VPs) and denoted by $Vor(s_i)$ for $s_i \in V$ such that all the points in $Vor(s_i)$ are closer to s_i than to any other node in V. According to the closeness property of VPs, if some portion of a VP is not covered by nodes inside the VP, it will not be covered by any other node, which implies a coverage hole. Therefore, it is claimed in [11, 13, 32] that each node can locally check whether it is on the coverage boundary under the assumption that VPs can be derived locally. However, it has been shown that in general VPs cannot be locally computed [34]. In fact, VP-based approach is not a real localized solution. It does not work when the survival nodes are sparsely distributed. In this paper, we still follow the line of polygon-based approaches, since the VP and its derivatives provide more information about the spatial distribution of one node's neighbors, which can be used to design more efficient detection schemes and simplify the updating procedures when the number of neighbors changes.

There is a trend in the literature to provide some basic functionalities of WSNs by only using directional information [5, 23, 30]. The boundary node detection with only directional information is an untouched topic since all the existing schemes are based on the directional and distance information of each node's neighbors. We will return to this topic in Section 4 to propose a solution for this scenario.

3 Localized Voronoi Polygons

In this section, we describe our first algorithm for identifying boundary nodes based on two novel geometric concepts called *Localized Voronoi Polygon* (LVP) and *Tentative LVP* (TLVP) which are nontrivial generalization of *Voronoi Polygons* (VPs) [27] from computational geometry. We must point out that a similar concept called *Localized Voronoi Diagrams* (LVDs) is introduced as the dual of *Localized Delaunay Triangulations* (LDTs) in the literature[18, 24]. The edge complexity of LDT is analyzed in [18] and its applications in topology control and routing for wireless networks are discussed in [24]. However, there is no indication on how to relate this concept to the coverage problems in WSNs.

Moreover, unlike our work, there is no description on how to efficiently construct LVDs given in [18, 24]. Furthermore the idea of using TLVP to reduce the overhead of the detection algorithm in this paper is completely new. Finally and most importantly, our scheme only uses the local information to detect the boundary instead of global information commonly used in either VP or DT.

3.1 Definition and properties of LVPs

To facilitate our illustration, we first define VPs, LVPs and TLVPs in terms of half planes. For two distinct points $s_i, s_j \in V$, the *dominance region* of s_i over s_j is defined as the set of points which are at least as close to s_i as to s_j, and is denoted by

$$Dom(s_i, s_j) = \{v \in \mathbb{R}^2 : \|v - s_i\| \leq \|v - s_j\|\}. \tag{6}$$

Obviously, $Dom(s_i, s_j)$ is a half plane bounded by the perpendicular bisector of s_i and s_j, which separates all points in the plane closer to s_i from those closer to s_j.

Definition 4. The VP associated with s_i is the subset of the place that lies in all the dominance regions of s_i over other points in V, namely,

$$Vor\,(s_i) = \bigcap\nolimits_{s_j \in V - \{s_i\}} Dom(s_i, s_j). \tag{7}$$

In the same way, the *localized Voronoi polygon* (LVP) $LVor(s_i)$ and the *tentative localized Voronoi polygon* (TLVP) $TLVor(s_i)$ associated with s_i are defined as:

$$LVor(s_i) = \bigcap\nolimits_{s_j \in Neig(s_i)} Dom(s_i, s_j); \tag{8}$$

$$TLVor(s_i) = \bigcap\nolimits_{s_j \in SubNeig(s_i)} Dom(s_i, s_j), \tag{9}$$

where $SubNeig\,(s_i) \subset Neig\,(s_i)$.

The collection of LVPs given by

$$\mathfrak{LVor}\,(V) = \{LVor\,(s_i) : s_i \in V\} \tag{10}$$

is called the *localized Voronoi diagram* (LVD) generated by the node set V. The boundary of $LVor(s_i)$, i.e., $\partial LVor(s_i)$, may consist of line segments, half lines, or infinite lines, which are all called *local Voronoi edges*.

Lemma 1. Properties of VPs, LVPs and TLVPs:

(i) $LVor\,(s_i)$, $TLVor\,(s_i)$ and $Vor\,(s_i)$ are convex sets;
(ii) $Vor\,(s_i) \subseteq LVor\,(s_i) \subset TLVor\,(s_i)$;
(iii) Plane $\mathbb{R}^2$ is completely covered by $\mathfrak{LVor}(V)$.

Proof: (i) Since a half plane is a convex set and the intersection of convex sets is a convex set,[3] an LVP (or a TLVP) as well as a VP is a convex set.

(ii) From (7), (8) and (9) we have

$$Vor(s_i) = LVor(s_i) \bigcap \left(\bigcap\nolimits_{s_j \in V, s_j \notin Neig(s_j)} Dom(s_i, s_j) \right),$$

$$LVor(s_i) = TLVor(s_i) \bigcap \left(\bigcap\nolimits_{s_j \in Neig, s_j \notin SubNeig(s_j)} Dom(s_i, s_j) \right),$$

which directly leads to Lemma 1(ii).

(iii) It is well known in computational geometry that

$$\bigcup\nolimits_{s_i \in V} Vor\,(s_i) = \mathbb{R}^2. \tag{11}$$

(cf. [27, Property V1, pp. 77] for a reference). Combining (11) with Lemma 1(ii) that $Vor\,(s_i) \subseteq LVor\,(s_i)$, we can directly obtain Lemma 1(iii). □

Therefore, the set $\mathfrak{LVor}\,(V) \cap A$ can fully cover the arbitrary set A where $A \subseteq \mathbb{R}^2$. Note that this result can be easily extended to any cluster in V, e.g., for $Clust\,(s_i)$ we have

$$\bigcup\nolimits_{s_j \in Clust(s_i)} LVor(s_j) = \mathbb{R}^2. \tag{12}$$

3.2 LVP-based boundary node detection

In this subsection, we present an algorithm for each node to detect whether it is on the coverage boundary based on its own LVP or TLVP, which is illustrated with node s_i as an example.

3.2.1 Input

Our BOND is a distributed scheme in that we only need positions of node s_i's neighbors as the input of our algorithm. We need to consider two cases based on whether the information about the border of A_l, i.e., ∂A_l, is available. In the first case where ∂A_l is unavailable at node s_i, our detection scheme is based on the construction of $LVor\,(s_i)$ (or $TLVor\,(s_i)$); in the second case where ∂A_l is available, we need to exploit this information by calculating $LVor\,(s_i) \cap A_l$ (or $TLVor\,(s_i) \cap A_l$). It can be shown that $LVor\,(s_i) \cap A_l$ must be a finite convex polygon. Thus, the second case can be transformed into the first case by introducing dummy nodes into $Neig(s_i)$. See Fig. 2(a) for an example, in which four dummy nodes, d_1 through d_4, are introduced such that perpendicular bisectors

[3] This sublemma can be proved as follows: Let B_i, $i \in \mathbb{I}$, be a convex set and $B = \bigcap_{i \in \mathbb{I}} B_i$. If u and v are two points in B, then they are in each B_i, so the line joining u to v lies in each B_i and therefore in B.

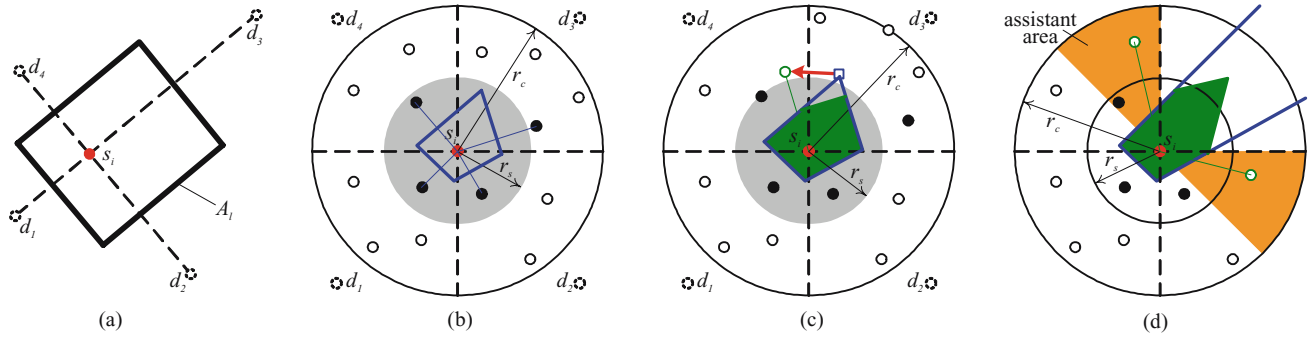


Fig. 2 Illustration of LVP-based boundary node detection algorithm

between s_i and the dummy nodes generate the four border edges of ROI. Then we can calculate $LVor(s_i) \cap A_l$ by following the same procedure for calculating $LVor(s_i)$. Therefore, we will discuss only the first case in what follows.

We notice that dummy nodes cannot be directly applied to the generalized cases, i.e., the border of A_l consisting of curves. However in these cases, it merely means that the information of A_l's border cannot be efficiently exploited. But the correctness of our scheme is not affected. There also exist two easy ways to remedy our BOND here. First, in general a curve can be approximated with straight line segments and thus the BOND is still applicable. Second, instead of checking whether the vertices of $LVor(s_i) \cap A_l$ are covered by $Disk(s_i, r_s)$ when A_l is a polygon, we can still correctly detect boundary nodes by checking every point on $\partial(LVor(s_i) \cap A_l)$ when A_l is not a polygon.

3.2.2 Algorithm

Our goal is to construct the $LVor(s_i)$ (or $TLVor(s_i)$) which is sufficient for the boundary node detection with the minimal information required about s_i's neighbors. We first divide $Disk(s_i, r_c)$ into four[4] quadrants. Then we construct the TLVP of s_i by using the nearest neighbors (solid nodes in Fig. 2(b)) in each of the four quadrants. Without loss of generality, we denote these four nearest neighbors as s_1, s_2, s_3, and s_4. The first TLVP is calculated by

$$TLVor(s_i) \leftarrow \bigcap_{j=1}^{4} Dom(s_i, s_j).$$

If all vertices of the TLVP are covered by $Disk(s_i, r_s)$, the procedure stops and this TLVP is saved. Otherwise, we need to find new neighbors which are the nearest to the uncovered vertices of the TLVP (cf. Fig. 2(c)), add those neighbors to $SubNeig(s_i)$, and calculate the TLVP again:

$$TLVor(s_i) \leftarrow TLVor(s_i) \bigcap \left(\bigcap_{s_j \in SubNeig(s_i), j \neq 1,2,3,4} Dom(s_i, s_j) \right).$$

The new vertices of the new TLVP will be checked to see whether they are covered by $Disk(s_i, r_s)$. This procedure continues until all the vertices of the TLVP are covered by $Disk(s_i, r_s)$ or the LVP of s_i is calculated and saved. We refer to the neighbors used to construct the LVP or TLVP at the end of this procedure as its *consulting neighbors*.

Note that when ∂A_l is unavailable, $LVor(s_i)$ may be infinite, which means that it is possible that we cannot find any node in one or more quadrants in the first step. See Fig. 2(d) for an example. If a quadrant contains no neighbors, we define two sectors of angle 45° which are directly adjacent to the quadrant as the assistant area, and add the nodes in this area to $SubNeig(s_i)$ first. If all the nodes in the assistant area cannot make TLVP finite, we can conclude that LVP must be infinite without need to do further calculation.

3.2.3 Output

If $LVor(s_i)$ is infinite, s_i must be a boundary node. If $LVor(s_i)$ (or the final $TLVor(s_i)$) is finite with all the vertices covered by s_i, then $s_i \in IN(s_i)$. Otherwise, $s_i \in BN(s_i)$.

3.3 Validating the algorithm

In the VD, the VPs of different nodes are mutually exclusive, but in the LVD, the LVPs of different nodes may overlap. This critical difference makes the validating of our algorithm totally different with the VP-based ones.

Theorem 1. If there is a point $v \in LVor(s_i)$ which is not covered by s_i, i.e., $v \notin Disk(s_i, r_s)$, there must exist a point $h \in LVor(s_i)$ that is not covered by any node, and s_i must be a boundary node.

[4] Other values will also work well.

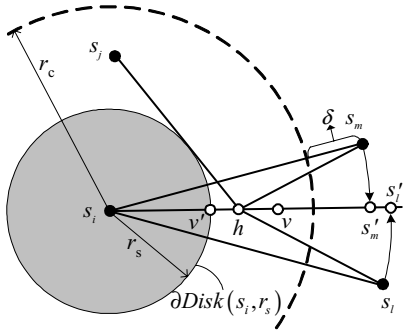


Fig. 3 Illustration of the proof of Theorem 1

Proof: Without loss of generality, we assume that the node nearest to s_i and outside $Disk(s_i, r_c)$ is s_m, and $\|s_i - s_m\| = r_c + \delta$ for $\delta > 0$. Let s'_m be the point on $\overline{s_i v}$ satisfying $\|s_i s'_m\| = \|s_i s_m\|$, and h be another point on $\overline{s_i v}$ such that $\|s_i h\| = r_s + \delta/2$ (see Fig. 3). By the triangular inequality, we have $\|s_m h\| + \|s_i h\| \geq \|s_i s_m\| = \|s_i s'_m\| = \|s_i h\| + \|h s'_m\|$. Therefore, $\|s_m h\| \geq \|h s'_m\| = \|s_i s'_m\| - \|s_i h\| = r_s + \delta/2$, which means that s_m cannot cover h and neither can any other node in $Disk(s_i, r_c)^{\mathsf{C}}$. The reason is that, since $\|s_i s_l\| > \|s_i s_m\|$ holds for any node $s_l \in Disk(s_i, r_s)^{\mathsf{C}}$ and $s_l \neq s_m$, we have $\|s'_l h\| > \|s'_m h\|$ where point s'_l is on the line $\overline{s_i v}$ and $\|s_i s'_l\| = \|s_i s_l\|$. Therefore, $\|s_l h\| \geq \|s'_l h\| > \|s'_m h\| = r_s + \delta/2$.

Since $v \in LVor(s_i)$, based on the convexity of $LVor(s_i)$ we have $\overline{s_i v} \in LVor(s_i)$. Therefore, $h \in LVor(s_i)$, which implies for any node $s_j \in Disk(s_i, r_c)$ and $s_i \neq s_j$, we have $\|s_j h\| \geq \|s_i h\| > r_s$, i.e., no node in $Disk(s_i, r_c)$ can cover h. Consequently, we can conclude that no node in the plane can cover h because $Disk(s_i, r_c) \cup Disk(s_i, r_c)^{\mathsf{C}} = \mathbb{R}^2$. Note that from the above proof process, we can see that h can be arbitrary close to v', the intersection of circle $\partial Disk(s_i, r_s)$ and $\overline{s_i v}$. Therefore, s_i is a boundary node. □

Theorem 2. If there is a point $v \in A_l$ not covered by any sensor node, for every cluster $Clust(s_i)$ there must exist at least one sensor $s_j \in V$ whose $LVor(s_j)$ is not completely covered by $Disk(s_j, r_s)$.

Proof: According to Lemma 1(iii) or (12), we have

$$\bigcup\nolimits_{s_j \in Clust(s_i)} (LVor(s_j) \cap A_l) = A_l \tag{13}$$

Therefore, for any $v \in A_l$, it must lie in at least one $LVor(s_j) \cap A_l$ for $s_j \in Clust(s_i)$. □

Theorems 1 and 2 prove that $LVor(s_i) \cap A_l$ is completely covered by s_i for all $s_i \in Clust(s_j)$ is the sufficient and necessary condition for $Clust(s_j)$ to completely cover A_l. The following theorem shows that when $LVor(s_i)$ or $LVor(s_i) \cap A_l$ is finite, the coverage of vertices of $LVor(s_i)$ (or final $TLVor(s_i)$, since $LVor(s_i) \subset TLVor(s_i)$) by s_i is equivalent to the coverage of the whole $LVor(s_i)$ by s_i, which guarantees the correctness of our LVP-based algorithm.

Theorem 3. $LVor(s_i)$ is fully covered by s_i if and only if $LVor(s_i)$ is finite and all the vertices are covered by s_i.

Proof: Let $Ve(s_i)$ be the set of vertices of $LVor(s_i)$. Obviously, when $LVor(s_i)$ is completely covered by s_i, i.e., $LVor(s_i) \subset Disk(s_i, r_s)$, we have $v \in Disk(s_i, r_s)$ for all $v \in Ve(s_i)$ and $LVor(s_i)$ is finite. Since

$$\max_{u \in LVor(s_i)} \{\|s_i - u\|\} \leq \max_{v \in Ve(s_i)} \{\|s_i - v\|\},$$

when $v \in Disk(s_i, r_s)$ for all $v \in Ve(s_i)$, we have $u \in Disk(s_i, r_s)$ for all $u \in LVor(s_i)$. □

3.4 Discussions on LVP-based detection

Our LVP-based detection is a truly localized polygon-based solution since computing $LVor(s_i)$ (or $TLVor(s_i)$) only needs one-hop information (this can be directly obtained from Definition 4, which is impossible for computing $Vor(s_i)$).

Assuming that the number of trusted neighbors is k, each node can compute its own $LVor(s_i)$ with complexity smaller than $O(k)$. In addition, the computation of the $LVor(s_i)$ only involves some simple operations on polygons which can be efficiently implemented (e.g., PolyBoolean library [22]). We further simplify the detection process by constructing TLVPs first. For a densely deployed WSN, we have $LVor(s_i)$ or $TLVor(s_i) \to Vor(s_i)$, and it is well known in computational geometry that under the homogeneous spatial Poisson point process, the average number of vertices of $Vor(s_i)$ is 6 [27]. Therefore, when the node density is high, our LVP-based detection on average needs only 4 to 6 nearest neighbors' information to successfully detect the boundary nodes. Moreover, when a neighbor node dies, our LVP-based detection need do nothing unless the dead node is used to construct the final $TLVor(s_i)$ or $LVor(s_i)$ in the last turn of LVP or TLVP construction. This unique property will greatly simplify the update of detection results and save precious energy of each sensor node. None of these advantages can be achieved by other localized boundary node detection schemes in the literature, such as the perimeter-coverage checking approach [17] and the crossing-coverage checking approach [16, 35]. We refer to Section 5.4 for a detailed comparison.

4 Neighbor embracing polygons

The *neighbor embracing polygon* (NEP) was first introduced in computational geometry as an alternative to the Voronoi

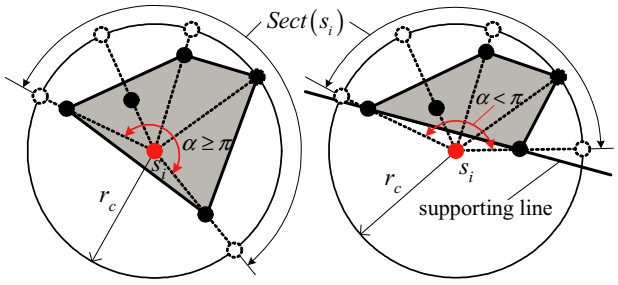


Fig. 4 Illustration of the convex hull of node s_i's neighbors (shaded area) and the smallest sector $Sect(s_i)$ containing all neighbors when (a) node s_i has the NEP and (b) node s_i does not have the NEP. Solid nodes represent neighbors of s_i and dotted open nodes are the projection of neighbors on the boundary of $Disk(s_i, r_c)$

polygon [8, 10]. In this section, we will show that the localized NEP can also be used as a complementary tool of the LVP for coverage boundary detection. We will also demonstrate the close relationship of NEPs with barrier coverage and network connectivity.

4.1 Definition and properties of NEPs

Definition 5. Given the point set $Neig(s_i)$, we define its *convex hull*, $CH\ (Neig(s_i))$, as the smallest convex set containing all the points in $Neig(s_i)$. If s_i is in the interior of $CH(Neig(s_i))$, i.e., $s_i \in CH(Neig(s_i))$ and $s_i \notin \partial CH(Neig(s_i))$, we call $CH(Neig(s_i))$ the NEP of s_i.

If s_i belongs to $CH(Neig(s_i))$, s_i has at least three neighbors. By the properties of the convex hull, we also know that $CH(Neig(s_i))$ is the unique convex polygon whose vertices are points from $Neig(s_i)$ [4].

The idea of using NEP in boundary node detection is rather intuitive. Figure 4 illustrates the relationship between s_i and its $CH(Neig(s_i))$. We can see that s_i is more likely to be far from the boundary when embraced by its neighbors. On the other hand, when $s_i \notin CH(Neig(s_i))$, we can find a line (*supporting line* for the convex set) which separates s_i from its neighbors, and node s_i is on the boundary almost for sure.

There is a clear difference between our definition and the existing one in [8, 10]. An NEP is constructed globally in [8, 10]: for a given node $s_i \in V$, they first connect s_i to the nearest node, then to the second nearest, and so on; the process continues either when s_i belongs to the interior of the convex hull of these nearest nodes or when all the nodes in V have been tested. In this way, only the vertices of $CH(V)$ do not have the NEP.[5] By our definition, when nodes without the NEP are found, some *local convex points* other than vertices of $CH(V)$ (global convex points) will also be identified, which can provide more detailed boundary information (cf. Section 5.1). More importantly, our scheme can be done locally.

Although efficient algorithms for computing the convex hull of a given point set are available, we still want to avoid using them if possible. The reason is that we only need to know whether there is an NEP for node s_i and do not care about the shape or size of the NEP. Let $Sect(s_i)$ be the smallest sector with angle α whose apex is s_i and contains all the points of $Neig(s_i)$. Note that $Sect(s_i)$ can be represented by two points on $\partial Disk(s_i, r_c)$. In fact, we can project all the points of $Neig(s_i)$ onto $\partial Disk(s_i, r_c)$,[6] and view $Sect(s_i)$ as the "1-D convex hull" of the projected points of $Neig(s_i)$ on $\partial Disk(s_i, r_c)$ (see Fig. 4). Intuitively, the existence of the NEP depends on the magnitude of angle α, which can be formally expressed by the following lemma:

Lemma 2. For a given finite set $Neig(s_i)$, node s_i has an NEP if and only if the angle α of $Sect(s_i)$ is larger than π.

Proof: See [8, Lemma 2]. Note that, different from [8], the degenerate case in which there are only two vertices of $CH(Neig(s_i))$ defining a line segment that contains s_i and $\alpha = \pi$, has been excluded here by Definition 5. □

Therefore, checking the existence of an NEP can be done solely based on directional information.

4.2 NEP-based boundary node detection

Based on Lemma 2, the NEP-based boundary node detection works as follows with node s_i as an example:

4.2.1 Input

The NEP-based algorithm does not require distances to s_i's neighbors, but needs only directions to s_i's neighbors. Therefore, we can use neighbors' projections on $\partial Disk(s_i, r_c)$ as their representations (i.e., s_j is a point on $\partial Disk(s_i, r_c)$). Accordingly, $Sect(s_i)$ can be determined by the two end points on $\partial Disk(s_i, r_c)$. We consider only the case when ∂A_l is unavailable in this section, since this information cannot be utilized even if it is available.

4.2.2 Algorithm

An intuitive way to check the existence of NEP is to sort $s_j \in Neig(s_i)$ according to their angles to s_i, and then check if there is a gap greater than π in these angles. The

[5] NEPs are never used in coverage boundary detection in [8, 10].

[6] r_c here can be replaced by any other value.

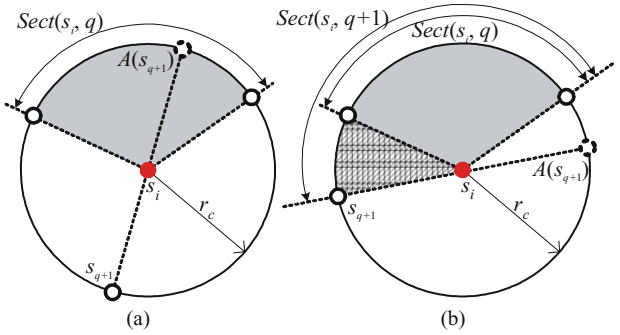


Fig. 5 Illustration of the computation of $Sect(s_i, q+1)$ from $Sect(s_i, q)$

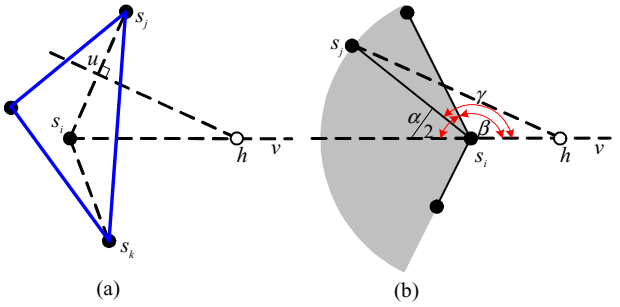


Fig. 6 Illustration of the Proof of Theorem 4

average complexity of sorting is $O(k \log k)$, where k is the number of neighbors. In the following, we describe how to decide whether $\alpha > \pi$ in $O(k)$ time by computing tentative $Sect(s_i, n)$, where n is the number of neighbors used in computing this tentative sector.

We first randomly take two points from $Neig(s_i)$, and construct a tentative sector $Sect(s_i, 2)$. Then we compute $Sect(s_i)$ iteratively, by adding points to the tentative sector one by one. Given $Sect(s_i, q)$ $(q < k)$ and the next point s_{q+1} randomly chosen from $Neig(s_i)$, if s_{q+1} is contained in $Sect(s_i, q)$, $Sect(s_i, q+1) = Sect(s_i, q)$. When $s_{q+1} \notin Sect(s_i, q)$ and the antipodal point[7] $A(s_{q+1})$ of s_{q+1} is contained in $Sect(s_i, q)$ (excluding end points), we can immediately decide $\alpha > \pi$ without further computation. When $A(s_{q+1}), s_{q+1} \notin Sect(s_i, q)$, we update $Sect(s_i, q)$ (the shaded area in Fig. 5(a)) by adding to $Sect(s_i, q)$ the sector (the dashed area in Fig. 5(b)) which is from an endpoint of $Sect(s_i, q)$ to s_{q+1} and does not contain $A(s_{q+1})$. This procedure continues until it can be decided that $\alpha > \pi$ or $Sect(s_i)$ is computed and α is obtained.

4.2.3 Output

If $\alpha \leq \pi$, $s_i \in BN(s_i)$. However, when $\alpha > \pi$, we cannot decide whether s_i is a boundary node.

4.3 Validating the algorithm

We first give the relationship between LVPs and NEPs.

Theorem 4. The *LVP* $LVor(s_i)$ is infinite if and only if there is no *NEP* for s_i.

Proof: We prove the first part by contradiction. Assume that *LVor* (s_i) is infinite. Since $LVor(s_i)$ is a convex set, $LVor(s_i)$ must contain a half-infinite line starting from s_i and denoted by $\overrightarrow{s_i v}$ in Fig. 6(a). Assuming that $s_i \in CH(Neig(s_i))$, we can find a triangle $\Delta s_j s_k s_l$ such that $s_i \in \Delta s_j s_k s_l$, where $s_j, s_k, s_l \in Neig(s_i)$. Without loss of generality, we assume that $\overrightarrow{s_i v}$ intersects with $\overline{s_j s_k}$ (if $\overrightarrow{s_i v}$ goes through s_j or s_k, we can directly get a contradiction). Since $\angle s_j s_i s_k < \pi$, then $\angle s_j s_i v$ or $\angle v s_i s_k$ must be smaller than $\pi/2$. If $\angle s_j s_i v < \pi/2$, the bisector and perpendicular of $\overline{s_j s_i}$, i.e., $\overline{uh}$, will intersect with $\overline{s_i v}$ at point h. Obviously, all the points on $\overline{hv}$ will be closer to s_j than s_i, which contradicts the assumption that $\overrightarrow{s_i v} \subset LVor(s_i)$.

If $s_i \notin CH(Neig(s_i))$, then all neighbors of s_i lie in a sector with angle $\alpha < \pi$ (the shaded area in Fig. 6(b)). Let $\overrightarrow{s_i v}$ be the half-infinite line starting from s_i with angle β where $\beta + \alpha/2 = \pi$ (cf. Fig. 6(b)). Therefore, for any point h on $\overrightarrow{s_i v}$ and $s_j \in Neig(s_i)$, we can get $\|s_j h\| > \|s_i h\|$, because in $\Delta s_i s_j h$ we have $\gamma \geq \beta > \pi/2$. Hence $\overrightarrow{s_i v} \in LVor(s_i)$, which implies that $LVor(s_i)$ is infinite. □

When $LVor(s_i)$ is infinite, it cannot be fully covered by $Disk(s_i, r_s)$. From Theorem 1, we can directly conclude that s_i must be a boundary node if there is no NEP for s_i. Therefore, the correctness of the algorithm is guaranteed.

4.4 Discussions on NEP-based detection

4.4.1 NEP-based detection for area coverage

Unlike the LVP-based algorithm, the NEP-based algorithm cannot identify all the boundary nodes, which is the cost of only using directional information. The two algorithms can be combined in the following way. Since directional information is relatively easier to obtain than distance information, we assume that the former is available while the latter is determined only when necessary. In the first step, a given node checks whether it has no NEP, and if so, decides that it is a boundary node. Otherwise, this node determines the distances to neighboring nodes and then performs the LVP-based algorithm. By doing so, although both algorithms need to be executed for a few nodes, the overall energy consumption and response time may be reduced in contrast to the

[7] The antipodal point $A(s_{q+1})$ is defined as the point on $\partial Disk(s_i, r_c)$ that is on the ray staring at s_i and along the opposite direction of s_{q+1} and represented by the dotted open node in Fig. 5.

case where only the LVP-based algorithm is used, as accurate distance estimation may be both time-consuming and energy-inefficient.

When each node can only obtain neighbors's direction information, it is easy to show that it is impossible to find an algorithm to locally detect all the boundary nodes for all situations. For NEP-based algorithm, it has already done its best. However, note that only when we want to know the coverage boundaries without any distortion, is the complete information about all the boundary nodes necessary. In practice, however, several degrees of distortion on the "coverage image" is usually tolerable for the users to make the decision. Moreover, the property of the coverage boundaries (e.g., boundaries always consist of continuous closed curves) can be utilized for the users to recover some lost data about boundaries. Therefore, we can still get the key information about the coverage boundaries from partial boundary nodes detected by NEP-based algorithm. The simulation result supports our argument and is quite positive: the positions of nodes without NEPs can depict the major topology shape of the connected coverage area (see Fig. 10 in Section 5.1).

4.4.2 NEP-based detection for barrier coverage

If only directional information is available, only part of the boundary nodes can be identified. In this case, how would the coverage quality of the WSN be? Inspired by [7, 25], in the following we will show that the NEP-based algorithm is very useful for characterizing the WSN's ability of detecting any penetrating behavior of mobile object (or *intruder*) through the protected area.

Definition 6. We say an intruder *penetrates* a polygonal area A, if it enters the polygon from one side and leaves from another along a continuous path. A penetrating path is *undetectable* for $Clust(s_i)$ if it does not intersect $Cover(s_i)$ in area A. If there exists no penetrating path undetectable for area A under the node set $Clust(s_i)$, we say $Clust(s_i)$ can provide *barrier coverage* over area A.

Barrier coverage is weaker than area coverage (which requires every point in A to be covered by nodes $Clust(s_i)$), but it is widely used in many WSN applications such as intrusion detection and border surveillance [7, 25]. The goal in these applications is to detect intruders as they penetrate a protected area instead of continuously surveilling every point in the area. The following theorem shows that the information about nodes without NEPs is sufficient to characterize the barrier-coverage capability of a WSN.

Theorem 5. *Let $NNEP(s_i)$ be the set of nodes without NEPs in $Clust(s_i)$, then $Clust(s_i)$ can provide barrier coverage over the polygon $CH(NNEP(s_i))$, and the largest polygon $LP(s_i)$ over which $Clust(s_i)$ can provide barrier coverage is bounded by*

$$CH(NNEP(s_i)) \subset LP(s_i) \subset CH(NNEP(s_i)) \oplus Disk_{\mathbf{0}}. \quad (14)$$

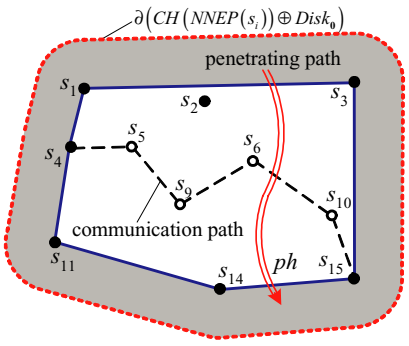


Fig. 7 Illustration of the proof of Theorem 5. Solid points represent the nodes without NEP, and solid lines represent the convex hull of all the nodes without NEP

Proof: We prove this theorem by contradiction. Assume that there is an undetectable penetrating path entering the polygon CH $(NNEP(s_i))$ from one side (side $\overline{s_1 s_3}$ in Fig. 7), and leaving from another side (side $\overline{s_{14} s_{15}}$ in Fig. 7). This penetrating path divides $NNEP(s_i)$ into two nonempty sets: $\{s_1, s_2, s_4, s_{11}, s_{14}\}$ and $\{s_3, s_{15}\}$. Let us randomly select a node from each set, say s_4 and s_{15}. Then there must exist a communication path between s_4 and s_{15} because they belong to the same cluster, i.e., $s_4 - s_5 - s_9 - s_6 - s_{10} - s_{15}$ in Fig. 7. Due to the property of the convex hull, this path at least intersects the penetrating path at one point in $CH(NNEP(s_i))$. By the definition of the communication path, all the points on it are covered by $Clust(s_i)$, including the crossing points, which contradicts the assumption that the penetrating path is undetectable. Therefore, there is no undetectable penetrating path in $CH(NNEP(s_i))$.

To give tight bounds of $LP(s_i)$, we need to prove that

$$CH(NNEP(s_i)) = CH(Clust(s_i)). \quad (15)$$

By the definition of the convex hull, node s_i is the vertex of $Clust(s_i)$ if and only if there is a supporting line such that s_i is in one side and all the other nodes $Clust(s_i) - \{s_i\}$ are in another side [4]. Therefore, the vertex of $CH(Clust(s_i))$ must be the node without an NEP and (15) holds. We have shown that $Clust(s_i)$ can provide barrier coverage over $CH(NNEP(s_i))$, thus $CH(NNEP(s_i)) \subset LP(s_i)$. Since $Cover(s_i) \subseteq CH(Clust(s_i)) \oplus Disk_{\mathbf{0}}$, we have $LP(s_i) \subset CH(Clust(s_i)) \oplus Disk_{\mathbf{0}}$. Combining this result with (15), we obtain (14). Therefore, we can bound $\partial LP(s_i)$ between $\partial CH(NNEP(s_i))$ and $\partial(CH(NNEP(s_i)) \oplus Disk_{\mathbf{0}})$ (the shaded area in Fig. 7). □

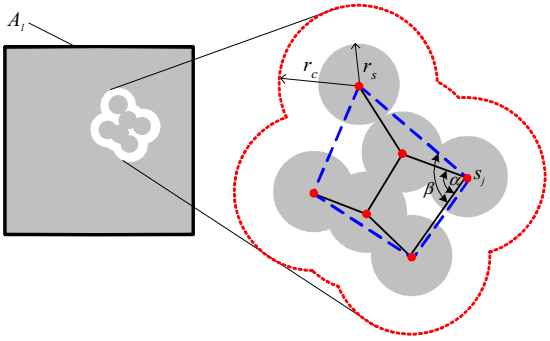


Fig. 8 Illustration of the Proof of Theorem 6, where dashed lines represent the convex hull of $Clust(s_i)$

Note that $\partial LP(s_i)$ is the boundary of barrier coverage of $Clust(s_i)$. Unfortunately, for the boundary of area coverage, i.e., $\partial Cover(s_i)$, we cannot obtain such tight bounds for all possible situations if knowing only the nodes without NEPs. Also note that Theorem 5 only holds when $r_c \leq 2r_s$, which shows a big difference between area coverage and barrier coverage.

4.4.3 NEP-based detection for connectedness

In addition to providing some important coverage information, NEPs are directly related to the network connectivity. This relationship can be used to facilitate distributed topology maintenance which is a nice feature not provided by LVPs. We first define the *sensor node on the border* ∂A_l, e.g. s_i, as the sensor which satisfies $Disk(s_i, r_s) \cap \partial A_l \neq \emptyset$.

Theorem 6. *If every sensor node (except sensor nodes on the border) has an NEP, the network is guaranteed to be connected.*

Proof: We prove this by contradiction. Suppose the network has two partitions, as shown in Fig. 8. For a node, e.g., s_j, on the boundary of the convex hull of cluster $Clust(s_j)$, all the neighbors all lie in the sector with angle α that is not greater than the angle β of the convex hull. By the definition of the convex hull, $\alpha \leq \beta < \pi$, which contradicts the assumption that s_j has an NEP and thus $\alpha > \pi$. □

Unfortunately, we cannot conclude that the network must be disconnected if one of the nodes that is not on the border does not have an NEP. Different from isolation (i.e., s_i has no neighbors), disconnection is a relationship between clusters, with which the detection cannot be completely localized. This sufficient condition, however, is still useful in preventing disconnection. Consider topology control [21] as an example. Ascertaining that all the active nodes have an NEP, we can design distributed sleep scheduling algorithms to guarantee network connectedness. As another example, in network self-monitoring applications where the BS needs a global view of the network, detecting nodes without an NEP can provide the early warning of network disconnectedness and partitions.

5 Performance evaluation

In this section, we first validate the accuracy of our algorithms by simulations. It is shown by theoretical analysis and simulations that our algorithms outperform the existing schemes in the literature in terms of energy consumption.

5.1 Validating the accuracy with simulation results

We have implemented the LVP-based and NEP-based algorithms in R [1] and tested their performance on the large-scale WSNs. Figure 9 shows the detection example of a large scale WSN with an intended attack (physically destruction such as the planned bombing of the WSN). The intention of adversary is very clear, by destroying parts of sensor nodes, disconnecting the WSN, and making a large part of the WSN lose its function. Figure 9 gives some snap-shots of this process, and the detection results of each state. The ROI is a square with the size of $14r_c \times 14r_c$ and originally deployed sensors completely covered the ROI (see Fig. 9(a)). The boundary nodes detected by the NEP-based algorithm are displayed as darkened dots, and the additional ones detected by the LVP-based algorithm are displayed as lightly shaded dots. In addition, the theoretical coverage boundary is formed by solid lines. It can be found that all the boundary nodes

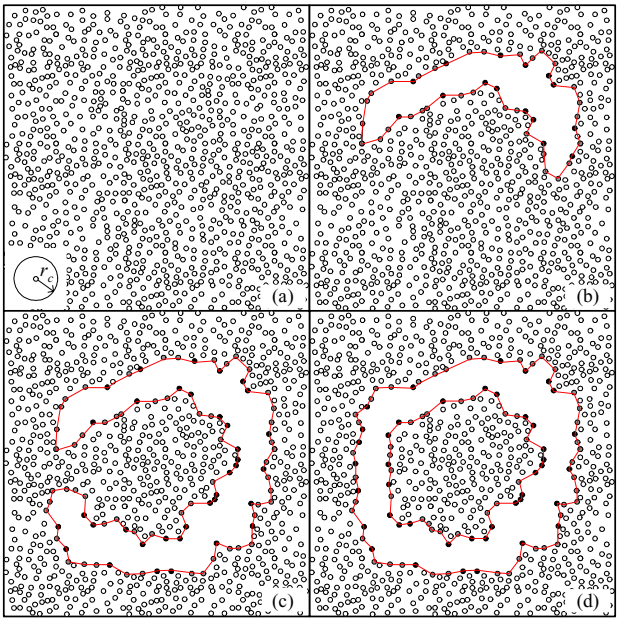


Fig. 9 Boundary detection results for large-scale WSNs

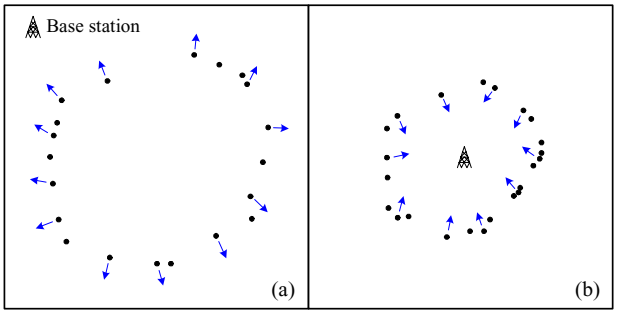


Fig. 10 Coverage boundary reconstruction results on the base station with the NEP-based algorithm for the situation shown in Fig. 9(d)

are correctly detected. The effectiveness of our algorithms is quite obvious.

As shown above, the LVP-based algorithm can detect all the boundary nodes and give perfect information about coverage boundaries. Although the NEP-based algorithm cannot detect all the boundary nodes, it can still offer very useful information. Consider Fig. 10 as an example. All the boundary nodes determined by the NEP-based algorithm send their positions and the directions of the areas that are not covered to the sink, which in turn, can reconstruct the coverage boundary based on such information. Figures 10(a) and (b) show the "images of the boundary" when the sink lies in the margin and center of the ROI, respectively. Such results are quite positive because although the details of the boundary are lost, the outline can be obtained.

5.2 Cost analysis

5.2.1 LVP-based approach vs. perimeter-based approach

After the deployment of the WSN, we assume localization techniques are available for sensor nodes to decide their positions. Each node then collects the position information of its neighbors by broadcasting its own positions. Since the connected coverage will change with time, each node needs to check the existence of its neighbors periodically. Our LVP-based approach and the perimeter-based approach can both provide truly localized boundary node detection with operational difference in the neighborhood checking phase. In particular, the perimeter-based approach requires each node to check the status of all its neighbors, which is quite inefficient when sensor nodes are densely deployed. This situation becomes worse every time when a node dies, as all its neighbors need recheck the coverage of their perimeters or crossings. In contrast, our LVP-based approach only uses consulting neighbors to perform boundary node detection. When sensor nodes are densely deployed, from Lemma 1, we have $LVor(s_i)$ or $TLVor(s_i) \rightarrow Vor(s_i)$, and it is well known in computational geometry that under the homogeneous SPPP, the average number of vertices of $Vor(s_i)$ is 6 [27], which implies that when the node density is high, each node on average only has 4 to 6 consulting neighbors. Therefore, the higher the node density, the greater the benefit using our scheme.

5.2.2 LVP-based approach vs. VP-based approach

Intuitively, LVP-based approach will have smaller communication overhead or equivalently energy consumption than the VP-based method since LVPs can be locally computed. In what follows, we prove this intuition in a formal way.

Theorem 7. *If there exist boundary nodes, the costs of the NEP-based and LVP-based algorithms are always smaller than the cost of the VP-based one.*

The proof of the theorem depends on the following lemma:

Lemma 3. For any $s_i \in V$, the VP $Vor(s_i)$ can be locally computed if and only if $Clust(s_i)$ can completely cover the plane $\mathbb{R}^2$ (or A_l, when the information of ∂A_l is available), i.e., $Cover(s_i) = \mathbb{R}^2$ (or $Cover(s_i) \cap A_l = A_l$).

Proof: From Theorems 1 and 2, a node set can completely cover $\mathbb{R}^2$ if and only if $LVor(s_i)$ is fully covered by $Disk(s_i, r_s)$ for any $s_i \in V$. From Lemma 1, this implies that $Vor(s_i) = LVor(s_i)$ for any $s_i \in V$. Therefore, $Vor(s_i)$ can be locally computed by s_i just as $LVor(s_i)$.

Let $d = \max \|v - s_i\|$ for any $v \in Vor(s_i)$. Since $Vor(s_i)$ is a convex set, $d = \infty$ if $Vor(s_i)$ is infinite, otherwise d is the distance from a vertex of $Vor(s_i)$ to s_i. $Vor(s_i)$ can also be computed in a similar way as LVP with set V as the input. We can determine that the construction of $Vor(s_i)$ is completed when all the nodes in $Disk(s_i, 2d)$ have been counted. Therefore, $Vor(s_i)$ can be locally computed, which implies that $2d \leq r_c$ or $d \leq r_s$ and thus guarantees the complete coverage of $Vor(s_i)$. Since this holds for all $s_i \in V$, we can ensure the complete coverage of the plane. □

Therefore, when there are boundary nodes, it is impossible to compute all $Vor(s_i)$'s locally based on only one-hop information. Since multi-hop communications are unavoidable, the cost of the VP-based approach will be higher than both LVP-based and NEP-based algorithms. Only when the node density is so high that the ROI is completely covered (not considering the ROI border), is the cost of the VP-based approach equal to that of ours. However, in this case, there is no need for coverage boundary detection at all. As a result Theorem 7 guarantees that when boundary detection algorithms are helpful, the cost of our algorithms is definitely smaller than the VP-based one. The next question is

how significant the cost savings are by using our algorithms, which is answered in the rest of this section.

5.3 Evaluation of energy consumption for VP- and LVP-based approachs

5.3.1 Evaluation settings

We assume that sensor nodes are distributed in a large square region A_l and form a homogeneous *Spatial Poisson Point Process* (SPPP) with density λ. Each node knows its own position by GPS or existing localization schemes such as [23]. For any measurable subset of A_l with area B,

$$\Pr\{\text{finding } i \text{ nodes in the region of area } B\} = \frac{(\lambda B)^i e^{-\lambda B}}{i!}.$$

Each node is expected to have $k = \pi r_c^2 \lambda$ neighbors on average, and the expected number of nodes in A_l is given by $n = \lambda \cdot A_l$. We also assume that each node fails independently and uniformly with probability p. It has been shown that functional nodes still form a homogeneous Poisson point process with density $\lambda' = (1-p)\lambda$ [31]. Therefore, the network can be uniquely identified by the current node density λ (or equivalently k).

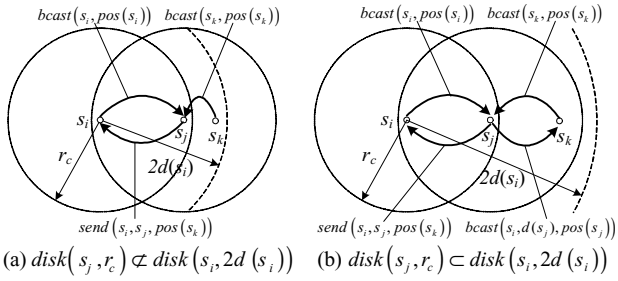
In general, no assumption should be made about the distribution of the sensor nodes in the environment. Our algorithms are designed to work correctly under arbitrary node distributions. However, here we utilize homogeneous SPPP as the node distribution model to facilitate the theoretical analysis and simulations in the performance evaluation of our algorithms. It is well known that this model is a good approximation of the distribution of sensor nodes massively or randomly deployed (e.g, via aerial scattering or artillery launching) and can be easily extended to characterize the process that nodes fail dynamically. In addition, we let the side length $l \to \infty$ (which implies $n = \lambda A_l \to \infty$). By doing so, we can infer the characteristics of the whole network by just analyzing some "typical nodes" (which are far away from ∂A_l) and ignoring the "boundary effects" [29].

Based on the continuum percolation theory [26, 29], if $k \leq 4.5$, a 2-D network will be partitioned into $O(n)$ small clusters, which implies that the WSN will completely fail. Previous work [14, 21] also points out that for $n \to \infty$, the ROI A_l is completely covered with a high probability when λ and k satisfy

$$\pi r_s^2 \lambda = \pi (r_c/2)^2 \lambda = k/4 = \log(n) + \log\log(n). \tag{16}$$

When k is greater than this critical value, it is guaranteed that there is no coverage hole in the network. Therefore, in what follows we will just consider the case when

$$4.5 < k < 4\log(n) + 4\log\log(n). \tag{17}$$



(a) $disk(s_j, r_c) \not\subset disk(s_i, 2d(s_i))$ (b) $disk(s_j, r_c) \subset disk(s_i, 2d(s_i))$

Fig. 11 Illustration of the communication procedure in VP/LVP computation

In our evaluation, in order to have a fair comparison, VPs are computed in a similar way as LVPs with set V as the input. Specifically, we first compute $LVor(s_i)$ as the tentative VP of s_i, and then refine the tentative VP iteratively. In each iteration, we add one more hop information about node positions. Let $d(s_i) = \max \|v - s_i\|$, for any $v \in LVor(s_i)$ ($d(s_i) = \infty$ when $LVor(s_i)$ is infinite). Obviously, to guarantee the accuracy of the results, we only need to check the nodes in the region $Disk(s_i, 2d(s_i))$.

To facilitate the analysis, we also assume that communications proceed in rounds (governed by a global clock) with each round taking one time unit, and that there are effective MAC-layer protocols supporting reliable communications. Let E_T and E_R denote the energy consumed to transmit and receive one bit, respectively, and S_M be the size of message M in bits. The procedure for computing VPs/LVPs/NEPs is shown below (cf. Fig. 11).

Step 1. Every node s_i broadcasts its position s_i, and receives its neighbors' position messages. This step is enough for computing LVPs and NEPs, and the energy consumption of node s_i is $(E_T + kE_R)S_{s_i}$. To compute VPs, we still need to do the the following steps:

Step 2. Node s_i computes its tentative VP $LVor(s_i)$ with position of s_j where $s_j \in Neig(s_i)$, and then broadcasts $d(s_i)$ if $d(s_i) > r_c$. The energy consumption of s_i in this step is $(E_T + kE_R)S_{d(s_i)}$.

Step 3. Upon receiving any $d(s_j)$, node s_i checks the positions of its neighbors. If there is a node s_k such that

$$s_k \in Disk(s_j, r_c)^{\mathsf{C}} \cap Disk(s_j, 2d(s_j)) \cap Disk(s_i, r_c),$$

node s_i reports s_k's position to node s_j. If $Disk(s_i, r_c) \subset Disk(s_j, 2d(s_j))$, node s_i still needs to broadcast $d(s_j)$ and s_j's position.

Step 4. repeat step 3 until $Disk(s_i, r_c) \not\subset Disk(s_j, 2d(s_j))$.

If constructing a VP needs m-hop information, the total energy consumption in steps 3 and 4 will be

$$(m-1)(E_T + kE_R)\big(S_{d(s_j)} + S_{s_j}\big). \tag{18}$$

5.3.2 Theoretical results

Given the density λ, from the proof of Lemma 3, $2d(s_i)$ can be computed from (16) as:

$$2d(s_i) = \left(\frac{4\log(n) + 4\log\log(n)}{\pi\lambda}\right)^{1/2}. \tag{19}$$

The next step is to compute the number of hops needed to reach $2d(s_i)$. For the homogeneous Poisson point process, hop-distance relationship has been derived in [9]:

$$\mathbf{E}(m) = \begin{cases} 1, d \le r_c \\ 0.5 + h \cdot c, d > r_c \end{cases} \tag{20}$$

where $d = h \cdot r_c$ is the distance to reach, $\mathbf{E}(m)$ is the corresponding expected number of hops, and c is a constant that is close to two for a small k and to one as k becomes large. Therefore, the number of hops needed for $2d(s_i) > r_c$ can be calculated as:

$$\mathbf{E}(m) = \frac{1}{2} + \frac{c}{r_c}\left(\frac{4\log(n) + 4\log\log(n)}{\pi\lambda}\right)^{1/2}. \tag{21}$$

The energy consumption of a typical node using the LVP-based or NEP-based algorithm is

$$EC_{LVP} = (E_T + kE_R)S_{s_i}. \tag{22}$$

It is difficult to precisely compute the total energy consumption for transmitting position information of sensor nodes in region $Disk(s_i, r_c)^C \cap Disk(s_i, 2d(s_i))$ to s_i. One way to handle this problem is to estimate it by the last hop energy consumption:

$$EC_{L\_Hop} = (E_T + E_R)\big(\pi(2d(s_i))^2\lambda - \pi r_c^2\lambda\big)S_{s_i}. \tag{23}$$

Therefore, the energy consumption for a typical node using the VP-based algorithms will not be less than

$$\begin{aligned} EC_{VP} = {} & EC_{LVP} + \mathbf{E}(m)(E_T + kE_R)S_{d(s_i)} \\ & + (\mathbf{E}(m) - 1)(E_T + kE_R)S_{s_i} + EC_{L\_Hop}. \end{aligned} \tag{24}$$

For $4.5 < k < 4\log(n) + 8\log\log(n)$, we have $\mathbf{E}(m) \ge 1.5$. Since $d(s_i)$ is 1-D while s_i) is 2-D data, we assume that $S_{s_j} = 2S_{d(s_j)}$. Then we can get

$$\frac{EC_{VP}}{EC_{LVP}} > 2.75. \tag{25}$$

Obviously, the energy savings are significant. Note that (25) holds for all $4.5 < k < 4\log(n) + 8\log\log(n)$. For a network with an inhomogeneous point distribution, we can divide the network into a finite number of partitions with different constant densities. If the densities are all in $(4.5, 4\log(n) + 8\log\log(n))$, the inequality (25) still holds.

5.4 Simulation results

5.4.1 LVP-based approach vs. VP-based approach

We simulate a WSN with $l = 200$ m, $r_c = 20$ m, $S_{s_j} = 64$ bytes, $E_R = 0.6\,\mu$J/bits, $E_T = 0.8\,\mu$J/bit, and $4.5 < k < 45$ (the range of k is derived from the Eq. (17)). Figure 12 shows the average node energy consumption for the VP-based (EC_{VP}) and the LVP-based or NEP-based (EC_{LVP}) algorithms as a function of k. We can see that the theoretical values of EC_{LVP} are always greater than the simulation results. The reason is that we treat nodes on the ROI boundary as typical nodes in theoretical analysis, while these nodes in fact have much fewer neighbors and thus have less energy consumption. In addition, both theoretical and simulation results of EC_{LVP} slightly increase as k becomes large, as the reception energy consumption increases with the increasing number of neighbors. By contrast, the theoretical results of EC_{VP} are always lower than the simulation results because of the approximation in (23). We can also observe that the difference becomes smaller with the increase of k. This is due to the fact that the number of hops needed to reach $2d(s_i)$ will become smaller with increasing k and our approximation will make more sense. In general, it is obvious that our LVP-based algorithms can achieve remarkable energy savings.

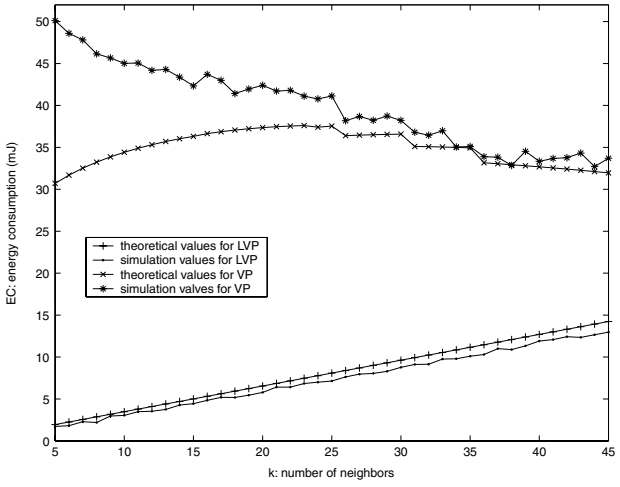


Fig. 12 Energy consumption for the LVP- and VP-based algorithms

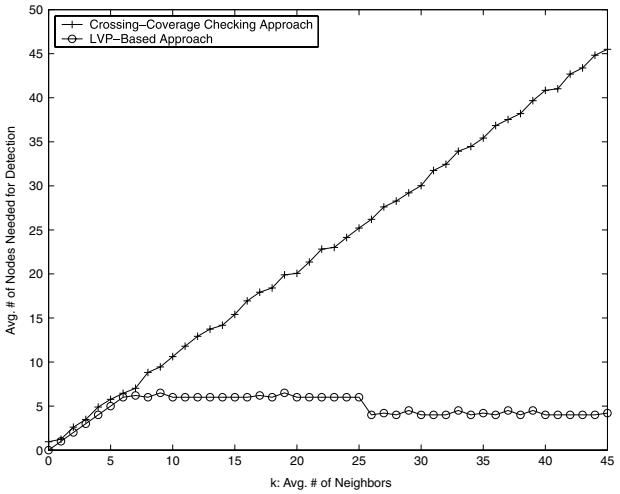


Fig. 13 Average number of neighbor nodes needed for the crossing-coverage checking approach and LVP-based approach

5.4.2 LVP-based approach vs. perimeter-based approach

The simulation settings are the same as above. Figure 13 shows the average number of neighbor nodes needed for the crossing-coverage checking approach and our LVP-based approach to detect boundary nodes as a function of k. It is of no surprise to observe that when the node density increases, the number of nodes needed remains constant for our LVP-based detection while increases dramatically for the crossing-coverage checking approach. This means that, in contrast to our approach, the crossing-coverage checking approach will incur a significant overhead at the initial stage of WSNs where sensor nodes are normally densely deployed to provide adequate redundancy and fault-tolerance.

To sum up, the VP-based approaches only perform well when functional nodes are densely deployed; the perimeter-based approaches only work well when functional nodes are sparsely deployed; and only our LVP-based approach works equally well in both cases.

6 Conclusion

In this paper, we develop two deterministic, localized algorithms for coverage boundary detection in WSNs. Our algorithms are based on two novel computational geometric techniques, namely, localized Voronoi and neighbor embracing polygons. Theoretical analysis and simulation results show that, our algorithms can be applied to WSNs of arbitrary topologies with varying node densities and have the minimal computation and communication costs, as compared to previous proposals.

Appendix A: Remarks on disk sensing and communication models

In this paper, we assume that both the sensing and communication ranges are regular disks (cf. Section 2.1). In practice, however, it is well known that both the sensing and the communication ranges are non-isotropic, i.e., sensors exhibit different ranges (in both sensing and communication) in different directions [6]. According to [6], the sensing range of the passive infrared (PIR) sensors in different directions roughly conforms to a normal distribution probability model. For the communication range, two non-isotropic models—the Degree of Irregularity (DOI) model and the Radio Irregularity (RIM) model—are presented in [36].

However, regular disk models are widely used in studying coverage and connectivity properties of WSNs as in [3, 12, 15, 20, 21, 33]. We still follow this approach for the following two reasons:

First, in the cases where the irregular sensing and communication ranges each has a lower bound, the sensing and communication areas can be regarded as a disk with radius equal to the lower bound. With this approach, our schemes can provide a conservative estimation on the connected coverage deterministically. For more general cases, our work can be further extended to provide statistical coverage inference. Since our BOND scheme detects boundary nodes based on the real communications between neighbors, the only thing we need to change is to adopt a statistical sensing range model such as the one proposed in [6] and estimate the corresponding statistical connected coverage.

Second, our schemes target at large-scale WSNs, where r_c and r_s are relatively very small compared to the size of ROI. Therefore, in general the inference error introduced by our disk models is negligible compared to the total area of connected coverage. The validity of disk model for large-scale WSNs is intensively studied in the literature. It is shown in [12] that claims about the connectivity and coverage made under this "disk" abstraction generally also hold for more irregular sensing and communication shapes found in practice.

Appendix B: Locality of boundary node detection

In this subsection we investigate further to show that it is impossible to find localized algorithms for boundary node detection with arbitrary node distributions when $r_c/r_s < 2$. We first define what we mean by localized algorithms. This definition is based on a model proposed in [28].

Definition 7. Assume that each computation step takes one unit of time and so does every message to get from one node to its directly connected neighbors. With this model, an

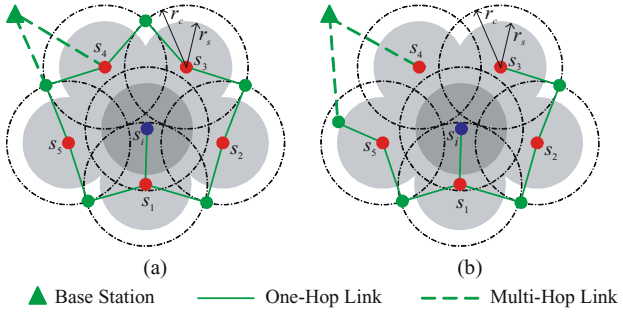


Fig. 14 Non-locality of the boundary node detection when $r_c < 2r_s$

algorithm is called *localized* if its computation time is $O(1)$, in terms of the number of nodes n in the system.

Our LVP-based algorithm shows that when $r_c = 2r_s$, sensors can locally determine if it is a boundary node. When $r_c > 2r_s$, since the node will have more information about other nodes around itself, it can still locally detect whether it is a boundary node. However, in the case of $r_c < 2r_s$, individual nodes can neither locally say "yes" nor "no" to the question of whether a given node is a boundary node. To see this, consider sensors deployed as in Fig. 14. Obviously, node s_i is an interior node. However, to confirm this, it needs to know the existence of nodes s_1 to s_5 with the help of some relay nodes (lightly shaded nodes). In Fig. 14(a), node s_4 is already 5 hops away from node s_i. In fact the distance between these two nodes can be arbitrary long, which is shown in Fig. 14(b). Therefore, for arbitrary node distributions, it is impossible to find a localized boundary node detection algorithm that always works. In [3], the authors considered general values of r_c/r_s with regular deployment patterns such as the hexagon, square grid, rhombus, and equilateral triangle. Unlike [3], in this paper, we prefer to make the strict assumption on the value of r_c/r_s rather than on the node distribution pattern. The reason is that even if in some scenarios, the regular distribution of nodes is possible, it is difficult to hold in practice due to inevitable node failures. In contrast, even in the case of $r_c/r_s < 2$, we can still assume a smaller value of r_s in our algorithms while obtaining a conservative inference of the coverage, which is desirable for some security-critical applications.

Acknowledgments This work was supported in part by the U.S. National Science Foundation under Grant ANI-0093241 (CAREER Award) and Grant DBI-0529012.

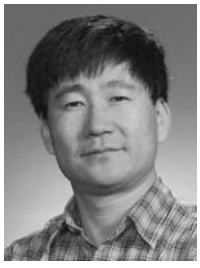
References

1. R Project. http://www.r-project.org/.
2. I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, Wireless sensor networks: A survey, Computer Networks (Elsevier) Journal 38(4) (2002) 169–181, 393–422.
3. X. Bai, S. Kumar, D. Xuan, Z. Yun and T. Lai, Deploying wireless sensors to achieve both coverage and connectivity, in: *MobiHoc'06*, Florence, Italy (May 2006).
4. M. Berg, *Computational Geometry: Algorithms and Applications* (Springer, New York, 2000).
5. S. Borbash and E. Jennings, Distributed topology control algorithm for multihop wireless networks, in: *Proc. IEEE International Joint Conference on Neural Networks*, Honolulu, HI (April 2002).
6. Q. Cao, T. Yan, J.A. Stankovic and T.F. Abdelzaher, Analysis of target detection performance for wireless sensor networks, in: *International Conference on Distributed Computing in Sensor Networks (DCOSS)*, Los Angeles, CA (June 2005).
7. M. Cardei and J. Wu, *Handbook of Sensor Networks*, chapter Coverage in Wireless Sensor Networks (CRC Press, Boca Raton, FL, 2004).
8. M. Chan, D. Chen, F.Y. Chin and C. Wang, Construction of the nearest neighbor embracing graph of a point set, in: *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory*, Humlebak, Denmark (July 2004).
9. S. Chandler, Calculation of number of relay hops required in randomly located radio network, Electronics Letters 25(24) (1989) 1669–1671.
10. S. Chiu and I. Molchanov, A new graph related to the directions of nearest neighbours in a point process, Advances in Applied Probability 35(1) (2003) 47–55.
11. Q. Fang, J. Gao and L. Guibas, Locating and bypassing routing holes in sensor networks, in: *INFOCOM'04*, Hong Kong, China (March 2004).
12. M. Franceschetti, L. Booth, M. Cook, R. Meester and J. Bruck, *Percolation of Multi-Hop Wireless Networks*, Technical Report UCB/ERL M03/18, EECS Department, University of California, Berkeley (2003).
13. A. Ghosh, Estimating coverage holes and enhancing coverage in mixed sensor networks, in: *LCN'04*, Washington, DC (November 2004).
14. P. Gupta and P. Kumar, The capacity of wireless networks, *IEEE Transactions on Information Theory* 46(2) (2000) 388–404.
15. P. Gupta and P.R. Kumar, *Stochastic Analysis, Control, Optimization and Applications: A volume in honor of W.H. Fleming*, chapter Critical Power for Asymptotic Connectivity in Wireless Networks, Birkhauser, Boston (1998).
16. P. Hall, *Introduction to the Theory of Coverage Processes* (John Wiley Sons Inc., New York, 1988).
17. C.F. Huang and Y.C. Tseng, The coverage problem in a wireless sensor network, in: *Proc. of the 2nd ACM international Workshop on Wireless Sensor Networks and Applications (WSNA '03)*, San Diego, CA (September 2003).
18. S. Kapoor and X. Li, Proximity structures for geometric graphs, in: *WADS 2003*, Ottawa, Canada (July 2003).
19. H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks* (John Wiley Sons Inc., New York, 2005).
20. S. Kumar, T.H. Lai and A. Arora, Barrier coverage with wireless sensors, in: *MobiCom'05*, Cologne, Germany (August 2005).
21. S. Kumar, T.H. Lai and J. Balogh, On k-coverage in a mostly sleeping sensor network, in: *MobiCom'04*, Philadelphia, PA (October 2004).
22. M. Leonov, Polyboolean library (2004), http://www.complex-a5.ru/polyboolean/.
23. L. Li, J. Halpern, P. Bahl, Y. Wang and R. Wattenhofer, A cone-based distributed topology-control algorithm for wireless multi-hop networks, *IEEE/ACM Transactions on Networking* 13(1) (2005) 147–159.
24. X. Li, G. Calinescu, P. Wan and Y. Wang, Localized delaunay triangulation with applications in wireless ad hoc networks, IEEE Transactions on Parallel and Distributed Systems 14(10) (2003) 1035–1047.
25. B. Liu and D. Towsley, A study of the coverage of largescale sensor networks, in: *IEEE 47th Annual International Vehicular Technology Conference*, Fort Lauderdale, FL (October 2004).

26. R. Meester and R. Roy, *Continuum Percolation* (Cambridge University Press, Cambridge, 1996).
27. A. Okabe, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams* (Wiley, New York, 2000).
28. D. Peleg, *Distributed Computing: A Loaclity-Sensitive Approach*, Society for Industrial and Applied Mathematics (SIAM) (2000).
29. M. Penrose, *Random Geometric Graphs* (Oxford University Press, Oxford, 2003).
30. I. Stojmenovic and X. Lin, Gedir: Loop-free location based routing in wireless networks, in: *Proc. of International Conference on Parallel and Distributed Computing and Systems*, Boston, MA (November 1999).
31. D. Stoyan, W. Kendall and J. Mecke, *Stochastic Geometry and its Applications*, 2nd edn. (Wiley, New York, 1995).
32. G. Wang, G. Cao and T.L. Porta, Movement-assisted sensor deployment, in: *INFOCOM'04*, Hong Kong, China (March 2004).
33. X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless and C. Gill, Integrated coverage and connectivity configuration in wireless sensor networks, in: *ACM SenSys'03*, Los Angeles, CA (November 2003).
34. C. Zhang, Y. Zhang and Y. Fang, Detecting boundary nodes in wireless sensor networks, in: *Proc. of 2006 IEEE International Conference On Networking, Sensing and Control*, Ft. Lauderdale, Florida (April 2006).
35. H. Zhang and J. Hou, Maintaining sensing coverage and connectivity in large sensor networks, Wireless Ad Hoc and Sensor Network 1(1–2) (2005) 89–123.
36. G. Zhou, T. He, S. Krishnamurthy and J.A. Stankovic, Impact of radio irregularity on wireless sensor networks, in: *ACM MobiSys*, Boston, MA (June 2004).

Chi Zhang received the B.E. and M.E. degrees in Electrical Engineering from Huazhong University of Science and Technology, Wuhan, China, in July 1999 and January 2002, respectively. Since September 2004, he has been working towards the Ph.D. degree in the Department of Electrical and Computer Engineering at the University of Florida, Gainesville, Florida, USA. His research interests are network and distributed system security, wireless networking, and mobile computing, with emphasis on mobile ad hoc networks, wireless sensor networks, wireless mesh networks, and heterogeneous wired/wireless networks.

Yanchao Zhang received the B.E. degree in computer communications from Nanjing University of Posts and Telecommunications, Nanjing, China, in July 1999, the M.E. degree in computer applications from Beijing University of Posts and Telecommunications, Beijing, China, in April 2002, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, in August 2006. Since September 2006, he has been an Assistant Professor in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark. His research interest include wireless and Internet security, wireless networking, and mobile computing. He is a member of the IEEE and ACM.



Yuguang Fang received the BS and MS degrees in Mathematics from Qufu Normal University, Qufu, Shandong, China, in 1984 and 1987, respectively, a Ph.D. degree in Systems and Control Engineering from Department of Systems, Control and Industrial Engineering at Case Western Reserve University, Cleveland, Ohio, in January 1994, and a Ph.D. degree in Electrical Engineering from Department of Electrical and Computer Engineering at Boston University, Massachusetts, in May 1997.

From 1987 to 1988, he held research and teaching position in both Department of Mathematics and the Institute of Automation at Qufu Normal University. From September 1989 to December 1993, he was a teaching/research assistant in Department of Systems, Control and Industrial Engineering at Case Western Reserve University, where he held a research associate position from January 1994 to May 1994. He held a post-doctoral position in Department of Electrical and Computer Engineering at Boston University from June 1994 to August 1995. From September 1995 to May 1997, he was a research assistant in Department of Electrical and Computer Engineering at Boston University. From June 1997 to July 1998, he was a Visiting Assistant Professor in Department of Electrical Engineering at the University of Texas at Dallas. From July 1998 to May 2000, he was an Assistant Professor in the Department of Electrical and Computer Engineering at New Jersey Institute of Technology, Newark, New Jersey. In May 2000, he joined the Department of Electrical and Computer Engineering at University of Florida, Gainesville, Florida, where he got early promotion to Associate Professor with tenure in August 2003, and to Full Professor in August 2005. His research interests span many areas including wireless networks, mobile computing, mobile communications, wireless security, automatic control, and neural networks. He has published over one hundred and fifty (150) papers in refereed professional journals and conferences. He received the National Science Foundation Faculty Early Career Award in 2001 and the Office of Naval Research Young Investigator Award in 2002. He also received the 2001 CAST Academic Award. He is listed in Marquis Who's Who in Science and Engineering, Who's Who in America and Who's Who in World.

Dr. Fang has actively engaged in many professional activities. He is a senior member of the IEEE and a member of the ACM. He is an Editor for IEEE Transactions on Communications, an Editor for IEEE Transactions on Wireless Communications, an Editor for IEEE Transactions on Mobile Computing, an Editor for ACM Wireless Networks, and an Editor for IEEE Wireless Communications. He was an Editor for IEEE Journal on Selected Areas in Communications: Wireless Communications Series, an Area Editor for ACM Mobile Computing and Communications Review, an Editor

for Wiley International Journal on Wireless Communications and Mobile Computing, and Feature Editor for Scanning the Literature in IEEE Personal Communications. He has also actively involved with many professional conferences such as ACM MobiCom'02 (Committee Co-Chair for Student Travel Award), MobiCom'01, IEEE INFOCOM'06, INFOCOM'05 (Vice-Chair for Technical Program Committee), INFOCOM'04, INFOCOM'03, INFOCOM'00, INFOCOM'98, IEEE WCNC'04, WCNC'02, WCNC'00 (Technical Program Vice-Chair), WCNC'99, IEEE Globecom'04 (Symposium Co-Chair), Globecom'02, and International Conference on Computer Communications and Networking (IC3N) (Technical Program Vice-Chair).