Identifying coverage perimeter of heterogeneous wireless sensor networks

WALID OSAMY[1]
AHMED M. KHEDR[1]

Mathematics Dept., Faculty of Science, Zagazig University
Zagazig, Egypt
[1] (w.osamy,amkhedr)@zu.edu.eg

Abstract. Perimeter surveillance is one of the major applications of sensor networks. The perimeter represents the physical extent of the region to be monitored and depending on the application, it is required to sense the intrusion enter the monitored region or leave from the monitored region of the object being monitored. In this paper, the problem of identifying the perimeter of the wireless network's coverage region is addressed. We present a distributed algorithm to find out those sensor nodes that are on the perimeter of coverage (or coverage holes) in the region. We have considered random sensor networks that have sensor nodes with heterogeneous sensing ranges. Our proposed algorithm uses the location neighborhood information to determine if the sensor node enclosed by neighboring nodes, and consequently, if it is located within the interior of the wireless sensor network. We provide performance metrics to analyze the performance of our approach and the simulation shows that the algorithm provides fairly accurate results.

Keywords: coverage perimeter, heterogeneous wireless sensor networks, distributed algorithm.

(Received November 30, 2008 / Accepted April 1, 2009)

1 Introduction

Sensor technologies have become vital today in gathering information about close by environments and its use in wireless sensor networks is getting widespread popular every day. These networks are characterized by a number of sensor nodes deployed in the field for the observation of some phenomena. Due to the limited battery capacity in sensor nodes, energy efficiency is a major and challenging physical problem. The study of wireless sensor networks (WSN) has become a rapidly developing research area that offers fascinating perspectives for combining technical progress with new applications of distributed computing. Typical scenarios involve a large swarm of small and inexpensive sensor nodes, each with limited computing and communication resources, that are distributed in some geometric region; communication is performed by wireless radio with limited range. As energy consumption is a limiting factor for the lifetime of a sensor unit, data communication has to be minimized. Upon startup, the swarm of sensors form a decentralized and self-organizing network that monitors the region. From an algorithmic point of view, the basic characteristic of a sensor network requires working under a paradigm that is different from classical models of computation: absence of a central control unit, restricted capabilities of nodes, and limited communication between nodes require developing new algorithmic ideas that combine methods of distributed computing and network protocols with traditional centralized network algorithms. In other words, how to use a limited amount of strictly local information in order to achieve distributed knowledge of global network properties?

A wireless sensor network for detecting large scale phenomena may be called upon to provide a description of the perimeter of the phenomena. Several phenomena (containment flows) can span large geographic areas. Sensing and detecting the perimeter of the phenomena can help scientists understand what factors affect the spread of theses phenomena. A representation of the perimeter of the coverage has the potential to be more concise and therefore more energy efficient that an enumeration of all the sensor nodes in the network for a specific query. We also argue that identifying coverage holes in the network, is not only used to detect regions with low sensor density due to depletion of node power (places where adding new nodes will significantly improve the coverage and connectivity of the network), but could also be used to identify the regions of interest for the end user. We are more concerned with detecting the sensor nodes that are on the perimeter of the region of interest or on the perimeter of the coverage. Since the emergence of coverage holes is unavoidable, distributed as well as centralized algorithms are provided to detect coverage holes in the region and also find the nodes that are on the perimeter of the coverage area. In the context of a deployed field, a distributed computation is said to be localized if the sensor nodes limit their communication to sensors within some neighborhood and achieve a desired global objective. Such algorithms are preferred as each node communicates only with nodes in the neighborhood, the communication overhead remains restricted with an increase in the network size. Secondly, for the similar reasons, these algorithms are robust to network partitioning and node failures.

Contribution : We propose distributed algorithm which is used to identify those sensor nodes that are on the perimeter of coverage (or coverage holes) of the region of interest or of the coverage. The identified perimeter nodes are used to form the perimeter for enclosing regions such as the monitoring region, and the user defined regions. So that, if the user wants to monitor specific region inside the monitoring region and find the perimeter of this region, it only needs the location of the user defined region. The user transmits the coordinate information of four vertexes of the rectangle to the sensor nodes within the user-defined region. The sensor nodes within the user defined region will become an independent wireless sensor network and only the sensor nodes within the network will execute our proposed approach to find out the perimeter of the region. The identified coverage perimeter nodes will be linked together with sensing range to form a loop and responsible for the detection of the target's entering or leaving the monitoring region.

Outline of the paper: The rest of the paper is organized as follows: Section 2 describes related research on our problem. Our problem formulation terms are introduced in Section 3. In Section 4, we introduce our approach to carry the proposed problem. In Section 5, we give an example scenario. The simulation of our approach is presented in Section 6. In Section 7, we conclude our work.

2 Related Research

With an emerging need for environmental monitoring, military surveillance and security protection, research on wireless sensor networks has recently received an increased interest. Many challenging research problems are being looked upon that include scalability of network protocols so as to incorporate larger number of nodes, design of simple and efficient protocols for different network operations, design of power-conserving protocols, design of security mechanisms, and development of exciting new applications that exploit the potential of wireless sensor networks [8]. In these applications, attributions of the monitored area should be collected and analyzed. The edge information is one of these important attributions. In the context of sensor networks, edge detection can be considered as a powerful primitive upon which a variety of other applications can be build. In [4], Chintalapudi and Govindan proposed three different approaches: statistical approach, filter-based method, and the classifier-based approach, all loosely based on known image processing techniques, to identify the nodes within the edge of a sensed phenomenon inside a WSN. The authors define an edge as a region that intersects both the interior and exterior areas of an observed phenomenon. The authors demonstrate the techniques on networks with a single large-scale continuous phenomenon. In [12], Nowak and Mitra discuss a technique for detecting an estimated boundary between two regions of relatively homogeneous sensed data. The technique takes advantage of hierarchical quad-trees in order to identify small clusters that estimate the regions in which the boundary between two sets of sensors with differing sensor readings passes. This technique builds upon Chintalapudi and Govindan's results by considering the existence of multiple large-scale phenomena.

The main difference between [12] and [4] is that a hierarchical network architecture is assumed and utilized in [12] whereas no hierarchy is assumed among sensors in [4]. Another major difference is that the real boundary has been approximated in [12] while only edge sensors were detected in [4]. Also, [4] outperforms [12] in terms of the communication cost, which is critical in WSNs.

In [15], the authors propose a technique for boundary estimation with observations from sensors aggregated and confidence intervals around the true boundary are obtained for a set of points. They have also provided a distributed technique for realizing a non-parametric regression technique. They have also tackled the problem of satisfying accuracy constraints in terms of the range of the confidence intervals, using the optimal number of active sensors. In [17], the authors propose two novel algorithms for outlying sensor identification and event boundary detection. These algorithms are purely local and thus scale well for large WSNs. They also show by simulation results that these algorithms can clearly detect the event boundary and can identify outlying sensors with a very high accuracy. More recently Mallery and Medidi proposed a distributed edge detection technique that identifies connected perimeters for sensed phenomena within wireless sensor networks [1]. Our approach differs from [1, 17, 15, 9, 12, 4] in that the perimeter detection is independent of any other observations made by sensor nodes, i.e., no recorded sensor information is used in the perimeter detection.

Related to the field of edge detection is that of boundary recognition. Boundary recognition techniques can be classified into one of three categories: geometric, statistical and topological [16]. Geometric-based techniques use location information to detect holes in connectivity. Martincic and Schwiebert have proposed a geometric based technique for identifying nodes on the outer perimeter of a WSN using valid enclosing cycles to differentiate internal nodes from nodes on the perimeter [10]. Fang et. al., propose two face routing strategies, one that pre-builds routes around holes in a WSN using Delaunay triangulation and a second simple greedy method for determining boundary cycles in the event that a transmission gets stuck at a node [13]. Statistical methods for boundary detection make assumptions about the statistical properties of a WSN deployment in order to detect holes, in addition to other common assumptions about WSN. Fekete et. al., present a boundary recognition technique that makes a statistical assumption regarding the degree of nodes on boundaries versus those in the interior of the network [6]. In order to handle different possible boundary shapes, the technique dynamically selects an appropriate threshold for the degree of a node in order to differentiate between a boundary node and an interior node. Another statistical approach towards boundary recognition, also proposed by Fekete et. al., computes the "restricted stress centrality" of a vertex, which is the measure of the number of shortest paths of bounded length that go through a vertex [6]. Similar to the first technique, nodes in the interior tend to have a greater centrality than nodes on the boundary. In order to ensure accuracy, both of these techniques require a uniform deployment and an average node degree greater than 100. Ghrist et. al., propose a centralized algorithm that uses homology to detect holes in a WSN for the purpose of determining insufficient coverage area [7]. An algorithm for boundary detection that searches for combinatorial structures called flowers and augmented cycles has been proposed by Kroller et al.[5]. This algorithm does not require the assumption that transmission ranges are perfect disks, instead requiring that the communication graph be a quasi-unit disk graph. Funke and Klein have developed a heuristic for identifying the nodes located on the edge of the network using only connectivity information [14]. The approach constructs hop based isocontours from a single root node and identifies where the contours break. Wang et al. have developed another approach for boundary recognition using only connectivity information [16]. It builds a shortest path tree from a root node and then identify adjacent nodes whose least common ancestor in the shortest path tree is farther away than it should be, thus identifying a connectivity hole. The work reported in [19] proposed a deterministic method for boundary node detection based on localized Voronoi polygons, the technique originated from the computational geometry. The authors in [3], proposed two deterministic localized algorithms for boundary detection in WSNs. Their algorithms are based on two novel computational geometric techniques called localized Voronoi and neighbor embracing polygons. As compared to their previous work in [19], their new algorithms outperforms in terms computation and communication costs. Our technique, unlike other boundary recognition techniques in the literature, considers the case in which every sensor has a different sensing and communication radius. It also operates effectively for networks containing multiple region of interest. Lastly, by taking advantage of location information to ensure the construction of accurate coverage perimeters.

3 Problem Formulation Terms

In order to solve identifying coverage perimeter problem for a region, it is assumed that the sensor networks [18] consisting of a large number of sensors which randomly distributed in geographical region. The perimeter identification algorithm can be applied to any shape of boundary and can be easily extended to border cover any arbitrary shape of a region with minor modifications but the region of interest is assumed to be a square region for the sake of ease of presentation. We will start by giving some definitions that will aid us in developing an algorithm to identifying coverage perimeter nodes.

Definition 1 *For a sensor node S, there is a region, called* sensing region $R(S)$*, which signifies the area in which S can sense a physical phenomenon.*

The *sensing range* of S indicates the maximum distance between S and any point p in the sensing region of S. A point p is covered or monitored by a sensor node S if the Euclidean distance between p and S is less than the sensing range of S. The sensing region of node S_i can be intersect with the sensing region of node S_j if the Euclidean distance between them is less than the sum of the sensing ranges of S_i and S_j. The sensor node S_i is a *coverage neighbor* of S_j if and only if they have non-empty overlapping sensing region, i.e., if $R(S_i) \bigcap R(S_j) \neq \phi$ then S_i is called coverage neighbor of S_j(as shown in Figure 1, S_j, S_l and S_m are coverage neighbors of S_i and both of S_k and S_n are not). The set of *coverage neighbors* of S is represented by $CN(S)$.

Definition 2 Communication region *of a sensor S defines the area in which S can communicate directly with other sensor nodes.*

The maximum distance between S_i and any other node S_j, where S_j is in the communication region of S_i, is called the *communication range* of S_i. S_i can communicate with any sensor S_j if the Euclidean distance between them is less than the communication range of S_i. In this case S_i is called a *communication neighbor* of S_j. $N(S)$ represents the set of communication neighbors of S. The two sensors S_i and S_j can communicate directly with each other if and only if $S_i \in N(S_j) \bigwedge S_j \in N(S_i)$.

Definition 3 *A set of sensor nodes C is said to be a* cover set *of a sensor node S if every point on the perimeter of S sensing region belongs to the sensing region of at least one sensor in C.*

For example, in Figure 1, S_j, S_i, S_m, and S_k could be cover set members of node S and both of S_l and S_n are not.

Definition 4 *A sensor node is called a non-perimeter sensor node or inner node if it has a cover set.*

For example, in Figure 1, the perimeter of S_n sensing region is fully covered by node S, therefore node S_n is inner node.

Definition 5 Cover range *defined as the portion of the perimeter of the node sensing region covered by the sensing region of neighbor sensor nodes and is represented in terms of angle and identified by the closed interval [$StartAngle$, $EndAngle$]*

It is easy to see from Figure 1 that only the sensor nodes with distance less than $R+r$ are able to cover the perimeter of node S. The cover range of a sensor node is determined by its sensing range and the distance between the node and the center of the node S. In Figure 1, $2 * Theta$ is the covered angle, r is the sensing range of the node S_k and d is the node distance. The covered angle can then be calculated by using the equation

$$Theta = \arccos\{\frac{R^2 + d^2 - r^2}{2d * R}\} \quad (1)$$

and, the cover range of node S_k can then be calculated by using the equations

$$StartAngle = \arctan 2(\frac{Y - Coordinate(p_1)}{X - Coordinate(p_1)}) \quad (2)$$

$$EndAngle = \arctan 2(\frac{Y - Coordinate(p_2)}{X - Coordinate(p_2)}) \quad (3)$$

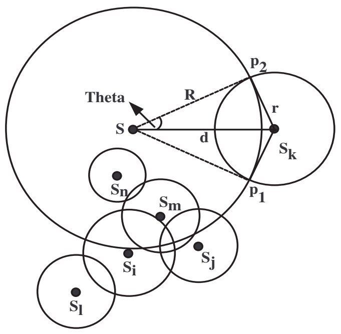

Figure 1: The relationship between the sensor node (S) sensing region's and its neighbors

4 Identifying coverage perimeter algorithm

In this section we outline our proposed algorithm to identify coverage perimeter nodes. First, we provide our algorithm assumption and then we provide a step by step description of the algorithm.

4.1 Algorithm Assumptions

Our approach relies on the following key assumptions regarding the sensor field and sensor nodes:

1. *Hardware capability:* We assume that each node is equipped with a sensing device (to sense the phenomenon), a communication device (to communicate with neighboring sensors), and a processing unit to process the data locally.

2. *Localization:* The position of each and every node is known in any arbitrary global coordinate system possibly by using a localization system from [2, 11]. For simplicity, we assume that every node knows its location in space in terms of an (x, y) coordinate. The neighbors of a particular node are determined based on its radio range.

3. *Stationary Nodes:* The sensor nodes are assumed to be static. The phenomenon being sensed can be dynamic.

4. *Underlying Communication Protocol:* We assume that there is an underlying protocol that takes care of all the necessary communication of information within the network.

5. All the sensor nodes are randomly deployed in the monitoring region.

6. We assume that the sensing range may be larger or smaller than the communication range (i.e., heterogeneous sensing ranges).

It is also assumed that each node knows its communication neighbors, including their identifications and coordinates. This information can be collected statically via one time beacon or hello message, or periodically if frequent changes in the topology is anticipated.

4.2 Algorithm description

To find out the perimeter nodes in the border area of the randomly deployed sensor network, we propose an algorithm has two phases: initial phase, and identification phase. In the beginning of our algorithm, each sensor node would collect its coverage neighbors, which is used in the second phase. Sensor nodes collect their coverage neighbors only once when they are deployed and enter the identification phase as receiving a new query. In identification phase, we identify the perimeter sensor nodes where each sensor node discovers itself if it lies along the outer edge of the perimeter or not.

4.2.1 Initial Phase

When sensor nodes have different sensing ranges and communication ranges, a sensor node may require more than one hop to communicate with its first hop coverage neighbors when its sensing range exceeds the communication range. Each sensor node exchanges its neighbors' information to collect the first hop coverage neighbors (CN). Each node has $CNList$ to record its CN neighbors' nodes information (Node information includes a node's id, sensing range and location). Each sensor node broadcasts its node information to neighbors. When node S_i receives node information from node S_j, node S_i will keep node S_j 's information if node S_j belongs to $CN(S_i)$. As the $CNList$ is updated, node S_i waits for a time period T. During the time period T, node S_i may receives new node information and updates its $CNList$. After the time period T is expired, node S_i broadcasts UPDATE message contains the new entries of the $CNList$ to its neighbors. After sending the update information, node S_i stops and resets its timer(Algorithm 1, Lines 3-10).

If node S_i receives the UPDATE message from node S_j, node S_i will check each node S_k in the received UPDATE message whether belongs to $CN(S_i)$ or not. If node S_k is belongs to $CN(S_i)$ and is not recorded in its list, node S_i will keep node S_k in the list and record node S_j as the next hop to node S_k (Algorithm 1, Lines 11-17). The exchanging of UPDATE messages will terminate when the list is not changed anymore and the operation of the initialization will finish in a finite steps since the network is static. The initialization phase to discover coverage neighbors of each node is shown in Algorithm 1.

4.2.2 Identification Phase

In this phase, based on coverage neighbors information each node will identify itself if it is coverage perimeter node or not. Each sensor node (S) in order to decides if it is perimeter node or inner node do the following:

- Node S computes the two intersection points between its sensing region and the sensing region of each node in its coverage neighbor nodes list $CNList$.

- Node S finds the cover ranges by calculating in terms of angle the portion of its sensing region circumference which is covered by each sensor node in $CNList$, considering the first intersection point in the calculations of start angle(StartAngle) and the other point for the end angle(EndAngle).

Algorithm 1 Initial Phase

{Each sensor node s_i executes this algorithm once it is deployed in a sensor network.}

```
1: Initially:Each sensor node $S_i$ exchanges its id, location, and sensing range with its neighbors.
   {$CNList$ : Coverage Neighbors List}
2: $CNList \leftarrow \Phi$
3: if node $S_i$ receives an node information message from its neighbor node $S_j$ AND $S_j \in CN(i)$ then
4:    $CNList \leftarrow CNList \bigcup S_k$ information's (i.e., $S_k$'s id, location, and sensing range).
5:    Waits for a time period T
6:    if the time period T is expired then
7:       Broadcast UPDATE message contains the new entries of the $CNList$ to neighbors
8:       Stop and reset the timer
9:    end if
10: end if
11: if node $S_i$ receives an UPDATE message from its neighbor node $S_j$ contains update part of the $S_j$ list then
12:    for each node $S_k$ in $S_j$ UPDATE message do
13:       if $S_k \notin N(S_i)$ AND $S_k \notin CNList$ AND $S_k \in CN(i)$ then
14:          $CNList \leftarrow CNList \bigcup s_k$ information's (i.e., $S_k$'s id, location, and sensing range) and record node $S_j$ as its next hop to node $S_k$.
15:       end if
16:    end for
17: end if
```

- Node S elects the senor node from its list, that has maximum covered angle as default node $S_{default}$ (the default node is used as initial node in order to find the cover set) then recalculate cover ranges for each sensor node in $CNList$ considering the line passing through itself and the default node as a reference axis.

- Node S executes algorithm 2 to find out its cover set. In algorithm 2, node S initially has a list of sensor nodes that overlapped with its sensing region circumference along with their cover ranges in terms of angle(Algorithm 2, Line 1). In order to find its cover set, node S tries to elect the cover range that covers as much its sensing region circumference that has yet been covered as possible (Algorithm 2, Lines 4-12).

- If the cover set founded node S decides to be an inner node; otherwise decides to be a perimeter node.

Algorithm 2 Find Cover Set ($C(S)$)

{Each node executes this algorithm in order to find its cover set $C(S)$.}

Output: True if the cover set $C(S)$ founded; otherwise False

```
1: Initially: node $S$ has a set $CNList$ of sensor nodes that overlapped with its sensing region circumference with their cover ranges calculated according to the default node $S_{default}$
   {StartAngle($S_i$)is the angle at which the first intersection point occurs between node $S_i$ sensing region circumference and the sensing region circumference of node $S$ and EndAngle($S_i$) is the angle of second intersection point}
2: $CurrentAngle \leftarrow EndAngle(S_{default})$
3: $C(S) \leftarrow S_{default}$
4: while $CurrentAngle < 360$ do
5:    Find node $S_i$ : $S_i \in CNList$ AND $StartAngle(S_i) < CurrentAngle$ AND $EndAngle(S_i)$ ends as far as possible.
6:    if $S_i$ founded then
7:       $C(S) \leftarrow C(S) \bigcup S_i$
8:       $CurrentAngle = EndAngle(S_i)$
9:    else
10:      Break
11:   end if
12: end while
13: if the last added node in $C(S)$ has end angle $\geq 360$ then
14:    Return True
15: else
16:    Return False
17: end if
```

5 Example Scenario

In this section, we provide a simple example with step by step explanation of our algorithm. Consider the simple wireless sensor network of Figure 2 where, the big circle indicates the sensing region for each node and the line join between them denotes that they can directly communicate with each other. Sensor nodes are uniquely identifiable and each node location specified as ordered pair of (x, y) coordinates. After neighbor discovery process ends, each node execute initial phase algorithm (Algorithm 1) to builds its coverage neighbors list ($CNList$).

For simplicity, consider nodes 2, and 8. Table 1 shows $CNList$ for nodes 2, and 8.

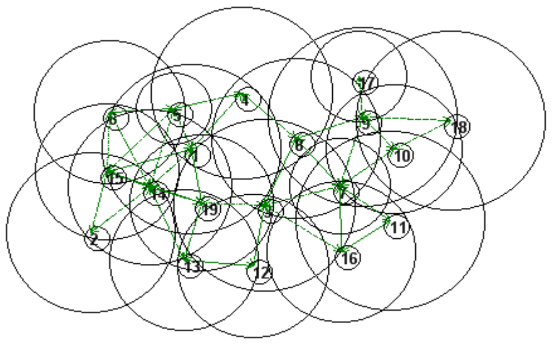

Figure 2: Wireless sensor network with variable sensing radii

Table 1: Coverage neighbors list for nodes 2 and 8

$NodeID$	$CNList$
2	1, 2, 13, 14, 15, 19
8	1, 3, 4, 7, 9, 10, 11, 12, 14, 16, 17, 18, 19

After formation of $CNList$, each node computes the two intersection points between its sensing region and the sensing region of each node in its coverage neighbor nodes list $CNList$. Then each node calculates the cover range for each sensor node in $CNList$, as shown in Tables 2, and 3 for nodes 2, and 8.

To this end, each node has list of sensor nodes that overlapped with the circumference of its sensing region along with their cover ranges in terms of angle and with the covered angle. Each node elects the senor node from its list, that has maximum covered angle or maximum covered angle as default node $S_{default}$, as shown from Tables 2, and 3.

Node 2 will elect node 14 as default node and node 8 will elect node 9 as default node. Then both of node 2 and 8 recalculate cover ranges for each sensor node in their $CNList$ considering the line passing through itself and the default node as a reference axis, as shown from Tables 4, and 5.

In order to identify perimeter sensor nodes, each node executes algorithm 2 to find out the its cover set. Node 2 start from its default node (node 14) and elect the next that has start angle less than 122.265° and end as far as possible.

As shown from Table 4, node 2 will elect node 13 where node 13 has the start angle of less than 122.27° and end angle greater than 122.27° then node 2 will add node 13 to its cover set (Algorithm 2, Lines 5-6). Since node 13 has end angle less 360°, node 2 try to elect the next node that has the start angle of less than 164.88° and end angle greater than 164.88°.

As shown from Table 4, node 2 will not find the node that has start angle of less than 164.88° and end angle greater than 164.88°, therefore the processes terminated and return false to indicate that there is no cover set for node 2. Since node 2 can not find cover set for itself it can decides to be perimeter node.

Also, node 8 will try to find its cover set, starts from its default node node 9. As shown from Table 5, node 8 try to elect the next node that has start angle of less than 148.78° and end angle greater than 148.78°.

Node 8 will elect node 3, where, node 3 has the start angle of less than 148.78° and end angle greater than 148.78°, then node 8 will add node 3 to its cover set. Since node 3 has end angle less 360°; node 8 try to elect

Table 2: Cover range and covered angle for node 2.

$NodeID$	$CoverRange$	$CoveredAngle$
1	[97.77°, 180°]	82.24°
6	[64.20°, 132.56°]	68.36°
13	[149.62°, 244.57°]	94.95°
14	[79.70°, 201.96°]	122.27°
15	[49.21°, 159.94°]	110.73°
19	[125.84°, 202.75°]	76.91°

Table 3: Cover range and covered angle for node 8.

$NodeID$	$CoverRange$	$CoveredAngle$
1	[298.47°, 48.65°]	110.18°
3	[225.60°, 2.56°]	136.96°
4	[331.53°, 111.96°]	140.44°
7	[192.16°, 265.73°]	73.57°
9	[86.58°, 235.36°]	148.78°
10	[144.16°, 229.21°]	85.05°
11	[174.88°, 270°]	95.11°
12	[261.38°, 313.78°]	52.40°
14	[319.87°, 1.71°]	41.85°
16	[215.84°, 282.99°]	67.16°
17	[102.16°, 165.34°]	63.19°
18	[142.28°, 203.88°]	61.60°
19	[286.54°, 1.71°]	75.18°

Table 4: Cover range and cover angle according to node 14 as default node for node 2.

$NodeID$	$CoverRange$	$CoveredAngle$
14	[0°, 122.27°]	122.27°
1	[18.07°, 100.31°]	82.24°
6	[344.51°, 412.87°]	68.36°
13	[69.93°, 164.88°]	94.95°
15	[329.52°, 440.25°]	110.73°
19	[46.14°, 123.06°]	76.91°

Table 5: Cover range and cover angle according to node 9 as default node for node 8.

$NodeID$	$CoverRange$	$CoveredAngle$
9	[0°, 148.78°]	148.78°
1	[211.89°, 322.07°]	110.18°
3	[139.02°, 275.98°]	136.96°
4	[244.94°, 385.38°]	140.44°
7	[105.57°, 179.15°]	73.58°
10	[57.58°, 142.63°]	85.05°
11	[88.30°, 183.42°]	95.12°
12	[174.80°, 227.20°]	52.40°
14	[233.28°, 275.13°]	41.85°
16	[129.25°, 196.41°]	67.16°
17	[15.57°, 78.76°]	63.19°
18	[55.69°, 117.29°]	61.60°
19	[199.95°, 275.13°]	75.18°

Table 6: The founded cover set for node 8 after executing algorithm 2.

$NodeID$	$CoverageRange$	$CoveredAngle$
9	[0°, 148.78°]	148.78°
3	[139.02°, 275.98°]	136.96°
4	[244.94°, 385.38°]	140.44°

the next node that has the start angle less than 275.98° and end angle greater than 275.98°. As shown from Table 5, node 8 will find node 4, where, node 4 has start angle of less than 275.98° and end angle greater than 275.98°. And since node 4 has end angle greater than 360°; therefore the processes terminated and return true to indicate that there is cover set for node 8. Whereas, node 8 can find the cover set for itself it can decides to be an inner node.

Table 6, shows the cover set for node 8. Each node performs what node 2 and 8 have done and each of them can decides to be an inner node or perimeter. Finally, nodes 4, 6, 9, 18, 11, 16, 12, 13, 2, and 15 decide to be perimeter nodes while nodes 5, 1, 14, 19, 3, 8, 7, 17, and 10 decide to be inner nodes.

6 Simulation Results

A simulator have been implemented in C#.NET language to evaluate the performance of the proposed perimeter nodes election algorithm. Wireless sensor nodes are randomly deployed in monitoring regions. The number of deployed sensor nodes varies from 500 to 3000 nodes with an increment of 500 nodes. The communication range of every sensor node is fixed at 40m, but the sensing range of each node varies from 20m to 60m. The metric for performance are as follows:

Number of perimeter nodes: the number of perimeter nodes selected to be close to the monitoring region.

Communication overhead: is measured by the total number of messages used in discovering perimeter sensor nodes.

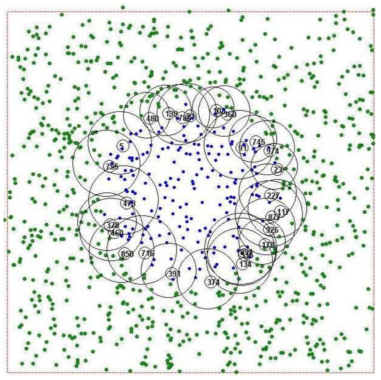

Figure 3: The final identified coverage perimeter using our approach for circular query region with sensor network of size 600 m × 600 m and with 1000 nodes are randomly deployed.

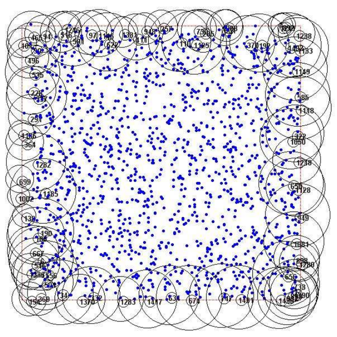

Figure 4: The final identified coverage perimeter using our approaches of sensor network of size 500 m × 500 m with 1500 nodes are randomly deployed.

Figures 3 and 4, show the identified coverage perimeter using our approaches. We randomly deployed the sensor nodes in monitoring region of size 600 m × 600 m. The number of deployed sensor nodes are 15000 nodes. In Figure 3, the nodes with black circle means the final identified coverage perimeter nodes and the dot circle represents the non-perimeter nodes. In Figure 4, we treat the whole sensor network region and execute our algorism; the nodes with black circle final identified coverage perimeter nodes. In Figure 4, we randomly deployed the sensor nodes in monitoring region

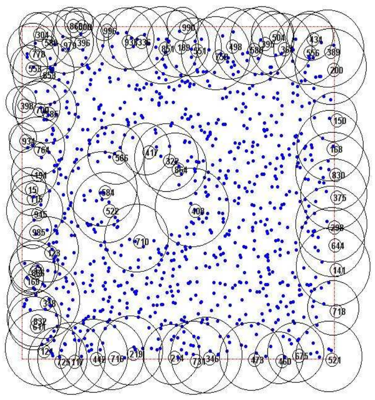

Figure 5: The final identified coverage perimeter using our approaches of sensor network of size 500 m × 500 m with 1000 nodes are randomly deployed and with one coverage hole.

of size 500 m × 500 m. The number of deployed sensor nodes are 1000 nodes with one coverage hole. We treat the whole sensor network region and execute our algorism; the nodes with black circle final identified coverage perimeter nodes for the whole network and for the coverage hole. Our simulation results show that the proposed algorithm can identify the coverage perimeter to form a loop that responsible to monitoring the perimeter of the interior area for query region and for the whole region, besides it can identify the coverage perimeter for coverage hole.

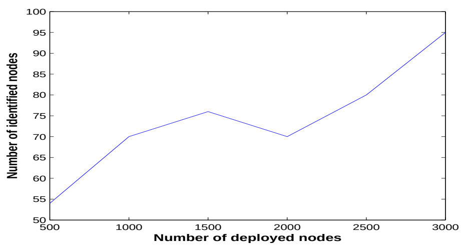

Figure 6: The number of identified coverage perimeter sensor nodes by our approach with Varying the number of deployed nodes.

Figure 6 shows the number of identified coverage perimeter sensor nodes by our approach. The monitoring region is assumed to be 500 m × 500 m with number of randomly deployed sensor nodes varies from 500 nodes to 3000 nodes with an interval of 500 nodes, and the communication range is fixed at 40 m, but the sensing range of each node varies from 20 m to 60 m. The simulation result demonstrates that when we increase the number of deployed nodes in the network, the number of identified coverage perimeter sensor nodes increases slightly.

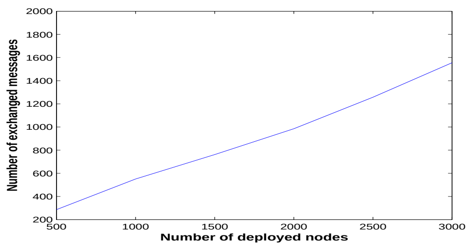

Figure 7: The number of messages overhead to identify coverage perimeter sensor nodes with Varying the number of deployed nodes.

In order to evaluate the communication overhead, we simulate different numbers of nodes from 500 to 3000 with an interval of 500 nodes. The sensor nodes are deployed in a region of size $500m \times 500m$ at random on a two dimensional plane and are uniformly distributed. The communication range of every sensor node is fixed at 40 m, but the sensing range of each node varies from 20 m to 60 m. In our algorithm, initially the coverage neighbors information of each node is collected then identification phase occurs locally at each node. After the sensor node identify itself as perimeter node it broadcast a declaration message to its neighbor to declare itself as perimeter node if it failed to find its cover set. According to the simulation results of Figure 7, the performance of messages overhead is mainly proportional to the number of sensor nodes and dose not increase dramatically when the number of sensor node increases. Thus, our approach consumes much less message over head.

7 Conclusion

In this paper, we have proposed a distributed algorithm for identify coverage perimeter sensor nodes in a wireless sensor network. We worked with sensor nodes that have heterogeneous sensing ranges. The algorithm employs coverage neighbors information to determine if the sensor node enclosed by neighboring nodes, and if it is located within the interior of the wireless sensor network. Our simulation results show that our algorithm successfully identify those sensor nodes that are on the perimeter of coverage (or coverage holes) in the region. Moreover, our algorithm successfully identify those sensor nodes that are on the perimeter of user defined region. In our proposed algorithm, only one exchange of messages in which the coverage neighbors are collected in initial phase. Thus, our algorithm requires minimal communication between nodes, since this is critical in wireless sensor networks.

References

[1] Andreas, S., Chih-Chieh, H., and B., S. M. Robust edge detection in wireless sensor networks. In *IEEE GLOBECOM 2008: Global Telecommunications Conference*, pages 1–5. IEEE, Nov. 30 – Dec. 4 2008.

[2] Chen, Y.-Q., Kim, Y.-K., and Yoo, S.-J. Accurate sensor position estimation for wireless sensor networks using neighborhood relationship. *IEICE Trans Commun*, E91-B(9):2907–2916, 2008.

[3] Chi, Z., Yanchao, Z., and Yuguang, F. Localized algorithms for coverage boundary detection in wireless sensor networks. *Wireless Network*, 15(1):3–20, 2009.

[4] Chintalapudi, K. and Govindan, R. Localized edge detection in sensor fields. *Ad Hoc Networks*, 1(2-3):273–291, 2003.

[5] Fekete, S. P., Kaufmann, M., Kröller, A., and Lehmann, K. A. A new approach for boundary recognition in geometric sensor networks. *CoRR*, abs/cs/0508006, Aug -01 2005.

[6] Fekete, S. P., Kröller, A., Pfisterer, D., Fischer, S., and Buschmann, C. Neighborhood-based topology recognition in sensor networks. In *ALGOSENSORS*, volume 3121 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 2004.

[7] Ghrist, R. and Muhammad, A. Coverage and hole-detection in sensor networks via homology. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, Piscataway, NJ, USA, 2005. IEEE Press.

[8] Krishnamachari, B. and Iyengar, S. S. Efficient and faulttolerant feature extraction in sensor networks. In *In Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks*, 2003.

[9] Liao, P.-K., Chang, M.-K., and Kuo., C. J. Distributed edge detection with composite hypothesis test in wireless sensor networks. In *In Proceedings of the Global Telecommunications Conference*, volume 1, pages 129–133. IEEE, 2004.

[10] Martincic, F. and Schwiebert, L. Distributed perimeter detection in wireless sensor networks, Jul- 02 2004.

[11] Moses, O. L., Krishnamurthy, D., and Patterson, R. A self-localization method for wireless sensor networks. *EURASIP Journal on Applied Signal Processing*, 4:348–358, 2003.

[12] Nowak, R. and Mitra, U. Boundary estimation in sensor networks: Theory and methods. In Zhao, F. and Guibas, L. J., editors, *IPSN*, volume 2634 of *Lecture Notes in Computer Science*, pages 80–95. Springer, 2003.

[13] Qing, F., Jie, G., and J., G. L. Locating and bypassing holes in sensor networks. *Mob. Netw. Appl.*, 11(2):187–200, 2006.

[14] Stefan, F. and Christian, K. Hole detection or: "how much geometry hides in connectivity?". In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 377–385, New York, NY, USA, 2006. ACM.

[15] Subhasri Duttagupta, K. R. and Ramanatha, P. Distributed boundary estimation using sensor networks. In *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 316–325, Oct. 2006.

[16] Wang, Y., Gao, J., and Mitchell, J. S. Boundary recognition in sensor networks by topological methods. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 122–133, New York, NY, USA, 2006. ACM.

[17] Weili, W., Xiuzhen, C., Min, D., Kai, X., Fang, L., and Ping, D. Localized outlying and boundary data detection in sensor networks. *IEEE Trans. on Knowl. and Data Eng.*, 19(8):1145–1157, 2007.

[18] Xing, G., Wang, X., Zhang, Y., Lu, C., Pless, R., and Gill, C. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Trans. Sen. Netw.*, 1(1):36–72, 2005.

[19] Zhang, C., Zhang, Y., and Fang, Y. Detecting coverage boundary nodes in wireless sensor networks. In *ICNSC '06. Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control*, pages 868–873. IEEE, 2006.