A Delaunay-based Coordinate-free Mechanism for Full Coverage in Wireless Sensor Networks

Chenxi Qiu  
Dept. of Electrical and Computer Engineering  
Clemson University  
Clemson, USA  
chenxiq@clemson.edu

Haiying Shen  
Dept. of Electrical and Computer Engineering  
Clemson University  
Clemson, USA  
shenh@clemson.edu

Abstract—Recently, many schemes have been proposed for detecting and healing coverage holes to achieve full coverage in wireless sensor networks (WSNs). However, none of these schemes aim to find the shortest node movement paths to heal the coverage holes, which could significantly reduce energy usage for node movement. Also, current hole healing schemes require accurate knowledge of sensor locations; obtaining this knowledge consumes high energy. In this paper, we propose a DElaunay-based Coordinate-free Mechanism (DECM) for full coverage. Based on rigorous mathematical analysis, DECM can detect coverage holes and find the locally shortest paths for healing holes in a distributed manner without requiring accurate node location information. Simulation results and experimental results from the real-world GENI Orbit testbed show that DECM achieves superior performance in terms of the energy-efficiency and effectiveness of hole healing compared to previous schemes.

Keywords—wireless sensor networks; coverage holes; node movement; energy usage;

I. INTRODUCTION

In wireless sensor networks (WSNs), sensor nodes may die due to battery drain or environmental causes. Also, nodes may deviate from their assigned positions due to the effects of uncontrollable factors (e.g., ocean waves). Coverage holes hamper the ability of WSNs to detect events and reduce network reliability. Therefore, it is crucial to equip the sensor nodes with efficient hole detection and healing capabilities in order to ensure full coverage of the target field.

Numerous schemes have been proposed for healing coverage holes in WSNs. Many of these [1]–[13] leverage sensor movement to improve network coverage. Since mechanical movement is much more energy-expensive than electronic communications [14], the node moving distance should be minimized [15]. However, none of the previous hole healing works aim to find the shortest paths of node movement, which could greatly enhance the energy-efficiency. Also, all of these schemes require accurate knowledge of node locations. However, simple localization solutions, such as equipping each node with a GPS receiver or manual configuration using coordinates [4], [16]–[18], are either energy-expensive or impractical for WSNs in some cases [19].

To overcome the drawbacks, we propose a DElaunay-based Coordinate-free Mechanism (DECM) for full coverage in WSNs. Based on rigorous mathematical analysis, DECM can find the locally shortest paths for node movement in healing coverage holes in a distributed manner without accurate location information.

In this paper, we first present a mathematical model that provides a sufficient and necessary condition for the full coverage of a triangle that has no other nodes inside its circumcircle. DECM is a distributed scheme in which every node checks for holes and makes movements to heal holes. As shown in Fig. 1, each node first conducts Delaunay triangulation that divides the target field into triangles that have no other nodes inside. Then, based on the sufficient and necessary condition, each node calculates the its safe area where the node can be located while still keeping full coverage of its triangles. Based on the calculated safe areas of each node, DECM can detect coverage holes and find the shortest paths for node movement to heal the holes. DECM utilizes the r-map coordinate system [20] to enable nodes to know the relative locations of nearby nodes for hole detection and healing.

DECM is similar to a Voronoi diagram based method (VOR) [7] since both construct a diagram for hole healing. VOR forms a WSN into Voronoi cells, each of which has one sensor called a 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, if a generating node finds some points in a Voronoi cell that are not covered by itself, it moves directly to the farthest point to heal the hole. However, VOR may generate numerous iterations of node movements because (1) node movement to cover one point rather than the cell area may generate holes in other points in the cell, and (2) only one node is in charge of the coverage of a cell. Unlike in VOR, a node in DECM moves to a point in order to cover the entire area of a Delaunay cell. Also, three nodes are in charge of the coverage of a cell. Thus, DECM achieves full coverage more quickly and
energy-efficiently.

This paper is organized as follows. Section II presents a concise review of related works on movement-assisted schemes for full WSN coverage deployment. Section III introduces mathematical models for analyzing WSN coverage problems and presents the DECM movement-assisted scheme for detecting and healing coverage holes based on this model. Then, Section IV presents a performance evaluation of DECM in comparison with several previous schemes. The final section concludes with a summary of contributions and a discussion on further research work.

II. RELATED WORKS

Node movement strategies for full area coverage have gained considerable attention during recent years. In virtual force methods [6], [8]–[10], sensors are likened to electromagnetic particles and have repulsive and attractive forces between them. When the distance between two sensors is too great, the attractive force makes them pull each other closer; when the distance is too small, the repulsive force makes them push each other away. 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 cost of node movement.

Wang et al. [7] used the Voronoi diagram for healing coverage holes. As explained in the previous section, in these methods, a node’s movement cannot completely heal the coverage holes in its Voronoi cell because one hole healing may produce other holes in its Voronoi cell, subsequently generating many iterative node movements.

Many grid quorum-based movement schemes [1]–[5] view the movement-assisted network re-deployment problem as a load balancing problem under the virtual grid model [21]. These schemes partition an entire target region into small grid cells, and consider the number of nodes in each cell as the cell’s load. The schemes schedule sensor movement in order to achieve a balanced load distribution among the grid cells. A node within a grid cell can directly communicate with other nodes in its four adjacent cells and makes movement decisions according to information from adjacent cells. Some of these schemes also try to minimize the sensor movement distances. For example, SMART [5] arranges communication between cells heads to identify overloaded and underloaded cells and direct nodes from overloaded cells to move to underloaded cells. Knowing the loads of other cells, each cell tries to avoid any unnecessary movement; thus, both the total moving distance and the total number of moves can be minimized. However, since the target a node moves to is a cell rather than a specific point, the schemes still cannot find the shortest moving paths.

There are other movement-assisted methods using different strategies. Luo et al. [15] 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. Heo and Varshney [13] proposed an intelligent energy-efficient deployment algorithm for cluster-based WAN by synergistic combination of cluster structuring and a peer-to-peer deployment scheme. Zhang and Arora [12] proposed the GS³ algorithm for multi-hop wireless networks. The algorithm enables nodes in a 2D plane to configure themselves into a cellular hexagonal structure such that cells have tightly bounded geographic radii and low overlap between neighboring cells. Butler and Rus [11] proposed an approach for positioning and organizing mobile sensors in response to events in their environment.

Admittedly, the above movement-assisted schemes have their own merits. However, all of these schemes require nodes to have accurate location information. Though Bejerano et al. [20] proposed a locality-free scheme, it is for hole detection rather than for hole healing. Also, none of these schemes aim to find the shortest movement paths only using local location information. DECM can find the locally shortest moving paths for healing coverage holes in a distributed manner without accurate location information.

III. THE DESIGN OF DECM

We consider a WSN comprised of numerous mobile nodes that are uniformly distributed over a large target field and are designed to detect specified events. Each node, denoted \( s \), can sense specified events in its sensing range, denoted by \( R_s \).

We first find the condition for full coverage of a triangle formed by three sensors with no other nodes inside the triangle’s circumsircle (Section III-A). Since a Delaunay triangle has no other nodes inside the triangle’s circumsircle, nodes conduct Delaunay triangulation [22] to divide the field into Delaunay triangles (Section III-B). Then, each node uses our observed full coverage condition to find a safe area where it can be located while still keeping full coverage of its triangles (Section III-C). Finally, each node can detect holes and discover the shortest path for its movement to heal holes (Section III-D).

A. Condition for A Triangle’s Full Coverage

Consider three nodes in a plane that construct a triangle. The sufficient and necessary condition for the triangle’s full coverage is described in Theorem 3.1.

**Theorem 3.1:** Consider a triangle formed by three nodes \( s_i, s_j \) and \( s_k \) with no other nodes placed inside the triangle’s circumsircle. Using \( d_{ij} \) to denote the distance between \( s_i \) and \( s_j \), and with the same convention for the other distances, we derive:

1) when the triangle is an acute triangle, the triangle is fully covered iff the following condition is satisfied:

\[
R_s \geq \sqrt{\frac{d_{ij}d_{jk}d_{ik}}{\left( d_{ij}^2 + d_{jk}^2 + d_{ik}^2 \right)^2 - 2 \left( d_{ij}^4 + d_{jk}^4 + d_{ik}^4 \right)}} \tag{1}
\]

2) when the triangle is an obtuse triangle, the triangle is fully covered iff the following condition is satisfied:

\[
R_s \geq \max \left\{ \frac{d_{ij}^2d_{jk}}{d_{ij}^2 + d_{jk}^2 - d_{ik}^2}, \frac{d_{ij}^2d_{ik}}{d_{ij}^2 + d_{ik}^2 - d_{jk}^2}, \frac{d_{jk}^2d_{ik}}{d_{jk}^2 + d_{ik}^2 - d_{ij}^2} \right\} \tag{2}
\]
Definition 1 (Delaunay triangulation [22]) A triangulation for a set $S$ of points in a plane is a Delaunay triangulation if no point in $S$ is inside the circumcircle of any triangle.

Based on Theorem 3.1, we first conduct Delaunay triangulation so that there are no other nodes placed inside each triangle’s circumcircle. Before we present how to conduct Delaunay triangulation on a WSN, we first introduce some definitions and theorems [22].

Definition 2 (edge flip) [22]: As Fig. 3 shows, consider an edge $e = s_is_j$ of a triangulation. If $e$ is not an edge of the unbounded face, it is and incident to two triangles $s_is_js_k$ and $s_is_js_l$. If the two triangles form a convex quadrilateral, we can obtain a new triangulation $T'$ by removing $s_is_j$ and inserting $s_ls_k$ in triangulation $T$. We call this operation an edge flip.

Definition 3 (illegal edge) [22]: As Fig. 3 shows, after an edge flip, the only difference between $T$ and $T'$ is that the six angles $\alpha_1, ..., \alpha_6$ are replaced by $\alpha'_1, ..., \alpha'_6$. We call the edge $e = s_is_j$ an illegal edge if

$$\min(\alpha_i) < \min(\alpha'_i) \quad (1 \leq i \leq 6)$$  \hspace{1cm} (3)

Definition 4 (legal triangulation/triangle) [22]: A triangulation/triangle that does not contain any illegal edge. 

Theorem 3.2: A triangulation for a set $S$ of points in a plane is a legal triangulation iff the triangulation is a Delaunay triangulation [22].

The Delaunay triangulation and hole healing require each node to obtain the distances and directions of nearby nodes. For this purpose, DECM uses the $r$-map system [20], in which each node measures the locations of its neighbors using its own arbitrary polar-axis.

Definition 5 ($r$-map [20]): The $r$-map of node $s_i$ is a variant of polar coordinates that specifies the relative location of its $r$-vicinity, denoted $N_{r_i}(r)$. The location of any node $s_j \in N_{s_i}(r)$ is presented as $(d_{ij}, \theta_{ij})$, where $d_{ij}$ is the radial coordinate that indicates the Euclidean distance between nodes $s_i$ and $s_j$, and $\theta_{ij}$ is the angular coordinate of node $s_j$ that denotes the direction of $s_j$ relative to an arbitrary polar-axis of $s_i$.

Neighboring nodes periodically exchange their measured $r$-maps with their neighbors and receive $r$-maps. When $s_i$ receives $s_j$’s $r$-map containing the location of $s_k$ measured by $s_j$, $s_i$ can transform $s_k$’s location to the location measured by $s_i$’s polar-axis using the methods in [20], [23].

Definition 6 (shared nodes): Shared nodes of $s_i$ and $s_j$ are nodes that exist in both $s_i$’s $r$-map and $s_j$’s $r$-map.

Definition 7 (potential edge): A potential edge is an edge that has at least one side with no constructed triangle and with at least one shared node.

In the Delaunay triangulation algorithm, DECM first selects one node as a seed node. The seed node finds its nearest neighbor and builds an edge to it. The two nodes connected by this edge (edged nodes) choose a nearby node to form a triangle so that the triangle’s circumcircle is minimized. We call this node the nearest triangle neighbor (NTN) of the edged nodes or the edge.
To find the NTN on one side of edge \( e_{ij} \), two edged nodes, \( s_i \) and \( s_j \), communicate with each other to determine their shared nodes, denoted \( N_{s_i}(r) \cap N_{s_j}(r) \). Suppose \( s_{k_1}, s_{k_2}, \ldots, s_{k_m} \in N_{s_i}(r) \cap N_{s_j}(r) \). As shown in Fig. 4, for one side of \( e_{ij} \), all nodes \( s_{k_1}, s_{k_2}, \ldots, s_{k_m} \) are connected with \( s_i \) (or \( s_j \)), thus generating edges \( e_{k_1}, e_{k_2}, \ldots, e_{k_m} \) (or \( e_{k_1}, e_{k_2}, \ldots, e_{k_m} \)). These edges’ perpendicular bisectors intersect with the perpendicular bisector of \( e_{ij} \) at point \( O_{k_1,ij}, O_{k_2,ij}, \ldots, O_{k_m,ij} \), respectively. We use variable \( h_{k,ij} \) to denote the distance between \( O_{k,ij} \) and \( e_{ij} \), i.e., \( h_{k,ij} = M_{ij}O_{k,ij} \), where \( M_{ij} \) is the middle point of \( e_{ij} \). If \( O_{k,ij} \) and \( s_k \) are located at the same side of \( e_{ij} \), then \( h_{k,ij} > 0 \); and if \( O_{k,ij} \) and \( s_k \) are located at different sides, then \( h_{k,ij} < 0 \). Suppose \( h_{k,n,ij} \) has the smallest value among \( h_{k,ij} \); then \( s_{k_n} \) is the NTN of \( e_{ij} \).

As shown in Fig. 3, an edge \( e_{ij} = s_is_j \) can be used for building one triangle on each of its two sides. Suppose the formed triangle is \( \triangle s_is_j s_k \). When two nodes are connected by a new edge (e.g., \( s_is_k \) and \( s_js_k \)), if the new edge is a potential edge, the edged nodes conduct the same operation by choosing their NTN to construct a new triangle with the minimum circumcircle.

Each node stores the triangles it belongs to in its Delaunay triangulation table (DT-table). This process continues until a newly added edge intersects with an existing edge, which means the triangulation is completed. However, it is difficult to globally notify all nodes of completion. Also, a node isolated from the others may keep looking for nodes to build triangle edges. Thus, DECM sets an appropriate time period for each node to conduct the triangulation process. After the period, each node checks to see if an illegal edge exists in the triangles in its DT-Table, and conducts an edge flip if an illegal edge exists. This step is to ensure that Delaunay triangulation is formed according to Theorem 3.2. There should not be many illegal edges because:

**Theorem 3.3:** When two edged nodes connect to their NTN to construct a triangle, there are no other nodes inside the triangle’s circumcircle on the same side as the NTN.

**Proof:** As Fig. 5 (a) and (b) show, \( s_k \) is the NTN of \( e_{ij} \), and \( s_i \), \( s_j \), and \( s_k \) construct a triangle (either acute or obtuse), in which \( O_{k,ij} \) is the circumcenter of the triangle. Suppose there is one sensor node \( s_{k'} \) located inside the triangle on the same side of \( e_{ij} \) as \( s_k \) (i.e., \( s_kO_{k,ij} < s_jO_{k',ij} \)). The perpendicular bisector of \( s_is_{k'} \) definitely intersects \( s_iO_{k,ij} \) and \( M_{ij}O_{k,ij} \) at point \( O_{k',ij} \). Therefore, \( h_{k',ij} = h_{k,ij} - O_{k,j}O_{k',ij} \), which indicates that \( h_{k,ij} \leq h_{k',ij} \). This contradicts the definition of the NTN.

Note that two edged nodes are not necessarily neighbors. Nodes can communicate with each other by multi-hop routing [24]. Though two edged nodes calculate their NTN individually, they will find the same NTN result when their r-maps include their nearby nodes. Also, because only one NTN exists on one side of a potential edge of two edged nodes, it is impossible that a new edge will intersect with an existing edge of the two nodes.

The Delaunay triangulation can take \( O(n^2) \) (\( n \) is the number of nodes in the network) edge flips even if all node locations are globally known [22]. A local edge flip might generate new illegal triangles, and then the computation might be endless, though the probability of such an event is low according to Theorem 3.3. To avoid endless edge flipping, we set a Time To Live (TTL) for each sensor node. Once the TTL expires, a node stops flipping edges even though it finds illegal triangles. The TTL strategy does not prevent DECM from achieving full coverage but might generate unnecessary node movements. The strategy might reduce the accuracy of Delaunay triangulation since there could exist some illegal triangles. That is, some other node besides the triangle’s three nodes could cover the circumcircle’s center. Then, the coverage holes that the three sensor nodes have detected might be covered by some other nodes, leading to unnecessary node movements. In this paper, we assume that the node density is high enough for Delaunay triangulation and that the number of node movement iterations for hole healing is limited.

Specifically, in conducting Delaunay triangulation on a WSN, each node \( s_i \) executes Algorithm 1.

**Algorithm 1** The algorithm for Delaunay triangulation executed by \( s_i \).

**Step 1:** If it is selected as the seed node, go to Step 2; otherwise, go to Step 3.

**Step 2:** Find the nearest node in its r-map, say \( s_j \), and build an edge to \( s_j \). Store edge \( e_{ij} \) into its DT-table and also ask \( s_j \) to store \( e_{ij} \) into its DT-table.

**Step 3:** Check if its DT-table contains potential edges. If yes, find the NTN of each potential edge and build an edge to the NTN; otherwise, go to Step 4.

**Step 4:** Check whether the time is expired. If no, go to step 3; otherwise go to Step 5.

**Step 5:** In TTL, flip edge for each existing illegal edge.

**C. Safe Area Detection**

**Definition 8 (safe area):** Consider a triangle formed by three nodes \( s_i, s_j, \) and \( s_k \), where \( s_i \) and \( s_j \) are not moving. The safe area of \( s_k \) for \( \triangle s_is_js_k \) is defined as the area where \( s_k \) can be located without breaking the full coverage of \( \triangle s_is_js_k \).

In the following, we calculate \( s_k \)’s safe area. We assume that \( s_i \)’s r-map contains \( s_j \) and \( s_k \) and use \( s_i \)’s r-map for the calculation. Using the r-maps of \( s_j \) or \( s_k \) can retrieve the same results. First, we assume that \( \theta_{ik} \in [\theta_{ij}, \theta_{ij} + \pi) \). We will discuss the situation when \( \theta_{ik} \in [0, \theta_{ij}] \cup [\theta_{ij} + \pi, 2\pi) \) later on. There are three different kinds of triangle shapes for \( \triangle s_is_js_k \):

Case I: an acute triangle (Fig. 6 (a));
To guarantee full coverage of the triangle, Eq. (2) can be reduced to:

\[ \frac{d_{ik}^2 - 2d_{ik}d_j \cos (\theta_{ik} - \theta_j) + d_{jk}^2}{d_{ij} - d_{ik} \cos (\theta_{ik} - \theta_j)} \leq 2R_s \]  

(11)

Similarly, we notice that when \( d_{ik} < 2R_s \), if Eq. (8) is satisfied, Eq. (11) is automatically satisfied, indicating that the triangle is fully covered. This is because Eq. (8) can be reduced to

\[ d_{ij}^2 - 2d_{ik}d_j \cos \theta'_{ik}d_{ij} + d_{jk}^2 \leq d_{ij}^2 - d_{ik} \cos \theta'_{ik}d_{ij}, \]  

(12)

where \( \theta'_{ik} = \theta_{ik} - \theta_j \). Because \( d_{ij} - d_{ik} \cos (\theta_{ik} - \theta_j) > 0 \) in Eq. (8), Eq. (12) can be reduced to

\[ \frac{d_{ik}^2 - 2d_{ik} \cos (\theta_{ik} - \theta_j)d_{ij} + d_{jk}^2}{d_{ij} - d_{ik} \cos (\theta_{ik} - \theta_j)} < d_{ij} < 2R_s \]  

(13)

Thus, Eq. (11) is satisfied. Accordingly,

**Lemma 3.2:** For an obtuse triangle with \( \triangle s_{i,j} s_k \) (Case II), when \( d_{ij} < 2R_s \), Eq. (8) is a sufficient condition for the full coverage of the triangle, and the area formed by Eq. (8) is the safe area of \( s_k \), which is the gray area in Fig. 6 (c); when \( d_{ij} \geq 2R_s \), the area formed by Eq. (8), Eq. (9), and Eq. (11) is the safe area of \( s_k \), which is the gray area in Fig. 6 (b).

**Case III** When \( \triangle s_{i,j} s_k \) is an obtuse triangle with \( d_{ij} < \max \{ d_{ik}, d_{jk} \} \) (Case III), when \( d_{ij} < 2R_s \), Eq. (8) is a sufficient condition for the full coverage of the triangle. We discuss the problem in the following two cases:

**Case III.1** When \( d_{ik} > \max \{ d_{ij}, d_{jk} \} \): As Fig. 6 (d) shows, \( d_{ij}, d_{jk}, \) and \( d_{ik} \) must satisfy \( d_{ik}^2 > d_{ij}^2 + d_{jk}^2 \), which can be reduced to:

\[ d_{ik} \cos (\theta_{ik} - \theta_j) > d_{ij} \]  

(14)

where \( \cos (\theta_{ik} - \theta_j) > 0 \) because \( d_{ik} > d_{jk} \). To guarantee full coverage of the triangle, Eq. (2) in Theorem 3.1 should be satisfied. We discuss the problem in two cases:

**Case III.1.1** When \( d_{ik} > d_{ij} \geq d_{jk} \): From Eq. (2), the safe area of \( s_k \) can be calculated as:

\[ \frac{d_{ij}}{2 \cos (\theta_{ik} - \theta_j)} \leq R_s \quad \text{and} \quad \frac{d_{ik}}{2 \cos (\theta_{ik} - \theta_j)} < d_{ij} \]  

(15)

From Eqs. (14) and (15), we can infer that:

\[ \frac{d_{ij}}{2R_s} \leq \cos (\theta_{ik} - \theta_j) < 1 \to d_{ij} < 2R_s \]  

(16)

**Case III.1.2** When \( d_{ik} > d_{jk} > d_{ij} \): From Eq. (2), the safe area of \( s_k \) can be calculated as:

\[ \frac{d_{ik}^2 - 2d_{ik}d_j \cos (\theta_{ik} - \theta_j) + d_{jk}^2}{2(d_{ij} - d_{ik} \cos (\theta_{ik} - \theta_j))} \leq R_s \]  

(17)

\[ \frac{d_{ik}}{2 \cos (\theta_{ik} - \theta_j)} > d_{ij} \]  

(18)
Recall that \( d_{ij} < 2R_s \) when \( d_{ij} > d_{jk} \). Similarly, we can derive that \( d_{jk} < 2R_s \) when \( d_{jk} > d_{ij} \). Then,

\[
d_{ij} < d_{jk} < 2R_s \rightarrow d_{ij} < 2R_s
\]

From Eqs. (16) and (19), we know that

**Lemma 3.3:** \( d_{ij} < 2R_s \) is a necessary condition for the full coverage of an obtuse triangle with \( d_{ik} > \max\{d_{ij}, d_{jk}\} \) (Case III). The area formed by Eq. (14), Eq. (15), and Eq. (17) is the safe area of \( s_k \), which is the gray area in Fig. 6 (c).

**Case III.2** When \( d_{jk} \geq \max\{d_{ij}, d_{ik}\} \): If we use \( s_j \)'s r-map to calculate \( s_k \)'s safe area in the obtuse triangle, because this is symmetrical to Case III.1, the calculated \( s_k \)'s safe area is similar to that of case III.1 shown in the slashed area of Fig. 6 (d). In other cases, using \( s_j \)'s r-map to calculate \( s_k \)'s safe area leads to exactly the same result. When \( \theta_{ik} \in [0, \theta_{ij}] \cup [\theta_{ij} + \pi, 2\pi] \), it only results in a safe area that is symmetrical to the area when \( \theta_{ik} \in [\theta_{ij}, \theta_{ij} + \pi] \) as shown in the dotted and gray areas in Fig. 6 (a)-(d).

From Lemmas 3.1, 3.2, and 3.3, we know that when \( s_i \) looks for the safe area of \( s_k \) for full coverage of \( \triangle s_is_js_k \), given \( d_{ij} \), if \( d_{ij} \geq 2R_s \), \( s_k \)'s safe area is only Fig. 6(b) in Case II, which is re-drawn in Fig. 7 (b). If \( d_{ij} < 2R_s \), \( s_k \)'s safe area can be Fig. 6 (a), (c), and (d) in Case I, II, and III, the combination of which is shown in Fig. 7 (b).

**Theorem 3.4:** When \( s_i \) checks the full coverage of \( \triangle s_is_js_k \), given \( d_{ij} \) when \( d_{ij} \geq 2R_s \), the gray area in Fig. 7(b) is the safe area of \( s_k \); otherwise, the gray area in Fig. 7(a) is the safe area of \( s_k \).

If every node only has the relative location information of other nodes, it would be difficult for the system to detect holes on the edge of the target region [20]. There are several ways to solve this problem, such as configuring nodes before placing them. In our system, we deploy a small number of static nodes on the edge of the region as anchor nodes to help detect holes.

**D. Shortest-path Movement**

Recall that after the Delaunay triangulation, each node stores the nodes that are in the same triangle as itself in its DT-Table. We call these nodes triangle neighbors of the node. Each node calculates the safe areas of its triangle neighbors, checks to see if they are in their own safe areas using its r-map based on Theorem 3.4, and then notifies those which are not in their own safe areas. After a node receives an out-of-safe-area notification, it calculates the target point that leads to the shortest path in order to cover the hole. A node may receive multiple out-of-safe-area notifications, similar to VOR [7]; the node then moves to the farthest target point in order to fix the largest hole, so that the sizes of all holes in the WSN are quickly reduced. Differently from VOR, in which a node movement may lead to numerous iterations to cover a hole, DECM enables a node to move to its destination directly to cover a hole and meanwhile keep the full coverage of its triangle. In the following, we present the algorithm to calculate the shortest path for node movement.

![Figure 7. Shortest path in node movement for hole healing.](image-url)

Assume that \( s_k \) receives the notification from \( s_i \) along with \( d_{ij} \) and \( s_k \)'s safe area. \( s_k \) then calculates its shortest path and the target point to move into its safe area. Similar to hole detection, \( s_k \) needs to consider two cases according to \( d_{ij} \).

**Case I** When \( d_{ij} < 2R_s \): As Fig. 7 (a) shows, the safe area for \( s_k \) is composed of two parts: area I and area II. From Eq. (7) we can get the polar coordinates of \( C_1 \)'s center \( O_1 \) and \( C_2 \)'s center \( O_2 \): \( (R_s, \arccos \left( \frac{d_{ij}}{2R_s} \right) + \theta_{ij}) \) and \( (R_s, 2\pi - \arccos \left( \frac{d_{ij}}{2R_s} \right) - \theta_{ij}) \). \( s_k \) has two choices: (1) constructing an acute triangle (moving towards area I), and (2) constructing an obtuse triangle (moving towards area II). We only let nodes construct acute triangles because this makes the node positions more similar to the vertices of the static hexagon based permutation, which is considered the best for node distribution [25]. To find the shortest path and target point to move to region I, using itself as the circle center, \( s_k \) draws a circle \( C' \), which generates a point of tangent between \( C' \) and region I, denoted \( T \); then \( s_k \) moves towards \( T \). Thus, to cover the hole, \( s_k \) moves towards point \( O_1 \) or \( O_2 \), whichever is nearer, and stops once it enters its safe area. As a result, it moves along the shortest path, the length of which is:

\[
s_k T = \sqrt{\left( x_{ik} - \frac{d_{ij}}{2} \right)^2 + \left( y_{ik} - \sqrt{4R_s^2 - d_{ij}^2} \right)^2} - R_s
\]

where \( x_{ik} = d_{ik} \cos (\theta_{ik} - \theta_{ij}) \), \( y_{ik} = d_{ik} \sin (\theta_{ik} - \theta_{ij}) \).

Node \( s_k \) can independently calculate the coordinates of \( O_1 \) and \( O_2 \) in its own r-map. We notice that the polar coordinates of point \( O_1 \) and \( O_2 \) must be positioned at \( s_i \) and \( s_j \)'s perpendicular bisector and that \( O_1s_i = O_2s_i = R_s \). Accordingly, the coordinates of \( O_1 \) in \( s_k \)'s r-map, \((d_{kO_1}, \theta_{kO_1})\), can be calculated by:

\[
d_{kO_1} = \sqrt{(A + B)^2 + (C + D)^2}
\]

\[
\theta_{kO_1} = \begin{cases} \arctan \left( \frac{C + D}{A + B} \right) & \text{if } C + D \geq 0 \\ \arctan \left( \frac{C + D}{A + B} \right) + \pi & \text{if } C + D < 0 \end{cases}
\]

The coordinates of \( O_2 \) in \( s_k \)'s r-map, \((d_{kO_2}, \theta_{kO_2})\), can be calculated by:

\[
d_{kO_2} = \sqrt{(A - B)^2 + (C - D)^2}
\]
\[ \theta_{kO_1} = \begin{cases} \arctan \left( \frac{C-D}{\sqrt{x^2+y^2}} \right) & \text{if } C-D \geq 0 \\ \arctan \left( \frac{C-D}{\sqrt{x^2+y^2}} \right) + \pi & \text{if } C-D < 0 \end{cases} \] (24)

where

\[ A = \frac{d_{ki} \cos \theta_{ki} + d_{kj} \cos \theta_{kj}}{2}, \quad C = \frac{d_{ki} \sin \theta_{ki} + d_{kj} \sin \theta_{kj}}{2} \]

\[ B = \sqrt{R_s^2 - \left( \frac{d_{ij}}{2} \right)^2} \left( d_{kj} \sin \theta_{kj} - d_{ki} \sin \theta_{ki} \right) \]

\[ D = \sqrt{R_s^2 - \left( \frac{d_{ij}}{2} \right)^2} \left( d_{ki} \cos \theta_{ki} - d_{kj} \cos \theta_{kj} \right) \]

and \( d = \sqrt{d_{ki}^2 - 2 \cos (\theta_{ki} - \theta_{kj}) d_{ki}d_{kj} + d_{kj}^2} \)

Based on the coordinates of \( O_1 \) and \( O_2 \), \( s_k \) chooses the one closer to itself as the target point, which leads to the shortest moving path.

**Case II** When \( d_{ij} \geq 2R_s \): As Fig. 7 (b) shows, the polar coordinate of \( O_1 \) and \( O_2 \) in \( s_k \)'s r-map are \( (R_s, \theta_{ij}) \) and \( (d_{ij} - R_s, \theta_{ij}) \), respectively. \( s_k \) also has two choices: (1) moving toward region I, and (2) moving towards region II. Node \( s_k \) chooses the option that leads to the shorter moving path. Specifically, \( s_k \) produces the circle \( C_1 \), which has \( s_k \) as its center and has a tangent with region I at point \( T_1 \). \( s_k \) also produces the circle \( C_2 \), which has \( s_k \) as its center and has a tangent with region II at point \( T_2 \).

If \( T_1 s_k < T_2 s_k \), the direction of \( s_k \)'s movement is towards \( O_1 \) and the length of the shortest path \( T_1 s_k \) is:

\[ T_1 s_k = \sqrt{d_{ik}^2 - d_{ik}d_{ij} \cos (\theta_{ik} - \theta_{ij}) + \frac{d_{ij}^2}{4} - R_s} \] (25)

If \( T_1 s_k \geq T_2 s_k \), the direction of \( s_k \)'s movement is towards \( O_2 \) and the length of the shortest path \( T_2 s_k \) is:

\[ T_2 s_k = \sqrt{d_{ik}^2 - 2d_{ik} \cos \theta_{ik}' \left( R_s - \frac{d_{ij}}{2} \right) + \left( R_s - \frac{d_{ij}}{2} \right)^2 - R_s} \] (26)

where \( \theta_{ik}' = \theta_{ik} - \theta_{ij} \). Similar to Case I, in Case II \( s_k \) can figure out the coordinates of \( O_1 \) and \( O_2 \) through its own r-map’s information. Both \( O_1 \) and \( O_2 \) are located at \( s_i s_j \), and \( O_1 s_i = O_2 s_j = R_s \). The coordinates of \( O_1 \) in \( s_k \)'s r-map, \((d_{kO_1}, \theta_{kO_1})\), can be calculated by:

\[ d_{kO_1} = \sqrt{A^2 + B^2} \] (27)

\[ \theta_{kO_1} = \begin{cases} \arctan \left( \frac{B}{A} \right) & \text{if } B \geq 0 \\ \arctan \left( \frac{B}{A} \right) + \pi & \text{if } B < 0 \end{cases} \] (28)

The coordinate of \( O_2 \) in \( s_i \)'s r-map, \((d_{kO_2}, \theta_{kO_2})\), can be calculated by:

\[ d_{kO_2} = \sqrt{C^2 + D^2} \] (29)

\[ \theta_{kO_2} = \begin{cases} \arctan \left( \frac{D}{C} \right) & \text{if } D \geq 0 \\ \arctan \left( \frac{D}{C} \right) + \pi & \text{if } D < 0 \end{cases} \] (30)

IV. PERFORMANCE EVALUATION

In this section, we present the experimental results from the simulation and the experiments on GENI Orbit testbed [26], [27]. The testbed uses a large two-dimensional grid of 400 802.11 radio nodes, which can be dynamically interconnected into specified topologies. We compared DECM with other three movement schemes for WSN full coverage: VORonoi-based algorithm (VOR) [7], Scan-based Movement-Assisted Sensor Deployment (SMART) [5] and Sea Surface Coverage (SSC) [15]. In SSC, nearby nodes can only move in four directions to inherit others’ lost “interest points”. Since the target region is not fully covered at the beginning in our simulation environment, we assume that every “interest point” in SSC is assigned to its nearest node. We also compared DECM with DECM-R, in which each node randomly selects one target point from multiple notifications, and with DECM-S, in which each node selects the nearest target point from multiple notifications. The nodes in Orbit do not move. We simulated the node movement by letting nodes exchange the information of their virtual locations.

The target field is a 1200m \( \times \) 1200m area. The number of sensors was varied from 290 to 340. The radius of the sensing range was varied from 55.2m to 59.2m and the
transmission range was set to 120m. Initially, we randomly distributed all sensors in the target field. Then we used a scheme to heal the holes until the coverage reaches 99.9%. We measured the following metrics.

(1) **Total moving distance.** This is the sum of the moving distances of all nodes for hole healing. It reflects the delay and energy cost of node movement in hole healing. As in the work in [15], we do not consider the energy consumption for node communication and computing, because it is much lower than that of physical movement.

(2) **Total number of moves.** This is the sum of the number of moves of all nodes for hole healing. Since node movement startup usually consumes more energy than moving, this metric also reflects the energy consumption.

(3) **Coverage.** We distribute 10,000 points uniformly throughout the entire field. The coverage equals the percent points covered. This metric represents the effectiveness of full coverage schemes.

### A. Energy Cost of Healing Holes

Fig. 8 and Fig. 9 show the total moving distance and total number of moves versus the number of nodes in different schemes in simulation and Orbit testbed, respectively. From these figures, we find that DECM has the best performance in total moving distance. The total moving distance follows \( \text{SMART} \gg \text{SSC} > \text{VOR} > \text{DECM} \), and the total number of moves follows \( \text{SSC} > \text{VOR} > \text{DECM} \gg \text{SMART} \) in both simulation and Orbit.

Both SSC and VOR have higher costs in both metrics than DECM because they cannot find the shortest paths for node movement to heal the holes. Long moving paths may produce new holes because a node may not be able to cover its original area after moving, thus resulting in more movements. In DECM, each triangle is managed by three nodes. A node’s movement aims to fully cover its triangle. Even when a node selects its farthest target point when receiving multiple notifications, a potential new hole in the triangle can be healed by the other two sensor nodes. In VOR, each Voronoi cell is managed by only one node. A node’s movement aims to cover one point in its cell, which makes it very likely to generate new holes within its Voronoi cell that cannot be managed by any other nodes. As a result, this node may move back, or other nodes may need to move to cover the hole, resulting in more iterations of node movement. Thus, VOR produces a longer total moving distance and a greater total number of moves than DECM.

It is very intriguing to see that SMART generates a significantly higher total moving distance than the others while producing the lowest total number of moves. Recall that the main task of SMART is to balance the distribution of sensor nodes throughout the entire target region in order to achieve full coverage. Even when the region has no hole, nodes need to make movements to achieve balance. Since the decision of node movement is made by cluster heads, each of which holds all the information of nodes in its cluster, there is only one iteration and every node only needs one movement to reach its final target point. Consequently, SMART generates a long total moving path but a low number of moves.

Fig. 10 and Fig. 11 show the total moving distance and total number of moves versus the node sensing radius in different schemes in simulation and Orbit testbed, respectively. We see that the total moving distance follows SMART \( \gg \) VOR \( \approx \) SSC \( > \) DECM, and the total number of moves follows VOR \( \approx \) SSC \( > \) DECM \( \gg \) SMART in both simulation and Orbit. This is for the same reasons as in Fig. 8. We also observe that for DECM, VOR, and SSC, the two metric results decrease as the radius of sensing range increases. This is because a larger sensing range can reduce the number and size of coverage holes. SMART is not affected significantly by the change of sensing radius because it aims to balance the sensor distribution.

### B. Effectiveness of Healing Holes

Fig. 11 and Fig. 12 show that the coverage versus the total moving distance in different schemes in simulation and Orbit testbed, respectively. Fig. 11 (a) and Fig. 12 (a) show that to achieve 99.9% coverage, the total moving distance is 5600m in DECM, but is over 8000m in other schemes. Fig. 11 (b) and Fig. 12 (b) show that to achieve 99.9% coverage, the total moving distance is 4000m in
DEC M, is 5600m in VOR, and is over 8000m in SSC and SMART. DEC M always finds the shortest paths for healing holes. A shorter movement distance has a lower probability of generating new holes. Also, three nodes managing a triangle area rather than one node makes it easier to heal newly generated holes caused by a node’s movement. More importantly, a node’s movement covers the entire triangle rather than a point. Thus, DEC M avoids excessive iterations of node movement and achieves full coverage more rapidly than other schemes. The figures show that SMART achieves full coverage very slowly. This is because the objective of SMART is to balance the distribution of nodes and thus hole sizes are not decreased rapidly. Comparing (a) and (b) in Fig. 12 and Fig. 12, we see that more sensors help achieve full coverage faster in all schemes because a higher node density reduces the number and size of holes.

C. Healing Holes Due to Dead Nodes

In this experiment, 50 nodes died immediately after all holes were healed from the initial deployment. Fig. 14 (a) and (b), and Fig. 15 (a) and (b) show the total moving distance and number of moves of the schemes in healing the holes caused by the dead nodes in simulation and Orbit testbed, respectively. We see that the total moving distance follows SMART >> SSC > VOR > DEC M, and that the total number of moves follows SSC > VOR > DEC M > SMART in both simulation and Orbit. This is for the same reasons as in Fig. 8. The results show that DEC M still exhibits superior performance over others even with dead nodes.

D. Different Target Point Selections

Fig. 16 and Fig. 17 show the total moving distance and total number of moves of DEC M, DEC M-R and DEC M-S in different schemes in simulation and Orbit testbed, respectively. We see that DEC M generates significantly lower results in both metrics than DEC M-R and DEC M-S. The results verify the effectiveness of DEC M in choosing the farthest target point. It can quickly reduce the sizes of large holes, and hence reduce the number of moves. Also, moving towards large holes can balance the distribution of nodes over the entire target region more quickly. From our observations, when nodes move to heal small-size holes, new holes may arise in the original position. Then, the node will move back to heal the newly generated small holes, leading to more iterations.

Fig. 18 and Fig. 19 show the coverage versus the total moving distance in DEC M, DEC M-R, and DEC M-S when the number of sensors is set to 300 and 330, in simulation and Orbit testbed, respectively. In both figures, DEC M achieves full coverage more rapidly than DEC M-R and DEC M-S. In Fig. 18 (a) and Fig. 19 (a), when the total moving distance reaches 5600m, DEC M achieves 99.9% coverage, while DEC M-R and DEC M-S achieve 98.12% and 96.01% coverage, respectively. In Fig. 18 (b) and Fig. 19 (b), when the total moving distance reaches 4800m, DEC M achieves 99.9% coverage, while DEC M-R and DEC M-S only achieve 98.27% and 96.89% coverage, respectively. Because DEC M always fixes the largest hole first since it selects the farthest target point, it reduces the size of holes more and achieves the full coverage faster than DEC M-R and DEC M-S.

From the experimental results, we find that there is no big difference between the simulation results and Orbit real-world testbed results. It is because in the Orbit testbed, we used virtual location exchanges between static nodes to simulate node movement and there is no packet loss during the testing process. The experimental results verify that selecting the farthest target point when finding several uncovered triangles is the optimal method in healing holes. In summary, the simulation and real testbed results show:

1) DEC M is more energy-efficient for full coverage than other schemes, even when some sensor nodes die.
2) DEC M can heal coverage holes more quickly than other schemes.
3) Selecting the farthest target point when finding several
The number of nodes is 330

Figure 18. Effectiveness of healing holes in DECM with different target point selections (simulation).

Figure 19. Effectiveness of healing holes in DECM with different target point selections (Orbit Testbed).

triangles that are not fully covered is an optimal method in DECM.

ACKNOWLEDGEMENTS

This research was supported in part by U.S. NSF grants OCI-1064230, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, CSR-2008826, CSR-2008827, Microsoft Research Faculty Fellowship 8300751, and Oak Ridge Award 4000111689.

The authors are grateful to Dr. Xiangyang Li for his helpful comments on this paper.

REFERENCES


