A Time and Energy Efficient Protocol for Locating Coverage Holes in WSNs

Phi Le Nguyen*, Khanh-Van Nguyen‡, Vu Quoc Huy‡, Yusheng Ji*†
*Department of Informatics, SOKENDAI (The Graduate University for Advanced Studies), Tokyo, Japan
†National Institute of Informatics, Tokyo, Japan
‡School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam
Email: *{nguyenle, kei}@nii.ac.jp, ‡vannk@soict.hust.edu.vn

***Abstract*—There are two main requirements in dealing with coverage holes in wireless sensor networks (WSNs): locating the hole boundary and finding the locations to deploy new sensors for hole patching. The current protocols on finding the patching locations always require re-running the protocols from scratch many times. This constraint causes the time complexity and energy overhead to increase proportionally to the hole size. In this paper, we propose a lightweight protocol to determine coverage holes in wireless sensor network. Our protocol does not only can determine the exact hole boundary but also approximate the boundary by a simpler shape which can help to speed up the patching location finding process. The simulation experiments show that our protocol can reduce more than 56% of time complexity and save more than 46% of energy overhead in comparison with existing protocols.**

***Index Terms*—Wireless sensor networks, coverage hole, hole locating, hole healing.**

I. Introduction

One of the most important issues in sensor networks is to assure a sufficient sensing coverage and connectivity, which reflects the quality of service of the network. Unfortunately, for reasons such as natural disruptions, adversarial attacks, or energy depletion, coverage holes unavoidably emerge. Therefore, these holes must be healed as soon as they appear. There are two processes required for healing the hole, that are: locating the hole boundary and determining the patching locations (i.e. the locations where the new sensors will be deployed).

In [2], [3], the authors designed algorithms for each sensor to detect whether it stays on the boundary of a coverage hole on the basis of localized Voronoi polygons. Li *et al.* [6] recently exploited the Delaunay triangulation to detect all coverage holes in the networks. Other researchers [4], [10], [7] proposed distributed protocols to determine the sensors on the coverage holes on the basis of the mathematical analysis of the arc or angle created by the overlapped sensing area of two neighboring sensors.

Although many protocols have been proposed to identify the hole boundary, protocols for determining the patching locations have only rarely been studied. Aliouane *et al.* [1] and Xiong *et al.* [11] proposed protocols for detecting the hole boundary and patching locations on the basis of *boundary critical points*. Yao *el al.* [9] developed a patching location determining algorithm on the basis of the concept of a perpendicular bisector line. Although these protocols can find out the patching locations, they suffer from a large energy overhead and time complexity. The reason is they require re-running the protocols from scratch many times. This process consumes a lot of energy of the boundary sensors. Moreover, the complexity of these algorithms is proportional to the size of the hole. That is, the bigger the hole, the more complicated the algorithm and the more time and energy it consumes. In addition, re-running the *hole boundary determining* protocol also causes the boundary sensors to exhaust energy more quickly than the other sensors and thus may consequently result in the hole enlargement problem.

The main reason that makes patching location determination becomes a hard problem is due to the complication of the hole boundary. Therefore, in this paper we propose a hole locating protocol which can not only identify exact hole boundary but also approximate the boundary by a simpler shape which can help to speedup the patching location determining process. We note that Kershner [5] proved that the triangular tessellation achieves full coverage with an asymptotic minimum number of sensors. In this tessellation, each sensor is located at the center of a regular hexagon that has six neighbors at a distance of $R_s\sqrt{3}$, where R_s is the sensing range. Motivated by this result, our idea is to approximate the hole by a simpler polygon covering the hole, whose vertices and edges are the nodes and edges of a given regular triangle grid.

The remainder of the paper is organized as follows. We describe the network model and give definitions in section II. In section III, we present a protocol to determine the coverage hole and approximate the hole by a simpler polygon. Section IV presents our experiments to evaluate the performance of the protocols. We conclude the paper in section V.

II. Network model and definitions

A. Network model

We assume that the outermost boundary of the sensor network is known and a set of sensors (denoted by $S = \{S_1, S_2, ..., S_m\}$) is deployed randomly inside the boundary. We assume that all sensors have the same sensing range, R_s, and the same transmission range, R_t. We use a widely accepted assumption that $R_t \geq 2R_s$. A point p is said to be in sensing range of a sensor S_i if their Euclidean distance is less than the sensing range. We also assume that each sensor knows its own position and that of its 1-hop neighbors.

DOI 10.1109/LCN.2016.37

B. Definitions

$\overline{AB}$ denotes the Euclidean between A and B. $\widehat{AB}$ denotes a circular arc from A to B in clockwise order (when seeing from the center of the circle). Given a sensor S_i, the *sensing circle* of S_i (denoted by $SC(S_i)$) is defined as the circle with the center of S_i and radius of R_s. The sensing area of S_i (denoted by $SA(S_i)$) is defined as the area inside the sensing circle of S_i, i.e. $SA\left(S_i\right) = \left\{p|\overline{pS_i} < R_s\right\}$. A *coverage hole* is defined as a region bounded by a non-self-intersecting curve and consists of points that do not belong to the sensing area of any sensors. The boundary of a coverage hole H, ∂H, is defined by the set of points p of H such that every neighborhood of p contains at least one point of H and at least one point not of H. Also, any point on the hole boundary is called a *boundary point*. In this paper, we assume that the network does not contain coverage holes whose boundary is discontinuous and thus a coverage hole boundary is a Jordan curve, i.e. a simple closed curve. The *sensing neighbors* of a sensor S_i are defined as the sensors whose distance to S_i is less than or equal to twice the sensing range, R_s. Let S_{i_j} be a sensing neighbor of S_i, and I_j^1, I_j^2 denote the two intersection points of $SC(S_{i_j})$ with $SC(S_i)$ that stay on the left and right sides of ray $\overrightarrow{S_iS_{i_j}}$, respectively, then the *intersection arc* of $SC(S_{i_j})$ with $SC(S_i)$ is defined as the arc $\widehat{I_j^1 I_j^2}$ that belongs to $SC(S_i)$. Also, I_j^1 and I_j^2 are called the left and right intersection points of $SC(S_{i_j})$ with $SC(S_i)$, respectively. Throughout this paper, we use notation $N(S_i) = \{S_{i_1}, ..., S_{i_k}\}$ to denote the set of all sensing neighbors of S_i and $\widehat{I_j^1 I_j^2}$ to denote the intersection arc of $SC(S_{i_j})$ with $SC(S_i)$ ($\forall j = \overline{1,k}$). S_{i_j} is called a *non-redundant sensing neighbor* of S_i if $\widehat{I_j^1 I_j^2}$ is not covered by any intersection arc of other sensor's sensing circles to $SC(S_i)$. Also, a sensing neighbor that is not non-redundant is called a *redundant sensing neighbor*. Let S_{i_u}, S_{i_v} be two sensing neighbors of S_i, then S_{i_v} is called the *right adjacent neighbor* of S_{i_u} (and S_{i_u} is called the left adjacent neighbor of S_{i_v}, vice versa) if there is no other non-redundant sensing neighbor S_{i_w} of S_i such that I_w^1 stays between I_i^1 and I_j^1, or I_w^1 coincides with I_i^1 (or I_j^1) and I_w^2 stays between I_i^2 and I_j^2. (S_{i_u}, S_{i_v}) is called a *disjoint adjacent sensing neighbor pair* of S_i if S_{i_v} is the right adjacent neighbor of S_{i_u} and $\widehat{I_u^2 I_v^1}$ overlaps with neither $\widehat{I_u^1 I_u^2}$ nor $\widehat{I_v^1 I_v^2}$. An arc (in clockwise order) of a sensing circle is called a *non-covered arc* if it is not covered by the sensing area of any sensor and its two ending points are intersection points of sensing circles. Also, a sensor is called a *non-covered sensor* if its sensing circle contains at least one non-covered.

III. Proposed protocol

A. Protocol overview

Our goals are (1) identifying the boundaries of the coverage holes and (2) approximate that holes by simpler polygons. For the first one, we do it in two steps. First, we check whether coverage holes exist by searching for non-covered sensors. If non-covered sensors are detected, we use them to find the boundaries of the holes. For the second one, the *approximate polygons* or *A-polygons* for short can be obtained by the

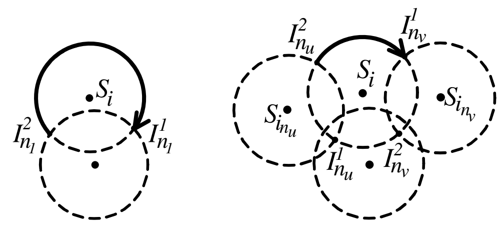

Fig. 1: Illustration of Property 2.

specific edges of unit triangles that intersect the boundary of the hole, where the specific edges are not inside the hole. To satisfy the definition of coverage hole, the unit triangle's edge length, d, is set approximately equal to the sensing range, R_s (i.e. $d = R_s - \gamma$, where γ is a tiny positive number). The process of our protocol can be summarized as follows.

- Each sensor periodically broadcasts *neighbor notification messages* that contain its information such as its ID and coordinates. Because $R_t \geq 2R_s$, all sensors can obtain information of their sensing neighbors using these neighbor notification messages.
- Each sensor identifies whether it is a non-covered sensor by using the *non-covered sensor detecting protocol*.
- Each non-covered sensor uses *hole boundary determining protocol* to locate the boundaries of the corresponding coverage holes. This protocol is conducted by having special packets traveling around the non-covered sensors to collect information of the hole boundaries. This protocol also performs an algorithm to approximate the holes by simpler polygons using the regular triangle grid.

B. Non-covered sensor detecting protocol

Before going further, we make some important observations about coverage hole.

Property 1. *Suppose* $\left\{\widehat{I_1I_2}, \widehat{I_2I_3}, ..., \widehat{I_{k-1}I_k}, \widehat{I_kI_1}\right\}$ *is a set of non-covered arcs and* δ *is the line connecting these arcs, i.e.* $\delta = \bigcup_{i=1}^{k} \widehat{I_iI_{i+1}}$ *(*$I_{k+1} \equiv I_1$*). Then the area enclosed by* δ *and staying on the left side of* δ *is a coverage hole.*

Property 2. *Let* S_i *be a sensor and* $\left\{S_{i_{n_1}}, ..., S_{i_{n_l}}\right\}$ *be the set of all non-redundant sensing neighbors of* S_i*. Then:*

- *If* $l = 1$*, then* $\widehat{I_{n_1}^2 I_{n_1}^1}$ *is a non-covered arc (Fig. 1(a)).*
- *If* $l > 1$ *and* $(S_{i_{n_u}}, S_{i_{n_v}})$ *is a disjoint adjacent sensing neighbor pair of* S_i*, then* $\widehat{I_{n_u}^2 I_{n_v}^1}$ *is a non-covered arc (Fig. 1(b)).*

On the basis of these properties, we propose a *non-covered sensor detecting protocol* as described below. The protocol is divided into two steps. In the first step, each sensor constructs the list of its non-redundant sensing neighbors by using the information obtained from the periodically broadcast neighbor notification messages. Each sensor then sorts this list in the order such that any item of the sorted list will be the right adjacent neighbor of its previous item. In the second step, each sensor checks the sorted non-redundant sensing neighbor

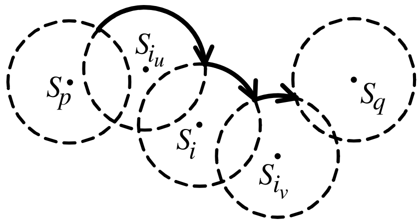

Fig. 2: Illustration of Property 3.

list to verify whether it has only one non-redundant sensing neighbor or at least one disjoint adjacent sensing neighbor pair. If the answer is "yes", the sensor is non-covered. Otherwise, it is not non-covered.

C. Hole boundary determining protocol

In the following, we describe a distributed protocol to locate the boundary of a coverage hole and approximate the boundary by a simpler polygon, i.e. an *A-polygon*. Let us start by informally discussing the main idea of this protocol, which is based on an important observation we already mentioned in Property 1: the hole boundary can be seen as the union of a collection of non-covered arcs, which are created by a collection of non-covered sensors. Let us call them *boundary sensors* for this context. Thus, a special message, the *hole boundary detection* (or *HBD* for short) message, can be formed to travel around these boundary sensors. This special message can be created by a non-covered sensor upon being detected (as in III.B); multiples of them can be created, but we will eliminate the late-coming, redundant ones. To fulfill our protocol, we also need another important observation to decide how a boundary sensor can find the next boundary sensor (in the counterclockwise direction). This final point is formally presented by the following property.

Property 3. *Let (S_{i_u}, S_{i_v}) be a disjoint adjacent sensing neighbor pair of a non-covered sensor S_i. Let S_q be the non-redundant sensing neighbor of S_{i_v} such that S_q is right adjacent to S_i. Then, (S_i, S_q) is a disjoint adjacent sensing neighbor pair of S_{i_v}. Similarly, Let S_p be the non-redundant sensing neighbor of S_{i_u} such that S_p is left adjacent to S_i. Then, (S_p, S_i) is a disjoint adjacent sensing neighbor pair of S_{i_v} (Fig. 2).*

This property implies that if S_i is a boundary sensor and (S_{i_u}, S_{i_v}) is its disjoint adjacent sensing neighbor pair, then S_{i_u} and S_{i_v} are also boundary sensors, and furthermore, (S_{i_u}, S_i, S_{i_v}) are consecutive boundary sensors. On the basis of this property, the next boundary sensor of a boundary sensor S_i (i.e. S_{i_v}), is a non-redundant sensing neighbor of S_i and right adjacent to the previous boundary sensor of S_i (i.e. S_{i_u}). This move from S_i to S_{i_v} is the main move of our algorithm: we call it the *Bound_Move*.

Below we restate our *hole boundary detecting protocol* in full detail. Each sensor detects if it is a boundary sensor as in section III-B. Assume S_* is detected as a boundary sensor. A non-covered arc and the corresponding disjoint adjacent sensing neighbor pair can be found at S_*, which are to be used as the starting point of the process of creating a *HBD* message and forwarding it in the counterclockwise direction to collect information about the other boundary sensors. At each intermediate boundary sensor where the *HBD* message arrives, the current boundary sensor determines the *current non-covered arc* (i.e. the non-covered arc of the hole boundary belonging to the sensing circle of the current sensor), then approximates it by a few unit triangle edges, and updates this *A-polygon* information to the *HBD* message. The *BHD* message then is forwarded to the next boundary sensor. This process terminates when the *HBD* message comes back to the creator, S_*.

The boundary of the *A-polygon* is described by the unit triangles that intersect the hole boundary. We call these triangles *I-triangles*. To minimize the size of the *HBD* message, we do not store the coordinates of the *I-triangles* but store only the *relative position* of each *I-triangle*. This relative position can be as short as a two-bit string that is defined as 10, 01, 11, or 00 if the current *I-triangle* stays on the right of, left of, above, or below the previous *I-triangle*, respectively. Thus, below are the steps each boundary sensor needs to follow when it receives the *HBD* message and becomes the current sensor in this algorithmic context.

- **Step 1 - Check for termination:** Check if the current sensor is the *HBD*'s creator. If it is, the *hole boundary determining* protocol terminates, and the *patching location determining* protocol starts.
- **Step 2 - Check for redundant *HBD*:** All the boundary sensors automatically create a HBD message, so this redundancy can be lessened by allowing only the one with the highest creator ID and dropping the others.
- **Step 3 - Approximation of the current non-covered arc**: Execute the *Bound_Move* to determine the next boundary sensor. Having determined next boundary sensor, the *current non-covered arc* can be determined straightforwardly (by using Property 2). The current sensor determines the *I-triangles* that intersect the *current non-covered arc* and inserts the relative positions of these *I-triangles* to the *HBD* message.
- **Step 4 - Moving to the next boundary sensor**: Forward the *HBD* message to the next boundary sensor.

IV. Performance Evaluation

In this section, we compare our protocol with the existing protocols: HACH [1], BCP [11] and HPA [9]. For the sake of simplicity, we call our proposed protocol TELC (which stands for Time and Energy efficient protocol for Locating Coverage holes). To see the impact of various shapes of the hole, we placed holes with different shapes as shown in Fig. 3. We run the experiments on NS-2 simulator and chose IEEE 802.11 (legacy mode) as our MAC protocol. The nodes follow the energy model suggested by Shnayder *et al.* [8] (see Table I). The experiments are deployed on a computer with the CPU of *Xeon E5-2620 v2 2.10GHz x 4*, RAM of *10GB* and the OS of *Ubuntu 14.04 64-bit*. The results are the average of 10 run times (as the error is very small , it is not shown here).

A. Time complexity

Fig. 4 shows that TELC strongly outperforms the others strongly in term of time complexity. The time complexities

Factor	*Value*
Transmission range	40m
Sensing range	20m
Idle power	9.6mW
Receiving power	45mW
Transmitting power	88.5mW

TABLE I: Factor setting of simulated sensors.

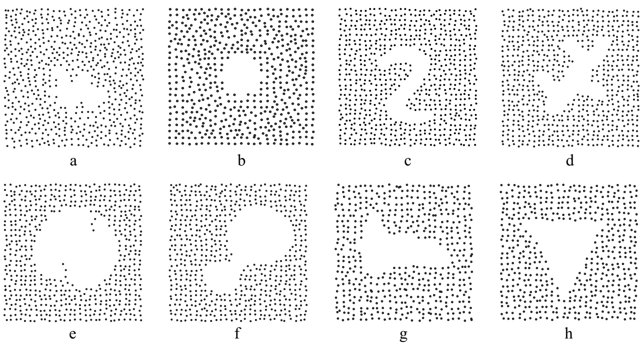

Fig. 3: Captures of simulated areas.

of HACH, BCP and HPA tend to increase very fast when the hole size increases, while the time complexity of TELC is quite stable. Furthermore, even with small holes, the time complexities of the other protocols are more than twice of TELC's time complexity, e.g. the ratios of the time complexities of HACH, BCP and HPA to TELC are $2.3, 11.2, 28.3$, respectively, for the hole with 15 boundary sensors.

B. Energy overhead

The energy overhead is the average energy consumed by one sensor. Similarly to the time complexity, the energy overhead caused by TELC is much less than that caused by the others as shown in Fig. 5. The energy overheads of all protocols tend to increase with the increase of the hole size but TELC increases slowest. When the number of boundary sensors is 15, HACH, BCP and HPA cause $1.8, 4.0$ and 10.7 times more energy overhead than TELC. This ratio even increases to $7.5, 298.6$ and 738.7 when the number of boundary sensors increases to 88. Similarly to the time complexity, the energy overhead of HPA is larger than that of HACH and BCP because HPA requires more patching locations than HACH and BCP. HACH causes a smaller overhead than BCP because it requires re-running the protocol less times than BCP.

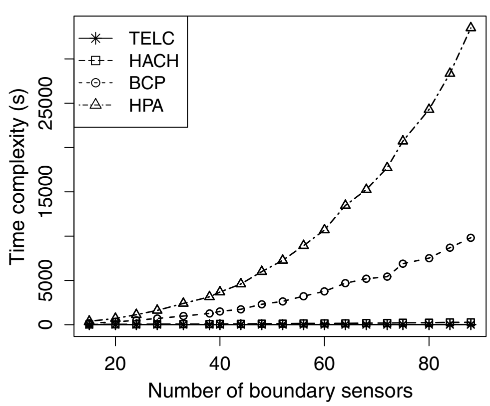

Fig. 4: Evaluation of time complexity.

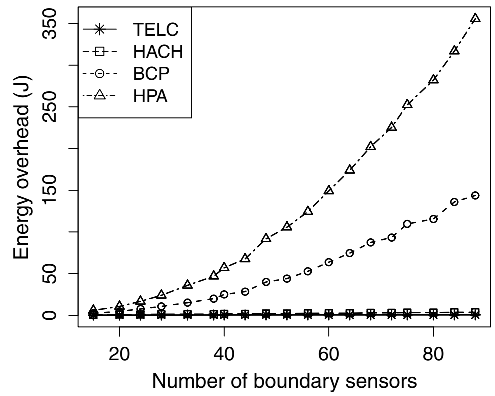

Fig. 5: Evaluation of energy overhead.

V. Conclusion and Future Work

In this paper, we have proposed protocol to locate the boundary of coverage holes. This protocol also approximates the holes by simpler polygons using a given regular triangle grid. The simulation results show that our protocol strongly outperforms the existing protocols in terms of time and energy consumption.

References

[1] L. Aliouane and M. Benchaïba, "HACH: healing algorithm of coverage hole in a wireless sensor network," in *Proc. of NGMAST*, 2014, pp. 215–220.

[2] Y. F. C. Zhang, Y. Zhang, "Localized algorithms for coverage boundary detection in wireless sensor networks," *Wireless Networks*, vol. 15, no. 1, pp. 3–20, 2009.

[3] Y. Z. C. Zhang and Y. Fang, "A coverage inference protocol for wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 6, pp. 850–864, jun 2010.

[4] Y. C. Hwa-Chun Ma, Prasan Kumar Sahoo, "Computational geometry based distributed coverage hole detection protocol for the wireless sensor networks," *J. Network and Computer Applications*, vol. 34, no. 5, pp. 1743–1756, 2011.

[5] R. Kershner, "The number of circles covering a set," *American Journal of Mathematics*, vol. 61, no. 3, pp. 665–671, Jul 1939.

[6] W. Li and W. Zhang, "Coverage hole and boundary nodes detection in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 48, pp. 35 – 43, 2015.

[7] S. C. M. Watfa, "Energy-efficient approaches to coverage holes detection in wireless sensor networks," in *ICEICE 11*, 2006, pp. 131–136.

[8] V. Shnayder, M. Hempstead, B. rong Chen, G. Werner-Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications." in *SenSys*, 2004, pp. 188–200.

[9] J. Yao, G. Zhang, J. Kanno, and R. Selmic, "Decentralized detection and patching of coverage holes in wireless sensor networks," in *SPIE Defense, Security, and Sensing*, vol. 73520, 2009.

[10] E. D. Zhao and Y. T. L. J. Yao, H. Wang, "A coverage hole detection method and improvement scheme in wsns," in *ICEICE 11*, 2011, pp. 985–988.

[11] Q. X. Zhiping Kang, Honglin Yu, "Detection and recovery of coverage holes in wireless sensor networks," *Journal of Networks*, vol. 8, no. 4, pp. 822–828, Apr 2013.