2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems

An Energy-Efficient and Distributed Cooperation Mechanism For k-Coverage Hole Detection And Healing in WSNs

Chenxi Qiu, Haiying Shen and Kang Chen
Department of Electrical and Computer Engineering
Clemson University
Clemson, USA
{*chenxiq, shenh, kangc*}*@clemson.edu*

***Abstract*—Present approaches to achieve k-coverage for Wireless Sensor Networks still rely on centralized techniques. In this paper, we devise a distributed method for this problem, namely Distributed VOronoi based Cooperation scheme (DVOC), where nodes cooperate in hole detection and recovery. In previous Voronoi based schemes, each node only monitors its own critical points. Such methods are inefficient for k-coverage because the critical points are far away from their generating nodes in k-order Voronoi diagram, causing high cost for transmission and computing. As a solution, DVOC enables nodes to monitor others' critical points around themselves by building local Voronoi diagrams (LVDs). Further, DVOC constrains the movement of every node to avoid generating new holes. If a node cannot reach its destination due to the constraint, its hole healing responsibility will fall to other cooperating nodes. The experimental results from the real world testbed demonstrate that DVOC outperforms the previous schemes.**

***Keywords*-Wireless Sensor Networks; k-coverage; Voronoi diagram; hole detection; hole healing;**

I. Introduction

Among numerous challenges confronted in designing protocols for WSNs, the coverage problem stands out as one of the most critical issues. Some WSN applications such as environment/ocean monitoring and animal tracking require to cover each point of the target region. However, even if we initially can deploy sensor nodes to make the entire target region fully covered, the nodes may die due to battery drain or environmental causes, which may generate coverage holes in the region. Also, nodes may deviate from their initially assigned positions due to uncontrollable factors (e.g., motion of ocean waves), leaving some areas uncovered [1]. Coverage holes reduce the ability of WSNs to detect events and network reliability. Therefore, it is crucial to equip sensor nodes with efficient hole detection and recovery capabilities to ensure full coverage of the target region.

A target field is termed k-covered ($k \geq 1$) if every point in the target field is in the sensing ranges of at least k nodes. A k-coverage hole is a continuous area in the target field comprised of points that are covered by at most $k-1$ sensors. The problem of k-coverage is motivated by robustness concerns as well as protocol requirements. Previous schemes [2], [3] for k-coverage hole detection generate a high time complexity in a large-scale WSN with a large number of nodes. Also, their centralized method makes them not feasible in large-scale WSNs because it burdens the central node while sensor nodes have limited energy and computation capacity, thus easily generating bottlenecks. Besides hole detection, hole recovery is a key issue in the coverage problem. Numerous hole recovery schemes [2], [4]–[8] use sensor movement to improve network coverage. In these schemes, sensor nodes equipped with mobile platforms move around after the initial deployment. However, most of these schemes use centralized methods, which are not feasible in large-scale WSNs due to the aforementioned reasons. Also, they assume that accurate location information of each node is known, which are impractical for WSNs in some cases [9].

Some previous schemes [10], [11] use distributed approaches to build 1-order Voronoi diagrams (VDs) for 1-coverage detection. A VD is composed of numerous Voronoi cells, each of which has one sensor called *generating node* residing in it. All points within a Voronoi cell are closer to their generating node in the cell than to those in other cells. Thus, a Voronoi cell is fully covered if a generating node covers all of its Voronoi cell's vertices. However, it would be very energy-expensive to directly extend these schemes for the k-order coverage problem because each sensor node requires much more location information for building k-order VD and must monitor its distant critical points.

Our work aims to solve two formidable challenges: 1) How can k-coverage holes be efficiently detected in a distributed manner? And 2) how can new holes be avoided while healing current holes? Accordingly, we propose a Distributed VOronoi based Cooperation scheme (DVOC) based on mathematical models for k-coverage in WSNs. In DVOC, nodes cooperate in hole detection and recovery by node movement, which significantly saves energy by reducing message trans-

978-1-4673-9101-6/15 $31.00 © 2015 IEEE
DOI 10.1109/MASS.2015.115

mission and avoiding generating new holes during node movement.

We first introduce a definition of k^{th} generating (n-earest) node of a critical point (*i.e.*, Voronoi vertex), so that as long as the k^{th} generating node covers its critical point, this point is k-covered. We also introduce a warm up method to find a point's k^{th} generating node from a set of nodes. Thus, simply enabling Voronoi vertex to be monitored by its k^{th} nearest node can achieve k-coverage hole detection. However, their long monitoring distance would lead to high transmission cost. To solve this problem, in DVOC, each node builds *local k-order VD (LVD)* that enables it to find its nearby critical point's generating nodes and check if the generating nodes cover this critical points. Thus, nodes cooperate in monitoring each other's critical points and informing the generating nodes of uncovered critical points. We have proven that DVOC never misses any holes if the accuracy of the LVD is guaranteed, and it alleviates the transmission burden for each node significantly. "Accuracy" here means the degree that the constructed VD approximates the actual VD. Further, in order to avoid generating new holes during node movement, we mathematically identify the safe area of each node, where the node should be located to avoid new hole generated. If a node cannot reach its destination due to the constraint, its hole healing responsibility falls to another k^{th} generating node. Compared with previous schemes, DVOC costs less for both transmission and mechanical movement; since DVOC can avoid oscillating movement with its cooperation mechanism, it converges more rapidly than previous movement schemes. Specially, DVOC consists of three components.

1) *Distributed k-coverage checking*: each node first collects location information from its neighbor nodes, and builds a LVD, which allows it to find the critical points of others around itself within the diagram. Then, nodes cooperate in monitoring nearby critical points for each other. When a node detects a hole, it informs the generating nodes to move towards the hole.

2) *Safe area identification*: By transforming the k-coverage problem to the problem that whether the radius of circumcircle of each k-order Delaunay triangle (DT) [12] formed by each critical point's three generating nodes is smaller than the radius of nodes' sensing ranges, we calculates each node's safe area.

3) *Movement-based hole healing strategy*: When a sensor is informed to move a point, it calculates its *safe area* using the information of its critical points retrieved from nearby nodes. Once the sensor finds itself unable to reach the destination due to the constraint of its safe area, its hole healing responsibility falls to the point's another generating node (sensor).

The rest of this paper is organized as follows. Section II presents related work. Section III introduces the models for the coverage problem and presents the movement-assisted scheme for detecting and healing coverage holes. Section IV presents a performance evaluation of DVOC in comparison with several previous schemes. The final section concludes with a summary of contributions and discussions on further research work.

II. Related Work

Over the past years, intensive research efforts have been devoted to the study of the coverage problem (including *hole detection* and *hole healing*) in WSNs and VD has been served as a very useful tool to solve this problem [2], [4], [5]. In VD based schemes, each node checks its critical points (*i.e.,* vertices of the Voronoi cell). Once a sensor finds holes at its critical points, it moves to heal the holes (*e.g.*, in [4], a node moves to the furthest vertex of its Voronoi cell). A challenge for VD based algorithm is how to build VD distributively and efficiently. Some previous works have introduced distributed algorithms for constructing 1-order VD, which only exploits locality information, rather than broadcasting location information to all nodes in the WSN. For example, Sharifzadeh and Shahabi [10] proposed a method in which a node uses its collected location information of some nodes to build a 1-order VD. Bash and Desnoyers [11] proposed a method to improve the accuracy. The method begins with an initial approximation of a local k-order Voronoi cell at each node based on its neighboring nodes and then leverages geographic routing primitives (*e.g.*, GPSR [13]) to systematically refine the Voronoi cell and verify its correctness. To judge whether its constructed Voronoi cell is accurate, a node only needs to check whether there is a node unknown by itself that is closer to any of the cell's vertices than itself. However, none of previous works propose an algorithm for building a k-order VD in a distributed manner, which requires each node to hold much more location information of other nodes and generates higher energy costs for transmission and computing.

Besides VD based schemes, many other methods have also been proposed for coverage detection and recovery. For example, "Virtual force", as a movement strategy for healing coverage holes, have been used for hole recovery [6]–[8] to adjust the distance between any two nodes, *i.e.*, when the distance between two sensors is too long, the attractive force makes them "pull" each other closer; and when the distance is too short, the repulsive force makes them "push" each other further. Consequently, sensor nodes are exploded from dense regions to sparse regions or holes. However, these methods require sensors to move over a series

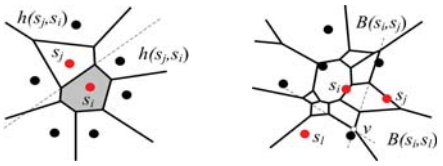
of iterations to balance "virtual forces" between themselves, which may take a long time to converge and is not practical for real applications due to the high energy costs of moving. Luo *et al.* [1] assumed that the entire target region is fully covered initially, and each node dominates a number of interest points which are randomly and uniformly distributed throughout the entire region. When some nodes lose their interest points, their neighbors move to inherit them. Though the above movement-assisted schemes have their own merits., most of these schemes only focus on the 1-coverage case, or require nodes to have a knowledge of location information of all other nodes.

III. The Design of DVOC

A. Network scenario

Consider a WSN comprised of N mobile nodes, denoted by $S = \{s_1, s_2, ..., s_N\}$, that are uniformly distributed over a *target field* Λ to detect specified events. The location of each s_i can be represented by (x_i, y_i). We use the Euclidean metric to measure the distance, use $\mathrm{dist}\,(s_i, s_j)$ or d_{ij} to denote the distance between any two nodes s_i and s_j, and use $B\,(s_i, s_j)$ to denote the perpendicular bisector of s_i and s_j. Each node s_i can sense specified events in its *sensing range* modeled as a disk D_i with radius R_s. We assume that each sensor knows its location (*e.g.*, by GPS), and the sensor nodes are dense enough to enable the entire network to be well connected. We also assume that sensors can freely move in any direction in the target region, and there is no obstruction area where sensors cannot move in.

Variable	Description
$\mathcal{S}$	The set of sensor nodes
S	The set of sensor nodes' locations
$\mathcal{F}$	The target region
s_i	Sensor node i
p_i	The location of s_i
d_{ij}	The Euclidean distance between p_i and p_j, which can also be denoted by $\mathrm{dist}\,(p_i, p_j)$
$\mathcal{V}_k(p_i)$	The 1-order Voronoi cell of p_i
$V_k(\mathcal{S})$	The k-order VD of $\mathcal{S}$, where $\mathcal{S}$ is a set of nodes
$\hat{V}_k^i\,(\mathcal{N}_i)$	The local k order VD of s_i, where $\mathcal{N}_i$ is a set of nodes stored in s_i
$B\,(p_i, p_j)$	Perpendicular bisector of p_i and p_j
$h\,(p_i, p_j)$	Use $B\,(p_i, p_j)$ to divide $\mathcal{F}$ into two half planes, and $h\,(p_i, p_j)$ denotes the half plane that contains p_i
$\mathcal{G}_v$	The set of k^{th} generating nodes of a Voronoi vertex v.
$\mathcal{N}_i$	The node set stored in s_i's storage



(a) 1-order VD (b) 2-order VD

Figure 1. 1-order and 2-order VDs.

Definition 3.1: A region Λ is k-covered by all nodes S if

$$\forall p \in \Lambda,\ \exists \mathcal{S} \subseteq S,\ \text{where } |\mathcal{S}| = k$$
$$s.t. \qquad p \in \bigcap_{s_i \in \mathcal{S}} D_i. \tag{1}$$

In other words, Λ is k-covered by S when every point in Λ is covered by the sensing ranges of at least k sensors. If there exists a point that is covered less than k times, then this point is called a *coverage hole*.

B. Preliminary of VD

The VD of a collection of nodes partitions the plane into *Voronoi cells* (or *cells* for simplicity) with one node inside each cell. The node located in a cell is called the cell's *generating node* and the cell is called this generating node's cell. Every point in a cell is closer to the cell's generating node than to any other nodes [4], [14]. Every vertex is called the *critical point* of the generating node of the cell. Fig. 1 (a) shows an example of a 1-order VD consisting of 8 cells.

Given any pair of nodes $s_i, s_j \in S$, we use their perpendicular bisector (dotted line in the figure) to divide the plane into two half-planes: $h\,(s_i, s_j)$ containing s_i and $h\,(s_j, s_i)$ containing s_j. Then, the 1-order Voronoi cell of s_i, denoted by $\mathcal{V}_1\,(s_i)$, is the intersection of $N-1$ half-planes with all other nodes (the grey part in Fig. 1 (a)). It can be represented by:

$$\mathcal{V}_1\,(s_i) \triangleq \bigcap_{1 \le j \le N,\ j \ne i} h\,(s_i, s_j)\,. \tag{2}$$

The 1-order VD consists of the 1-order Voronoi cell of each node in the network. Notice that we use $\mathcal{V}$ and V to denote Voronoi cell and VD respectively.

An extension form of the 1-order VD is a k-order VD [15]–[17], in which each cell $\mathcal{V}_k\,(\mathcal{S})$ is associated with a subset of points, denoted by $\mathcal{S}$ ($\mathcal{S} \subset S$, $|\mathcal{S}| = k$). $\mathcal{V}_k\,(\mathcal{S})$ can be calculated by

$$\mathcal{V}_k\,(\mathcal{S}) \triangleq \bigcap_{s_i \in \mathcal{S},\ s_j \in S \setminus \mathcal{S}} h\,(s_i, s_j)\,. \tag{3}$$

$\mathcal{V}_k\,(\mathcal{S})$ is the locus of points; each node's maximum distance to all points in $\mathcal{S}$ is shorter than its distance to any other points not in $\mathcal{S}$. Fig. 1 (b) shows the 2-order VD for the 8 points.

Property 3.1: To detect whether an entire target region is k-covered, it is sufficient to check whether the vertices of each $\mathcal{V}_k(\mathcal{S})$ are k-covered by k sensors [2].

C. Distributed k-coverage checking

We order the distances between a vertex and the k sensors in $\mathcal{S}$ from the 1^{st} nearest to the k^{th} nearest (*i.e.*, from the shortest distance to the longer distance). Based on Property 3.1, if the vertex can be covered by the k^{th} nearest sensor, then it definitely can be covered by other $k-1$ sensor in S. Thus, we define the *critical points* and k^{th} *generating nodes* as below, and extend Property 3.1 to Property 3.2.

Definition 3.2: For a k-order Voronoi cell $\mathcal{V}_k$, the *critical points* of $s_i \in \mathcal{S}$ are the cell vertices whose k^{th} nearest node is s_i; s_i is called the k^{th} *generating node* of its critical points.

In a k-order VD, a vertex has at least three k^{th} generating nodes, and each k^{th} generating node has a set of critical points.

Property 3.2: To detect whether an entire target region is k-covered, it is sufficient to check whether each vertex of each $\mathcal{V}_k(\mathcal{S})$ is covered by its k^{th} generating node [2].

Theorem 3.1: Given a k-order Voronoi cell $\mathcal{V}_k(\mathcal{S})$, if v is a point on a bisector edge $B(s_i, s_j)$ of $\mathcal{V}_k(\mathcal{S})$, then the k^{th} nearest nodes of v are s_i and s_j [2].

Corollary 3.1: Given a k-order Voronoi cell $\mathcal{V}_k(\mathcal{S})$ if v is one of its vertices which is intersected by $B(s_i, s_j)$ and $B(s_i, s_l)$, then the k^{th} generating nodes of v are s_i, s_j and s_l.

Fig. 1 (b) gives an example for Corollary 3.1. In the figure, $B(s_i, s_j)$ intersects with $B(s_i, s_l)$ at point v, which implies that s_i, s_j and s_l are the k^{th} generating nodes of v. Based on Corollary 3.1, using a k-order VD, the k^{th} generating nodes of each vertex of a cell can be easily found.

D. A warm up method: PVD based k-coverage checking

DVOC conducts k-order VD construction and k-coverage checking in a distributed manner. A question is how to distribute these workload among sensors. Before introducing LVD, we first introduce a warm up method called *PVD-based k-coverage checking*, which can be simply extended from the distributed method for constructing 1-order VD [11]. Notice that in a k-order VD, a node s_i may be associated with several cells. We call the combination of these associated cells s_i's *partial k-order VD (PVD)* (Definition 3.3), and it requires each node to be responsible for its partial k-order VD in k-coverage checking.

Definition 3.3: Suppose $\mathcal{S}_i^1$, ..., $\mathcal{S}_i^{m_k}$ ($m_k = \binom{N-1}{k}$) are the subsets of S with cardinality k that contain node s_i. The combination of the cells associated with these subsets is called the partial k-order VD (PVD) of s_i.

In the previous distributed 1-order VD construction methods [10], [11], each node initializes its tentative cell using a small subset of nodes in its region. The node is the *generating node* of the vertices of its cell. The GPSR routing algorithm [13] always routes a packet to the reachable node with minimum distance. To increase the cell's accuracy, the generating node s_i checks if there exits a node that is closer to each of the cell's vertices than itself using GPSR; if yes, s_i reduces the area of the tentative cell by excluding the closer node from the cell. When each node stops cell modification, the 1-order VD is constructed. Similarly, to construct PVD, each node s_i first initializes its tentative PVD using the known nodes (including itself) in its storage. For each vertex of its PVD v, s_i first locates v's k^{th} generating nodes and then probes nearby nodes using GPSR to check if there is a node is closer to v than the v's current k^{th} generating node. It then adds such nodes into its storage. After the checking of all vertex, s_i rebuilds its PVD to enhance the accuracy of the PVD. Since a node's PVD only consists of its nearby nodes, each node only needs to probe its nearby nodes for the PVD construction [10], [11]. Node s_i stores the vertices and edges for its tentative PVD. This process repeats until s_i cannot find closer node to any of its PVD's vertices. After all nodes build their PVD, the k-order VD is constructed. Fig. 2 shows an example for the 2-order PVD construction. In Fig. 2 (a), s_1 modifies its PVD to 2 (b) when a new node s_5 is closer to the PVD's vertex v_3 than s_2 and s_3 (s_2 and s_3 are the 2^{nd} generating nodes of v_3). Similarly, in 2 (b), s_1 continues to modify its PVD to 2 (c) because a new node s_6 is closer to the PVD's vertex v_6 than s_1 and s_4 (s_1 and s_4 are the 2^{nd} generating nodes of v_6). While in Fig. 2 (d), s_{10} will not be observed by s_1 through GPSR as s_{10} is not closer to any vertex in the diagram than vertex's 2^{nd} generating nodes.

E. LVD based k-coverage checking

Accordingly, rather than relying on one central node to collect the location information of all nodes in the WSN and then build the k-order VD, PVD based k-coverage checking distributes this workload among the nodes by letting each node collect partial location information and build its own k-order PVD in order to build the k-order VD. Fig. 3 shows an example of a 16-node WSN, where s_1 has built its partial 2-order VD with 7 critical points so far. We draw circles with the critical points be the circle centers, and their distances to s_1 be the radius. Then, we draw the smallest circle centered at s_1 that embraces all these circles (denoted

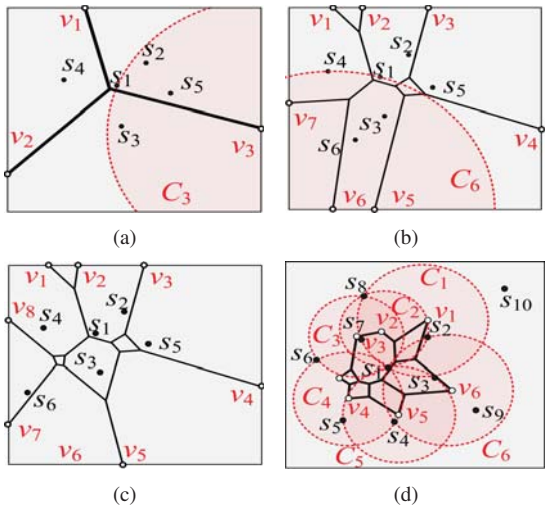


Figure 2. Construction of 2-order VD.

by C_1). To guarantee the accuracy of its partial diagram, s_1 only needs to use GPSR to collect the location information of some nodes inside C_1, because nodes outside of C_1 are farther to the critical points than s_1 and these nodes do not affect the PVD construction. However, higher value of k leads to larger C_1 and more location information that s_1 needs to collect by GPSR, leading to a high energy cost.

Also, we observe that nodes' partial k-order VDs always overlap with each other. In other words, each critical point is known by several nodes rather than only one node. Thus, multiple nodes checking the same critical point leads to duplicated checking and unnecessary energy cost. Therefore, simply extending the distributed 1-order VD construction method to a k-order VD construction method still leads to a high energy cost, especially when k is large. It is crucial to find an energy-efficient method to reduce the probing cost for high accuracy. To this end, we propose a k-coverage checking method based on *local k-order VD (LVD)*.

LVD designates the node s_i closest to a critical point of another node s_j to conduct the accuracy checking and k-coverage checking on this point. Thus, LVD shrinks the circle where s_i needs to collect location information, hence reducing GPSR probing distances and the number of probed nodes of each node. It also avoids unnecessary probing cost due to the overlapping.

A challenge in LVD is how to distribute the set of all critical points in the target field to sensors so that each critical point is only assigned to one sensor that is closest to itself. We notice that a point s_i's 1-order Voronoi cell is mutually exclusive to any 1-order Voronoi cell of other node's location. We then define LVD as the intersection of s_i's 1-order Voronoi cell and the k-order VD. Then, the local k-order VD of each node must be mutually exclusive. Also, the combination of LVD form

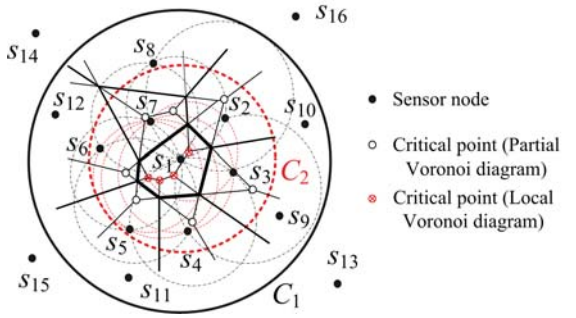


Figure 3. LVD vs. PVD.

the intact k-order VD, which will be proved later.

Definition 3.4: Given a set of nodes $\mathcal{N}_i$ known by s_i, the local k-order VD of a node s_i (denoted by $\hat{V}_k^i(\mathcal{N}_i)$) is defined as the intersection of s_i's 1-order Voronoi cell and the k-order VD of $\mathcal{N}_i$, or formally

$$\hat{V}_k^i(\mathcal{N}_i) \triangleq V_k(\mathcal{N}_i) \cap \mathcal{V}_1(s_i). \quad (4)$$

We say a LVD $\hat{V}_k^i(\mathcal{N}_i)$ is *accurate* if $\hat{V}_k^i(\mathcal{N}_i) = \hat{V}_k^i(S)$.

Lemma 3.1: To guarantee s_i's LVD's accuracy, there cannot be any node unknown to s_i that is closer to any of the cell's vertex than any of the cell's generating nodes.

Proof: Lemma 3.1 can be easily derived from [11]. ■

Below, we present how a node builds its LVD (Algorithm 1). Similar to PVD construction, each node also conducts the accuracy check on its identified critical points of other nodes. Basically, it checks whether there is a node within the circle with the critical point as the center and the distance between the critical point and its generating node as the radius. If yes, it additionally considers the location of the closer node to build a more accurate local k-order VD. We use $\mathcal{N}_i$ to denote the set of location points that node s_i has collected. Each node s_i calculates its LVD by Equ. (4).

For example, in Fig. 3, s_1 builds its local k-order VD (marked by bold lines in the center). It finds the 6 critical points within the diagram, whose generating nodes are $s_2 - s_7$. Then, s_1 only needs to probe nodes within circle C_2, which is the smallest circle that embraces all the circles; each having a critical point as the center and its distance with its generating node as the radius. When s_1 finds that a critical point cannot be reached by its k^{th} generating nodes' sensing ranges, s_1 informs the generating nodes. Compared to the PVD-based scheme, the LVD-based scheme reduces the number of critical nodes that should be checked by s_1, and reduces the probing scope from circle C_1 to C_2. In the following, we proves the correctness of our LVD-based scheme in Lemma 3.2.

Algorithm 1: Pseudo code for LVD.

input : $\mathcal{N}_i = \{s_i\}$ // At beginning s_i only has its own location;
output: $\hat{V}_k^i$;
repeat
  accuracy $\leftarrow$ TRUE; // Denote whether the LVD is accurate
  Calculate $\hat{V}_k^i$ based on the current $\mathcal{N}_i$ (Equ. (4));
  for *each* $v \in \hat{V}_k^i$ **do**
    Find a k^{th} generating node of v, say s_j;
    Use GPSR to probe the nearest node to v in $S/\mathcal{N}_i$, say s_l;
    if s_l *is closer to* v *than* s_j **then**
      Add s_l to $\mathcal{N}_i$;
      accuracy = FALSE;
until accuracy = TRUE;

Lemma 3.2: The union of all nodes' LVDs forms the VD and no critical points will be missed if the accuracy of every LVD is guaranteed.

Proof: Assuming the accuracy of every local k-order VD is guaranteed, by definition we can retrieve:

$$\begin{aligned} & \cup_{i=1}^{N} \hat{V}_k^i\left(\mathcal{N}_i\right) = \cup_{i=1}^{N} \hat{V}_k^i\left(S\right) \\ = \; & \cup_{i=1}^{N}\left(V_k\left(S\right) \cap \mathcal{V}_1\left(s_i\right)\right) \\ = \; & V_k\left(S\right) \cap \left(\cup_{i=1}^{N} \mathcal{V}_1\left(s_i\right)\right) = V_k\left(S\right) \cap \Lambda = V_k\left(S\right) \end{aligned}$$

■

We then compare the running time of building PVD and LVD. Here we use the k-order VD construction algorithm in [16] with time complexity $O\left(k^2\left(N\log N\right)\right)$, where N denotes the number of nodes in S. Due to limitations of space, we do not introduce the details of this algorithm. In this algorithm, whenever a new node location is added to $\mathcal{N}_i$, s_i needs to execute the algorithm one more time for accuracy checking. We denote c as the number of nodes' locations that s_i needs to collect to build a PVD. After the c^{th} node's location has been probed, the algorithm for building PVD is finished. The running time equals $\sum_{u=1}^{c} O\left(k^2 u\log u\right) = O\left(k^2 c^2 \log c\right)$. Similarly, the running time of LVD can be calculated as $O\left(k^2 e^2 \log e\right)$, where e denotes the number of nodes' locations that s_i needs to collect for the accuracy checking of the diagram. Since e is always less than c, the running time of k-order LVD construction scheme is less than that of the k-order PVD construction scheme. Thus, LVD is more computation- and energy-efficient, and hence is more practical than the previous schemes.

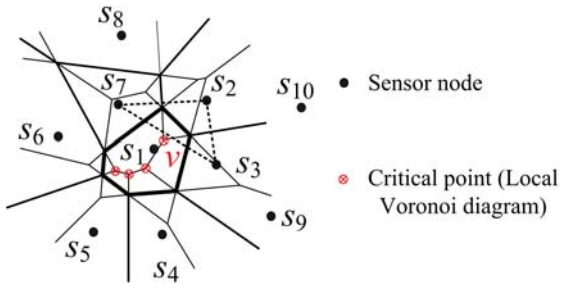


Figure 4. 1-order DT in 2-order VD

F. Safe Area Identification

In DVOC, after nodes build their own k-order LVDs, they collaborate to detect whether the vertices of the cells in the diagram (*i.e.*, critical points) are k-covered by finding out if each vertex is covered by its k^{th} generating node. In previous works such as VOR [4], a sensor node simply moves towards the Voronoi vertex that is not covered. However, such node movements might cause some areas to become uncovered, and several iterations might be needed to converge when a new hole cannot be covered by some other nodes. Therefore, we introduce a method to enable a node to identify its safe area that it should not move out. If the node moves out of the safe area, it no longer covers its critical point.

Definition 3.5: (*k-order Delaunay triangle (DT)* [12]) Let S be a set of sensor nodes locations in the plane. For s_i, s_j, $s_l \in S$, a triangle $\triangle s_i s_j s_l$ is a k-order DT if the circle through s_i, s_j, and s_l has at exactly k nodes of S inside.

Based on Corollary 3.1, in a VD, the three generating nodes s_i, s_j and s_l of a critical point v are the k^{th} nearest nodes to v (*i.e.*, the center of the circle of $\triangle s_i s_j s_l$). Therefore, in the circle of $\triangle s_i s_j s_l$, there must exist $(k-1)$ nodes nearer to the center of circle than the triangle's three vertices, which means that the circle must contains exactly $(k-1)$ nodes. Therefore, by connecting the three k^{th} generating nodes of any critical point in a node's LVD, we can always get a $(k-1)$-order DT.

If the k^{th} generating nodes' sensing ranges covers their critical point, then the radius of their $(k-1)$-order Delaunay triangle is smaller than their sensing range. Consequently, the problem that whether every point of target region is k-covered can be transformed to the problem that whether the radius of each of such $(k-1)$-order DTs is smaller than the sensing ranges of the k^{th} generating nodes. In Fig. 4, s_1 builds its

Algorithm 2: Hole detection conducted by s_i.

input : $\hat{V}_k^i$
output: $[\mathcal{C}_{\text{hole}},\ \mathcal{G}_{\text{hole}}]$
for *each critical point* v *in* $\hat{V}_k^i$ **do**
  Find v's k^{th} generating nodes $\mathcal{G}_v$;
  for *each node* $s_j \in \mathcal{G}$ **do**
    if s_j *cannot cover* v **then**
      Add v to $\mathcal{C}_{\text{hole}}$;
      Add s_j to $\mathcal{G}_{\text{hole}}$;

2-order LVD, which contains v and its 2^{nd} generating nodes are s_2, s_3 and s_7. The $\triangle s_2 s_3 s_7$ is a 1-order DT.

In order to make the three k^{th} generating nodes (*i.e.*, three vertices of a triangle) to cover their critical point (*i.e.*, the center of the circle), from the law of sine and cosine, we can derive that the relationship among the radius of triangle's circumscribed circle (denoted as r) and triangle's three edges (denoted as d_{ij}, d_{il} and d_{jl}). That is,

$$r = \frac{d_{ij} d_{il} d_{jl}}{\sqrt{\left(d_{ij}^2 + d_{jl}^2 + d_{il}^2\right)^2 - 2\left(d_{ij}^4 + d_{jl}^4 + d_{il}^4\right)}} \quad (5)$$

According to Equ. (5), we can retrieve that, any vertex of a k-order Voronoi cell can be covered by its k^{th} generating nodes iff the relationship of its $(k-1)$-order DT's three edges satisfy the following condition:

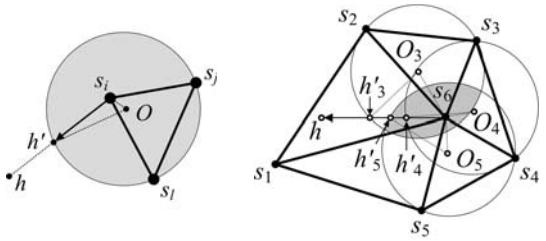
$$R_s \geq \frac{d_{ij} d_{il} d_{jl}}{\sqrt{\left(d_{ij}^2 + d_{jl}^2 + d_{il}^2\right)^2 - 2\left(d_{ij}^4 + d_{jl}^4 + d_{il}^4\right)}}. \quad (6)$$

Definition 3.6: (*Safe area*) Consider a $(k-1)$-order DT $\triangle s_i s_j s_l$, where s_i and s_j are not moving, the *safe area* of s_k for $\triangle s_i s_j s_l$ is defined as the area, where s_k can be located to guarantee that the triangle's circumcenter is k-covered (satisfying Equ. (6)).

According to Definition 3.6 above, we calculate s_l's *safe area* for $\triangle s_i s_j s_l$. According to Equ. (6), we can get that the location of s_l should satisfy one of the following two equations, where (x_l, y_l), (x_i, y_i) and (x_j, y_j) represent the Cartesian coordinates of s_l, s_i and s_j respectively:

$$(x_l - x' + f_x)^2 + (y_l - y' + f_y)^2 \leq R_s \quad (7)$$

$$(x_l - x' - f_x)^2 + (y_l - y' - f_y)^2 \leq R_s \quad (8)$$



(a) Single safe area (b) United safe area

Figure 5. Calculating the movement destination according to safe area

where

$$x' = \frac{x_i + x_j}{2},\ f_x = |x_j - x_i| \sqrt{\left(\frac{R_s}{d_{ij}}\right)^2 - \frac{1}{4}}$$

$$y' = \frac{y_i + y_j}{2},\ f_y = |y_j - y_i| \sqrt{\left(\frac{R_s}{d_{ij}}\right)^2 - \frac{1}{4}} \times \frac{x_i - x_j}{y_j - y_i}.$$

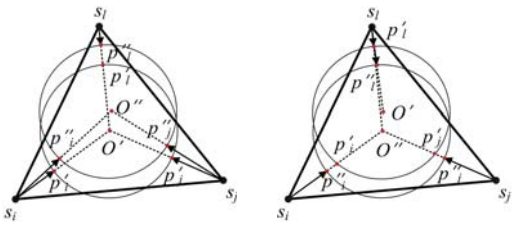
Note that a node can be the generating node for multiple critical points. When node s_l needs to move to cover a hole (*i.e.*, an uncovered critical point), it should not move out of its safe area of another covered critical point, that is, its location (x_l, y_l) should satisfy Equ. (7) and Equ. (8). Then, we achieve Proposition 3.2, which presents a necessary condition to k-cover a point.

Proposition 3.2: Let d_{ij} denote the distance between any pair of two vertices of a $(k-1)$-order DT, then $d_{ij} \leq 2R_s$ is a necessary condition to k-cover the center of the circumcircle of the triangle.

Proof: Suppose for the sake of contradiction that $d_{ij} > 2R_s$, then x_l and y_l have no solutions in Equ. (7) and Equ. (8), implying that d_{ij} should be no larger than $2R_s$. ■

According to Equ. (7) and Equ. (8), we know that a generating node must know the other two generating nodes of this critical point in order to know its safe area for their critical point. Hence, when a node s_i notifies the three generating nodes that their critical point is not covered, it also tells them the location information (x, y) of the critical point and the other two generating nodes. Based on Equ. (7) and Equ. (8), the node limits its movement within its safe area when it is moving.

Previously, we consider the safe area of a node when it belongs to only one $(k-1)$-order DT. Next, we consider the safe area of a sensor when it belongs to multiple $(k-1)$-order DTs. Suppose node s_i is in a number of $(k-1)$-order DTs denoted by $DT_{i,1}$, ...,$DT_{i,m}$ and the corresponding *safe area* of s_i in the triangles are $A_{i,1}$, ..., $A_{i,m}$, where m is the number of the triangles. The *united safe area* A^i_{united} for s_i can be calculated as $A^i_{\text{united}} = \bigcap_{j=1}^{m} A_{i,j}$. We define *coverage rate* of a WSN as the ratio of the area k-covered by



(a) When one node cannot move to its destination (b) When two nodes cannot move to their destinations

Figure 6. Cooperative movement.

given sensor nodes to the area of the entire target region of the WSN. Finally, once the sensor finds itself unable to reach the destination due to the constraint of its safe area, its hole healing responsibility falls to the point's another generating node (sensor).

Lemma 3.3: The coverage rate of a WSN increases monotonically with node movements if the nodes do not break the united safe area constraint.

Proof: Consider that when a node, say s_i, moves towards one hole, the area size of this hole is decreased; whereas if s_i stays in the united safe area it originally is located in, no new hole is generated. Thus, the coverage rate is increased monotonically with node movements. ■

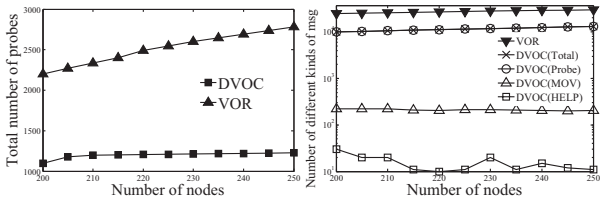
Lemma 3.3 indicates that by obeying the constraint of the safe area, DVOC is guaranteed to converge if the density of sensor nodes is high enough.

IV. Performance Evaluation

In this section, we present the experimental results of DVOC in comparison with typical VORonoi-based algorithm (VOR) [4], [11]. Also, we compare DVOC with the other two typical movement-assist schemes for full coverage in WSNs: Scan-based Movement-Assisted Sensor Deployment (SMART) [18] and Sea Surface Coverage (SSC) [1] on GENI Orbit testbed [19], [20]. The testbed uses a large two-dimensional grid of 400 802.11 radio nodes, which can be dynamically interconnected into specified topologies.

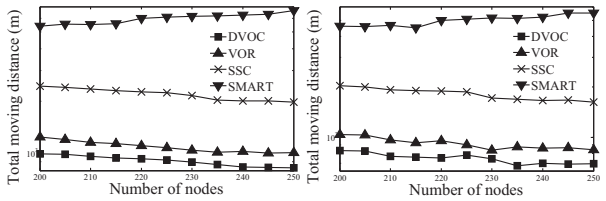
Since GENI-Orbit testbed has limited number of nodes (less than 400 nodes), here we only take the 2-coverage as an example for the k-coverage. The target field is a $400\text{m} \times 400\text{m}$ area. The number of sensors was varied from 200 to 250. The radius of the sensing ranges of sensors was varied from 45 to 50m. We measured the following metrics: 1) *total number of probes* 2) *number of messages* (includes messages for probing, informing other nodes to move, and asking help from partners) 3) *total moving distance*

Transmission Cost. Fig. 7 (a) and (b) show the transmission costs between DVOC and VOR measured by the total number of probes and messages, respectively,



(a) Total number of probes (b) Total number of messages

Figure 7. Transmission cost of DVOC and VOR.



(a) Sensing range = 45m (b) Sensing range = 50m

Figure 8. Total moving distance of different schemes

when the sensing range was set to 50m and the transmission range was set to 70m. Both schemes require that each node probes the locations of its surrounding nodes for building a VD. The difference is that under DVOC, every node needs to send messages when finding a hole or moving to heal it. The number of messages of DVOC includes the messages for probing (Probe), informing other nodes to move to heal holes (MOV), and asking for help from partners (HELP). We did not count the number of messages forcing partner to move (FORCE) since the number of such messages is extremely small. For VOR, only the messages for probing are included. From Fig. 7 (a) and (b), we find that in both metrics, DVOC is significantly superior to VOR. The total number of messages in VOR (all for probing) is about 2 (range from 1.936 to 2.176) times of that of DVOC, and the number of messages transmitted in DVOC is only about half of VOR's (range from 0.512 to 0.546). We also measured the proportion of different types of messages in DVOC; on average, probe messages constitute 98.60% (this explain why in Fig. 7 (b) the curve of the number of probes and the curve of the total number of messages are very close); MOV messages constitute 1.21% and HELP messages constitute 0.19%. The reason why DVOC has higher efficiency than VOR in terms of transmission cost is that DVOC uses a cooperation mechanism based on a local k-order VD, which requires less location information than the partial k-order VD used in VOR. Admittedly, DVOC still makes nodes communicate with each other after the VD is built, but such communication cost is very small compared with the cost for probing location information.

Energy Cost and Delay. Fig. 8 (a) and (b) show

that the *total moving distance* versus the number of nodes in different schemes. From the figures, we find that DVOC has the shortest *total moving distance*. On average, the *total moving distances* of VOR, SMART, and SSC are respectively 1.23, 2.45 and 6.65 times as long as that of DVOC when the sensing range is 45m, and are respectively 1.25, 2.08 and 8.65 times as large as that of DVOC when the sensing range is 50m. SSC generates a much longer total moving distance than VOR and DVOC. This is because SSC is a cell-based algorithm, in which every node is only able to move in four directions and cannot move along a straight line to the target. In contrast, VOR enables nodes to move in all directions, leading to a shorter total moving path than SSC. Both DVOC and VOR use VDs for hole detection and healing. The difference between DVOC and VOR's movement patterns is that DVOC is a cooperation based algorithm where each node needs to calculate whether it will move out of its *safe area* when informed to recover holes. If the destination is outside of its safe area, the node stops moving and asks for partners' help in covering the hole. Though it is possible that all the partners cannot move to their destinations, the cooperation mechanism in DVOC can prevent new holes from being generated in most cases.

V. Conclusions

In this paper, we propose a Distributed VOronoi-based Cooperation Mechanism for full coverage (DVOC). In this scheme, each node builds its own *local k-order VD* and helps other nodes to detect holes within their own diagrams. With the guarantee of the accuracy of the local k-order VDs, DVOC can greatly alleviate the burdens on nodes for transmission while ensuring no holes are missed. Further, DVOC uses a cooperation mechanism to prevent generating new holes during node movement, which greatly increases the efficiency of node movements. Experimental results from GENI's ORBIT testbed shows that DVOC has superior performance than previous schemes in terms of energy-efficiency and efficiency of coverage. In our future work, we will study techniques to shorten the moving distance to minimize the sum of all moving paths, *i.e.*, globally optimized paths.

VI. Acknowledgement

This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, and Microsoft Research Faculty Fellowship 8300751.

References

[1] J. Luo, D. Wang, and Q. Zhang, "Double mobility: Coverage of the sea surface with mobile sensor networks.," in *Proc. of INFOCOM*, 2009.

[2] A. M.-C. So and Y. Ye, "On solving coverage problems in a wireless sensor network using voronoi diagrams," in *Proc. of WINE*, 2005.

[3] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *Proc. of WSNA*, 2003.

[4] G. Wang, G. Cao, and T. F. L. Porta, "Movement-assisted sensor deployment," in *Proc. of INFOCOM*, 2004.

[5] A. Ghosh, "Estimating coverage holes and enhancing coverage in mixed sensor networks," in *Proc. of IEEE LCN*, 2004.

[6] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *Proc. of INFOCOM*, 2003.

[7] S. Poduri and G. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proc. of IEEE ICRA*, 2004).

[8] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proc. of DARS*, 2002.

[9] D. Niculescu, "Positioning in ad hoc sensor networks," in *IEEE Network Volume*, 2004.

[10] M. Sharifzadeh and C. Shahabi, "Supporting spatial aggregation in sensor network databases," in *International Symposium of ACM GIS*, 2004.

[11] B. A. Bash and P. J. Desnoyers, "Exact distributed voronoi cell computation in sensor networks," in *SIAM Journal on Computing*, 2007.

[12] J. Gudmundsson, M. Hammar, and M. van Kreveld, "Higher order delaunay triangulations," in *Computational Geometry: Theory and Applications*, 2002.

[13] B. Kar and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. of MobiCom*, 2000.

[14] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, "Computational geometry: Algorithm and applications," *Springer*, 2008.

[15] M. I. Shamos and D. Hoey, "Closest point problems," in *Foundations of Computer Science*, 1975.

[16] D. T. Lee, "On k-nearest neighbor voronoi diagram in the plane," in *IEEE Transaction on Computers*, 1982.

[17] H. Edelsbrunner, J. O. Rouke, and R. Seidel, "Constructing arrangements of lines and hyperplanes with applications," in *SIAM Journal on Computing*, 1986.

[18] S. Yang, M. Li, and J. Wu, "Scan-based movement-assisted sensor deployment methods in wireless sensor networks," in *IEEE TPDS*, 2007.

[19] "GENI project." http://www.geni.net/.

[20] "Orbit." http://www.orbit-lab.org/.