Localized Coverage Boundary Detection for Wireless Sensor Networks

Chi Zhang, Yanchao Zhang, Yuguang Fang
Department of Electrical and Computer Engineering
University of Florida, Gainesville, FL 32611
Email: {zhangchi@, yczhang@, fang@ece.}ufl.edu

***Abstract*— Connected coverage, which reflects how well a target field is monitored under the base station, is the most important performance metrics used to measure the quality of surveillance that wireless sensor networks (WSNs) can provide. To facilitate the measurement of this metrics, 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. As compared to previous work, our algorithms can be applied to WSNs of arbitrary topologies. They are also truly distributed and localized by merely needing the minimal position information of one-hop neighbors and a limited number of 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 simulations.**

I. Introduction

Wireless sensor networks (WSNs) are ideal candidates to monitor the physical space and enable a variety of applications such as battlefield surveillance, environment monitoring and biological detection, etc. 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 of interest (e.g., emergence of enemy's tanks, detonating a radiological dispersion device). Typically, each sensor node has a very limited sensing range within which it is able to perform its sensing operation, and 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, analyzes this data to draw conclusions about the activities in the ROI, and serves as the only bridge to connect the WSN with outside users [2].

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 application 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 base station that "none of the enemy's tanks has been observed in the ROI" is misleading if it is not reinforced with the description of the current connected coverage. Since 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 [8]. The reason is that overlooking coverage boundaries may cause problems in communication, 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 [13]. 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 [21], identifying coverage boundaries among randomly deployed static nodes would help determine movement strategies of mobile sensors to improve network coverage.

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 overage 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 based on *localized Voronoi polygon* (LVP) and *neighbor embracing polygon* (NEP) for coverage boundary detection in WSNs. In our scheme, the LVP-based algorithm requires both the directional information (the orientation of each neighbor) and distance information (the distance to each neighbor), and theoretically can detect all the boundary nodes no matter

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

QShine'06 The Third International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks
August 7-9 2006, Waterloo, ON, Canada © 2006 ACM 1-59593-472-3/06/08…$5.00

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. Both algorithms can be applied to WSNs of arbitrary topologies. They are also truly distributed and localized by merely needing the minimal position information of one-hop neighbors and a few simple local computations, and thus are of high scalability and energy efficiency.

II. Preliminaries

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

A. Notation and Assumption

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 $u, v \in \mathbb{R}^2$.
- r_c: the maximum 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 $Disk_{\mathbf{0}} = Disk(\mathbf{0}, r_s) = \left\{v : \|v-\mathbf{0}\| \le r_s, v \in \mathbb{R}^2\right\}$.
- *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 $A_u = A \oplus \{u\}$.

We assume that sensor nodes are homogeneous, i.e., r_c and r_s are the same for all nodes. We also assume that any two nodes can directly exchange messages if their Euclidean distance is not greater than r_c; 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.

For convenience only, we set $r_c = 2r_s$ throughout the rest of the paper. There are two reasons for doing so. First, it can be proved that, for arbitrary spatial distributions of sensor nodes, boundary nodes can be locally detected if and only if $r_c \ge 2r_s$. Therefore, we set $r_c = 2r_s$ to reduce energy consumption and interference. Second, as pointed out in [23], the specification of $r_c = 2r_s$ holds for most commercially available sensors such as Berkeley Motes. However, it should be noted that our algorithms are still applicable to the scenarios of $r_c > 2r_s$.

B. Network Model

For simplicity, we focus on a 2-D square planar field to be monitored (i.e., ROI) 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 l centered at the origin, i.e., $A_l = [-l/2, l/2]^2$, and ∂A_l be the *border* of A_l. We consider a network made of stationary sensor nodes only, and denote the sensor nodes which are deployed in the ROI to be $V = \{s_1, \cdots, s_i, \cdots, s_n\}$ ($s_i \in A_l$, for $1 \le i \le n, i \in \mathbb{N}$), where s_i represents the position of node i and n is the total number of sensor nodes, or *network size*.

We say that nodes s_i and s_j ($i \ne j$ and $s_i, s_j \in V$) are *one-hop neighbors* (or neighbors for short) if and only if the Euclidean distance between them is no larger than r_c, i.e., $\|s_i - s_j\| \le r_c$. We denote by $Neig\,(s_i)$ the neighbors of node s_i (not including s_i). We say there exists a *direct wireless links* between two nodes if and only if they are neighbors. Two nodes s_i and s_j are *connected* if there is at least one *path* consisting of direct wireless links between them. A set of nodes is called *connected* if at least one path exists between each pair of nodes in the set.

C. 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 it 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

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

We claim A_l as being *completely covered* if there is at least one cluster $Clust(s_i)$ whose nodes can cover every point in A_l, namely, $Cover(s_i) = A_l$.

***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 denoted them by

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

Accordingly, *interior nodes* is defined by

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

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

$$\begin{aligned} Cover(BS) = \Big\{ \bigcup_{1 \le i \le n} Cover(s_i) : \\ BS \bigcap \left(Clust(s_i) \oplus Disk(\mathbf{0}, r_c) \right) \ne \emptyset \Big\}. \end{aligned} \quad (4)$$

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 the Def. 2, we have $Cover(s_i) \cap Cover(s_k) = \emptyset$. Therefore,

$$\begin{aligned} \partial Cover(BS) = \big\{ \bigcup \partial Cover(s_i) : \\ Clust(s_i) \text{ is connected with } BS \big\} \end{aligned} \quad (5)$$

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

D. 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 [8], [9], [11], [12], [21], [23]. We classify them as perimeter-based and polygon-based approaches.

Perimeter-Based Approaches. The first localized boundary node detection algorithm[1] is proposed in [12], which is based on the information about the coverage of the perimeter of each node's sensing disk[2]. It can be shown that node s_i is a boundary node if and only if at least there exists 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 $\mathrm{O}(k\log k)$ is designed in [12] to locally check whether one node is a boundary node, when k is the number of neighbors. Crossing-coverage checking approach proposed in [11], [23] further simplifies 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 at least there exists one crossing $v \in \partial Disk(s_i, r_s) \cap \partial Disk(s_j, r_s)$ which is not covered by any other $s_k \in Neig(s_i) - \{s_j\}$. Fig. 1 (b) shows an example where c is a crossing determined by two perimeters $\partial Disk(s_i, r_s)$ 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 need to check positions and status of all of its neighbors, which is inefficient when the sensor nodes are densely deployed (cf. Section V) and every time when a node dies, all its neighbors need to check the coverage of their perimeters or crossings again.

Polygon-Based Approaches. In [8], [9], [21], *Voronoi diagram* (VP) is used for coverage boundary detection. Briefly speaking, the VP 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

[1]Localized boundary node detection algorithm in this paper is defined as the distributed detection algorithm which can be implemented on each node with the cooperation of its one-hop neighbors or neighbors within the range of constant number of hops.

[2]This information can be locally gathered when $r_c \geq 2r_s$.

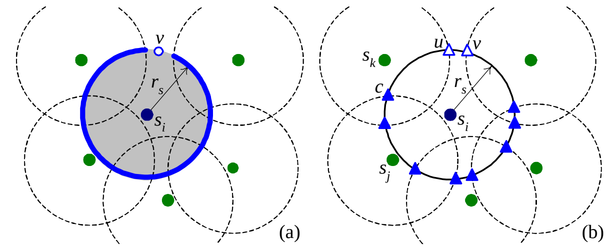

Fig. 1. Perimeter-based boundary node detection approaches. (a) Perimeter-coverage checking approach proposed in [12]. The solid curve represents the portion of perimeter of sensing disk covered by neighbor nodes. (b) Crossing-coverage checking approach proposed in [11], [23]. Solid and open triangles represent covered and uncovered crossings, respectively.

covered by any other node, which implies a coverage hole. Therefore, it is claimed in [8], [9], [21] that each node can locally check whether it is on the coverage boundary with the assumption that VPs can be derived locally. However, it has been shown that the VPs of boundary nodes cannot be locally computed [22]. Therefore, 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 changed.

There is a trend in the literature to provide some basic functionalities of WSNs by only using directional information [4], [15], [19]. 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 this topic in Section IV to propose a solution for this scenario.

III. Localized Voronoi Polygons

In this section, we describe our first algorithm for identifying boundary nodes based on LVPs.

A. Definition and Properties of LVPs

To facilitate our illustration, we first define VPs and LVPs 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

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

Obviously, $Dom(s_i, s_j)$ is a half place bounded by the perpendicular bisector of s_i and s_j, which separates all points in the plane than 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,

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

In the same way, the *localized Voronoi polygon* (LVP) $LVor(s_i)$ and the *tentative localized Voronoi polygon* (TLVP)

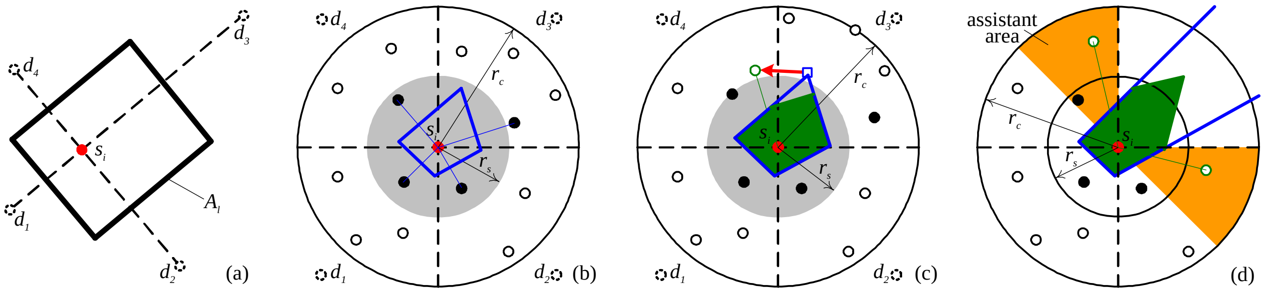

Fig. 2. Illustration of LVP-based boundary node detection algorithm.

$TLVor(s_i)$ associated with s_i are defined as:

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

$$TLVor\left(s_i\right) = \bigcap\nolimits_{s_j \in SubNeig(s_i)} Dom\left(s_i, s_j\right). \quad (9)$$

where $SubNeig\left(s_i\right) \subset Neig\left(s_i\right)$.
The collection of LVPs given by

$$\mathfrak{LVor}\left(V\right) = \left\{LVor\left(s_1\right), \cdots, LVor\left(s_n\right)\right\} \quad (10)$$

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

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

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

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

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

which directly leads to Lemma 1 (ii).

(iii) It is well known in computational geometry that

$$\bigcup\nolimits_{s_i \in V} Vor\left(s_i\right) = \mathbb{R}^2. \quad (11)$$

(cf. [17, Property V1, pp. 77] for a reference). Combined (11) with Lemma 1 (ii) that $Vor\left(s_i\right) \subseteq LVor\left(s_i\right)$, we can directly obtain Lemma 1 (iii). ■

Therefore, the set $\mathfrak{LVor}\left(V\right) \cap A$ can fully cover 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 $Clust\left(s_i\right)$ we have

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

[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.

B. 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, which is illustrated with node s_i as an example.

Input. Note that we need both directional and distance information (relative 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 is available. In the first case when ∂A_l is unavailable at node s_i, our detection scheme is based on the construction of $LVor\left(s_i\right)$ (or $TLVor\left(s_i\right)$). In the second case when ∂A_l is available, we need to exploit this information by calculating $LVor\left(s_i\right) \cap A_l$ (or $TLVor\left(s_i\right) \cap A_l$). It can be shown that $LVor\left(s_i\right) \cap A_l$ 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 between s_i and the dummy nodes generate the four edges of the border of ROI. Then we can calculate $LVor\left(s_i\right) \cap A_l$ by following the same procedure for calculating $LVor\left(s_i\right)$. Therefore, we will only address the first case in what follows.

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

$$TLVor\left(s_i\right) \leftarrow \bigcap\nolimits_{j=1}^{4} Dom(s_i, s_j).$$

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

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

The new vertices of the new TLVP will be checked to see whether they are covered by $Disk\left(s_i, r_s\right)$. This procedure

[4]Other values will also work well.

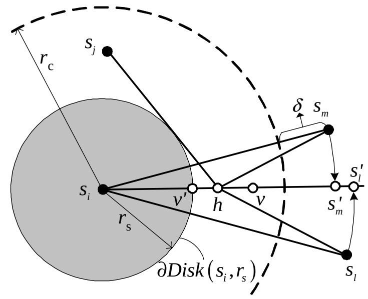

Fig. 3. Illustration of the proof of Theorem 1.

continues until the LVP is calculated and saved.

Note that when ∂A_l is unavailable, $LVor(s_i)$ may be infinite, which means it is possible that we cannot find any nodes in one or more quadrants. See Fig. 2 (d) for an example. If one quadrant contains no neighbors, we define two sectors with 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 assistant area cannot make TLVP finite, we can conclude that LVP must be infinite without further calculation.

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

C. 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.*

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 does any other node in $Disk\,(s_i, r_c)^{\complement}$. The reason is that, since $\|s_i s_l\| > \|s_i s_m\|$ holds for any node $s_l \in Disk\,(s_i, r_s)^{\complement}$ 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 nodes in $Disk\,(s_i, r_c)$ can cover h. Consequently, we can conclude that no node in the plane can cover h because $Disk\,(s_i, r_c) \cup Disk\,(s_i, r_c)^{\complement} = \mathbb{R}^2$. 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

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

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 correction 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 $u \in LVor\,(s_i)$. ■

D. Some Discussion 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 the def. 4), which is impossible for computing $Vor\,(s_i)$.

Assume the number of neighbors is k, each node can compute its own $LVor(s_i)$ with complexity smaller than $\mathrm{O}(k)$. The computing of the $LVor(s_i)$ only involves some simple operations on polygons which can be efficiently implemented (e.g., PolyBoolean library [14]). We further simplify the detection procedure by constructing TLVPs first. For a densely deployed WSN, we have $LVor(s_i)$ or $TLVor(s_i) \rightarrow Vor(s_i)$, and it is well known in computational geometry, under the Poisson point model, the average number of vertices of $Vor(s_i)$ is 6 [17]. Therefore, when the node density is high, only $4 \sim 6$ nearest neighbors's information will be needed for our LVP-based algorithm to detect the boundary nodes. As compared to perimeter-based approaches mentioned in Section II-D which require all information of k neighbors no matter how high the k will be, our LVP-based approach minimizes the needed information from the neighbors, and thus greatly reduces the computation and communication costs. Moreover, when a neighbor node dies, our LVP-based approach 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 detection. This unique property

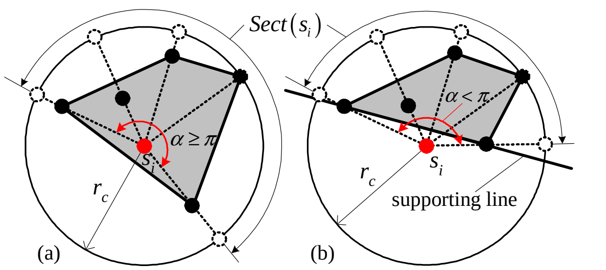

Fig. 4. Illustration of the convex hull of node s_i's neighbors (shaded area) and the smallest sector $Sect(s_i)$ containing all neighbors when (a) node s_i has the NEP and (b) node s_i does not have the NEP. Solid nodes represent neighbors of s_i and dotted open nodes are the projection of neighbors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

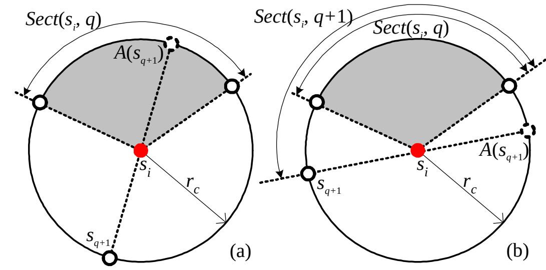

Fig. 5. Illustration of the computation of $Sect(s_i, q+1)$ from $Sect(s_i, q)$.

will simplify the updating of detection results and save precious energy of each sensor nodes.

IV. Neighbor Embracing Polygons

The *neighbor embracing polygon* (NEP) was first introduced in computational geometry as an alternative to the Voronoi polygon [5], [7]. 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.

A. 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 $CH\,(Neig\,(s_i))$, then 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 $Neig\,(s_i)$ [3].

The idea of using NEP in boundary node detection is quite intuitive. Fig. 4 illustrates the relationship between s_i and its $CH(Neig(s_i))$. We can see that s_i is more likely 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 [5], [7]. An NEP is constructed globally in [5], [7]: 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 V-A). More important, 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 [5, Lemma 2]. Note that, different from [5], the degenerate case in which there are only two vertices of $CH\,(Neig\,(s_i))$ defining a line segment containing s_i and $\alpha = \pi$, has been excluded here by the Definition 5. ∎

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

B. 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:

Input. The NEP-based algorithm does not require distances to s_i's neighbors, but only needs 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 decided by the two end points on $\partial Disk\,(s_i, r_c)$. We only consider the case when ∂A_l is unavailable in this section, since even this information is available, it still cannot be utilized.

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 average complexity of sorting is $\mathrm{O}\,(k \log k)$, where k is the number of neighbors. Below we describe how to decide whether $\alpha > \pi$ in $\mathrm{O}\,(k)$ 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

[5]NEPs are never used in coverage boundary detection in [5], [7].

[6]r_c can be any other value.

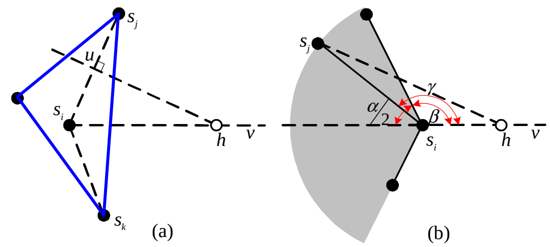

Fig. 6. Illustration of the proof of Theorem 4.

tentative sector one by one. Given $Sect(s_i, q)$ ($q < k$) and the next point s_{q+1} randomly choosed 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 by 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 the α is obtained.

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.

C. 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 $\gamma \geq \beta > \pi/2$. So $\overrightarrow{s_i v} \in LVor\,(s_i)$, 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.

[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.

D. Some Discussion on NEP-Based Detection

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. In doing so, although both algorithms need to be executed for some nodes, the overall energy consumption and response time may be reduced in contrast to the case when 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, the whole information about all the boundary nodes is necessary. In practice, however, some degrees' 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 consists of continuous closed curves) can be utilized for the users to recover some lost data about boundaries. Therefore, we can still get the main information about the coverage boundaries from partial boundary nodes detected by NEP-based algorithm. The simulation result supports our belief and quite positive: the positions of nodes without NEPs can depict the major topology shape of the connected coverage area (see Fig. 8 in Section V-A).

V. Performance Evaluation

In this section, we first validate the accuracy of our algorithms by simulations. We then show by theoretical analysis and simulations that our algorithms outperform the existing schemes in the literature.

A. 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. Fig. 7 shows the detection results for two WSNs with different coverage holes. The boundary nodes detected by the NEP-based algorithm are darkly shaded dots, and the additional ones detected by the LVP-based algorithm are lightly shaded dots. In addition, the theoretical coverage boundary is formed by solid lines. The effectiveness of our algorithms are quite obvious, the LVP-based algorithm can detect almost 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 the right side of

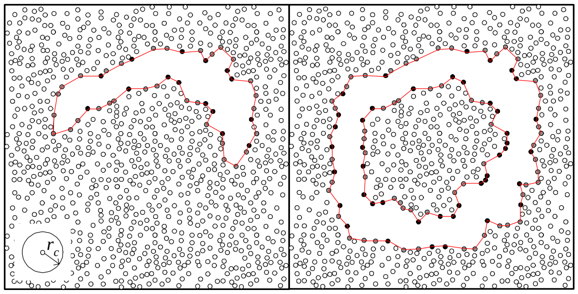

Fig. 7. Boundary detection results for large-scale WSNs.

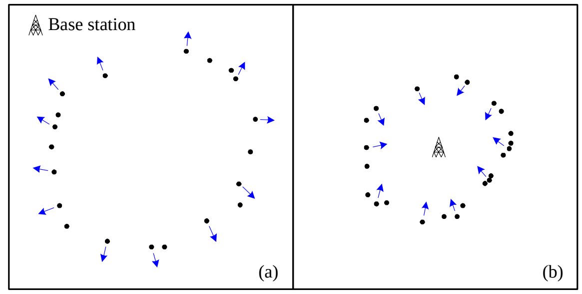

Fig. 8. Coverage boundary reconstruction results on the base station with the NEP-based algorithm.

Fig. 7 as an example. All the boundary nodes determined by the NEP-based algorithm send their positions or IDs to the BS, which, in turn, can reconstruct the coverage boundary based on such information. Fig. 8(a) and Fig. 8(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 some details of boundaries are lost, the outlines are still obtained.

B. Cost Analysis

1) LVP-based approach vs. perimeter-based approach: All these two approaches can provide truly localized solutions to the CCBD problems. The difference is that for densely deployed WSNs, the number of neighbors' information needed for our scheme is a constant, while for perimeter-based approach, all the neighbors' information is required (cf. Section III-D). Therefore, the higher the node density, the greater the benefit using our scheme.

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** 5: 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 $Vor\left(s_i\right)$ can be locally computed if and only if $Clust\left(s_i\right)$ can completely cover the plane $\mathbb{R}^2$ (or A_l, when the information of ∂A_l is available), i.e., $Cover\left(s_i\right) = \mathbb{R}^2$ (or $Cover\left(s_i\right) \cap A_l = A_l$).

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, then $d = \infty$ 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 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. So Theorem 5 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.

C. Evaluation of Energy Consumption for VP- and LVP-Based Approaches

1) Evaluation settings: To facilitate the theoretical analysis, we assume that sensor nodes are distributed in a large square region A_l and form a homogeneous Poisson point process with density λ. Each node knows its own position by GPS or existing localization schemes such as [15]. For any measurable subset of A_l with area B,

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

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 $\lambda' = (1-p)\lambda$ [20]. Therefore, the network can be uniquely identified by the current node density λ (or equivalently k).

Here we only consider the homogeneous Poisson point process because it is widely accepted in modeling sensor networks [10]. In addition, we let the side length $l \to \infty$ (which implies $n = \lambda A_l \to \infty$). In 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" [18].

Based on the continuum percolation theory [16], [18], if $k \leq 4.5$, a 2-D network will be partitioned into $\mathrm{O}\left(n\right)$ small clusters, which implies that the WSN will completely failed. Previous work [10], [13] also points out that for $n \to \infty$, the ROI A_l is completely covered with a high probability when λ and k satisfy

$$\pi r_s^2 \lambda = \pi (r_c/2)^2 \lambda = k/4 = \log(n) + 2 \log\log(n) \,. \quad (14)$$

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

$$4.5 < k < 4 \log(n) + 8 \log\log(n) \,. \quad (15)$$

In our evaluation, in order to have a fair comparison, VPs are computed in a similiar way as LVPs with set V as 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 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. Below shows the procedure for computing VPs/LVPs/NEPs.

Step 1: Every node s_i broadcasts its position s_i, and receives its neighbors' position message. 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

$$(m-1)\,(E_T + kE_R)\left(S_{d(s_j)} + S_{s_j}\right). \quad (16)$$

2) Theoretical results: Given the density λ, from the proof of Lemma 3, $2d(s_i)$ can be computed from (14) as:

$$2d\,(s_i) = \left(\frac{4\log(n) + 8\log\log(n)}{\pi\lambda}\right)^{1/2}. \quad (17)$$

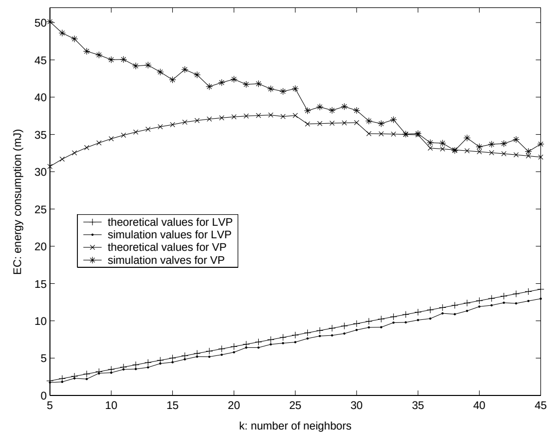

Fig. 9. Energy consumption for the LVP- and VP- based algorithms.

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 [6]:

$$\mathbf{E}\,(m) = \begin{cases} 1, d \le r_c \\ 0.5 + h \cdot c, d > r_c \end{cases} \quad (18)$$

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:

$$\mathbf{E}\,(m) = \frac{1}{2} + \frac{c}{r_c}\left(\frac{4\log(n) + 8\log\log(n)}{\pi\lambda}\right)^{1/2}. \quad (19)$$

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

$$EC_{LVP} = (E_T + kE_R)\,S_{s_i}. \quad (20)$$

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

$$EC_{L\_Hop} = (E_T + E_R)(\pi(2d(s_i))^2\lambda - \pi r_c^2 \lambda) S_{s_i}. \quad (21)$$

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

$$\begin{aligned} EC_{VP} = {} & EC_{LVP} + \mathbf{E}(m)(E_T + kE_R) S_{d(s_i)} \\ & + (\mathbf{E}(m) - 1)(E_T + kE_R) S_{s_i} + EC_{L\_Hop}. \end{aligned} \quad (22)$$

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

$$\frac{EC_{VP}}{EC_{LVP}} > 2.75. \quad (23)$$

Obviously, the energy savings are significant. Note that (23) 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 (23) still holds.

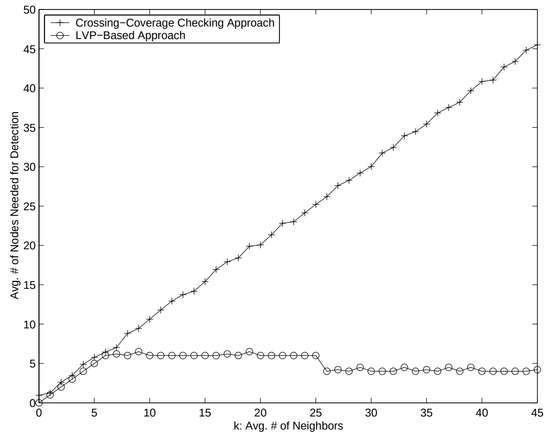

Fig. 10. Average number of neighbor nodes needed for the crossing-coverage checking approach and LVP-based approach.

D. Simulation results

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, $E_R = 0.6$ μJ/bits, $E_T = 0.8\mu\mathrm{J/bit}$, and $4.5 < k < 45$ (the range of k derives from the Eq.15). Fig. 9 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 (21). We can also observe that the difference becomes smaller with the increase of k. This is because the number of hops needed to reach $2d(s_i)$ will become smaller with the increase of k and our approximation will make more sense. In general, it is obvious that our LVP-based algorithms can achieve remarkable energy savings.

2) LVP-based approach vs. perimeter-based approach: The simulate settings are the same as the above. Fig. 10 shows the average number of neighbor nodes needed for the perimeter-based approach and LVP-based approach to detect boundary nodes as a function of k. We can see that, as we predict, when the node density increases, the number of nodes needed holds as a constant for LVP-based detection while increases synchronously for perimeter-based approaches. Therefore, compared to our scheme, perimeter-based approaches will be a burden especially at the beginning of the lifecycle of WSNs where WSNs are usually densely deployed to provide greater redundancy and fault-tolerance.

VI. 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.

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):169–181, 393-422 2002.

[3] M. Berg. *Computational Geometry: Algorithms and Applications*. Springer, New York, 2000.

[4] 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.

[5] 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.

[6] S. Chandler. Calculation of number of relay hops required in randomly located radio network. *Electronics Letters*, 25(24):1669–1671, November 1989.

[7] 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):47–55, March 2003.

[8] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *INFOCOM'04*, Hong Kong, China, March 2004.

[9] A. Ghosh. Estimating coverage holes and enhancing coverage in mixed sensor networks. In *LCN'04*, Washington, DC, Nov. 2004.

[10] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[11] P. Hall. *Introduction to the Theory of Coverage Processes*. John Wiley Sons Inc., New York, 1988.

[12] 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.

[13] S. Kumar, T. H. Lai, and J. Balogh. On k-coverage in a mostly sleeping sensor network. In *MobiCom'04*, Philadelphia, PA, Oct. 2004.

[14] M. Leonov. Polyboolean library (2004). http://www.complex-a5.ru/polyboolean/.

[15] 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):147–159, Feb. 2005.

[16] R. Meester and R. Roy. *Continuum Percolation*. Cambridge University Press, Cambridge, 1996.

[17] A. Okabe. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York, 2000.

[18] M. Penrose. *Random Geometric Graphs*. Oxford University Press, Oxford, 2003.

[19] 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, Nov. 1999.

[20] D. Stoyan, W. Kendall, and J. Mecke. *Stochastic Geometry and its Applications*. Wiley, New York, 2nd edition, 1995.

[21] G. Wang, G. Cao, and T. L. Porta. Movement-assisted sensor deployment. In *INFOCOM'04*, Hong Kong, China, March 2004.

[22] 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.

[23] H. Zhang and J. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *Wireless Ad Hoc and Sensor Network*, 1(1-2):89–123, January 2005.