DOI : 10.3745/JIPS.2009.5.3.145

Journal of Information Processing Systems, Vol.5, No.3, September 2009 **145**

Topological Boundary Detection in Wireless Sensor Networks

Thanh Le Dinh*

Abstract: The awareness of boundaries in wireless sensor networks has many benefits. The identification of boundaries is especially challenging since typical wireless sensor networks consist of low-capability nodes that are unaware of their geographic location.
In this paper, we propose a simple, efficient algorithm to detect nodes that are near the boundary of the sensor field as well as near the boundaries of holes. Our algorithm relies purely on the connectivity information of the underlying communication graph and does not require any information on the location of nodes. We introduce the 2-neighbor graph concept, and then make use of it to identify nodes near boundaries. The results of our experiment show that our algorithm carries out the task of topological boundary detection correctly and efficiently.

Keywords: *Wireless sensor network, Hole, Boundary detection, 2-neighbor graph*

1. Introduction

The task of boundary detection in wireless sensor networks is stated as follows: Given a wireless sensor network deployed in an area called the sensor field, each node must ascertain whether it is located near the boundary of the sensor field as well as the boundaries of holes.

In this paper, we focus on boundary detection in wireless sensor networks without information on the location of nodes. The proposed solutions will rely purely on topological features, i.e. the connectivity information of the underlying communication graph. We emphasize the topological (topology-based) methods for the following reasons. First, it would be costly to equip each node with a positioning device such as a GPS unit to obtain location information at the nodes. With thousands of nodes deployed, we would have to spend a lot of money on positioning devices. In order to reduce the cost, we may equip only a few nodes, called anchors, with positioning devices and apply a localization algorithm to infer the locations of non-anchor nodes [11]. Unfortunately, to date no localization algorithm that can give derived locations that reflect the true locations of nodes has been developed. Second, positioning devices consume a lot of the energy of the nodes, which cannot be recharged, thereby reducing the lifetime of the nodes. In addition, we cannot always obtain exact location information since positioning devices cannot work entirely free from error. Thus, the requirement of location information available at the nodes will lead to expensive and short-lived sensors networks.

Boundary detection has many applications. Hole formation is often caused by extreme events such as fire, earthquake, inundation, and so forth. As such, the identification of holes is very useful in wireless sensor network applications that monitor such events. For some sensor network applications such as data-centric storage, which does not require the true locations of sensor nodes, invented (virtual) locations can be used instead. Several methods for computing virtual locations have been proposed [11]. But, as already examined in [7], the resultant virtual coordinates are distorted in comparison to the true geometry of the communication graph. The authors of [7] showed that boundary awareness can be used to build less distorted virtual coordinates. In addition, boundary information is helpful to both topology-based [2,7-9] and location-based routing [4]. From our viewpoint, we may use boundary information to build a routing protocol that can avoid holes and produce optimal paths. This will be a part of our future research.

Up to now, and recently, only a few topological boundary detection algorithms have been proposed [5-7]. These algorithms are not competitors with our approach, with the exception of the one introduced in [7], since they seem feasible only for uniform and very high density node distributions. The algorithm in [7] uses beacon and isolevel concepts to identify nodes near boundaries. The issues posed in [7] concern beacon selections. As far as we know, beacon selection is as complex as leader election [13]. With four global beacons and many local beacons, the time required to select beacons incurs a considerable cost. In addition, it floods the network several times. This contributes to the convergence time remarkably.

Our contributions: In this paper, we propose a simple, efficient algorithm for boundary detection. Our algorithm relies purely on the connectivity information of the underlying communication graph and does not require any information

Manuscript received March 9, 2009; revised May 18, 2009; accepted June 18, 2009.
Corresponding Author: Thanh Le Dinh
* College of Technology, Vietnam National University, Hanoi, Vietnam (thanhld.hdu@gmail.com)

on the location of nodes. We introduce the 2-neighbor graph concept, and then make use of it to identify nodes near boundaries. Computations are processed locally. Nodes need to communicate only with their 1-neighbor and 2-neighbor nodes. Each node can identify whether it is located near boundaries as soon as it knows all its 2-neighbor nodes.

The rest of this paper is organized as follows. Section 2 presents our boundary-locating algorithm. Section 3 shows how well our algorithm can deal with network dynamics. Section 4 examines the performance of our algorithm by simulation. Section 5 provides a brief comparison of our algorithm with previous algorithms. Finally, some final remarks on the proposed algorithm and our future works are presented in Section 6.

2. Topological Boundary Finding

2.1 Intuition and Heuristic

Consider a region $R \subseteq R^2$ with some holes in it. For each point $p \in R$, consider the circle $c(p, r)$, called p's circle, that centers at p and has radius r, where r is a real constant. If p is near boundaries, i.e. there are points in boundaries where the Euclidean distances from them to p are less than r, then c is cut into *solid* and *dash* arcs, which are interlaid. Solid and dash arcs are our concepts: solid arcs are arcs that contain points not in holes; dash arcs are arcs that contain points in holes or points outside the sensor field. This is shown in Fig. 1.

Mimicking the continuous case, in wireless sensor networks, for each node p, consider the graph formed by nodes that are one-hop away from p and the connectivity between these nodes. We call this graph p's *2-neighbor graph (2NG)*. Intuitively, if p is not located near boundaries, then p's 2NG forms a "ring" (i.e. it has a ring-like shape); otherwise p's 2NG consists of one or more segments of a "broken ring". This is illustrated in Fig. 2.

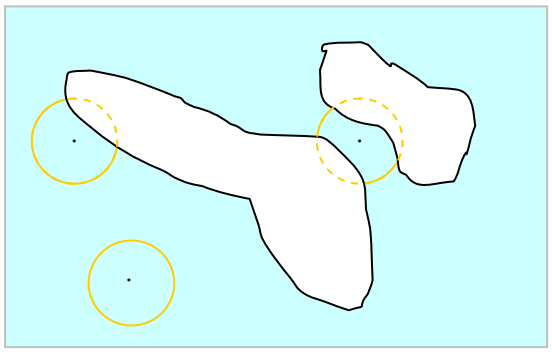


Fig. 1. A region with two holes and three examined points. The circles centering on points near to holes are cut into solid and dash arcs

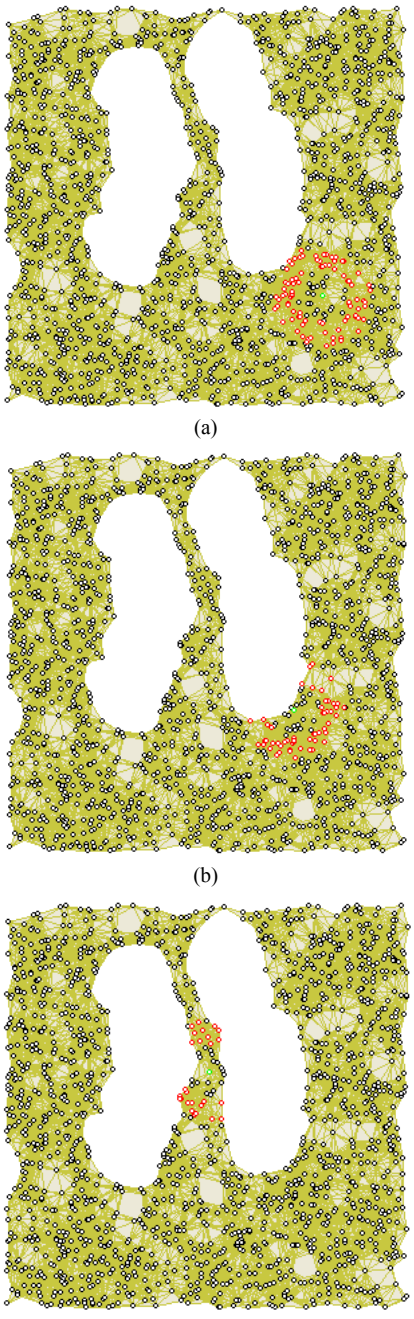


Fig. 2. A node with its 2NG. The edges represent connectivity between nodes. The red nodes are nodes in the 2NG of the green node. (a) Nodes not near boundaries; (b) Nodes near only one boundary; (c) Nodes near two boundaries

Based on this intuition, we have the following heuristic that mimics the continuous case: *p is near boundaries if its 2-neighbor graph does not form a ring*.

2.2 Algorithm

The heuristic given above leads to a simple algorithm to detect nodes near boundaries, as shown in Fig. 3.

Each node *p*:

- Discovers all its neighbors in order to build a list of neighbors.
- Sends the list of neighbors to all nodes that are one-hop away from *p*.
- If all the lists of neighbors of the nodes that are one-hop away from *p* have been received, it:
  - Constructs *p*'s 2-neighbor graph *G2* based on the lists of neighbors received
  - Examines if *G2* forms a "ring" by calling ***IsRing*** (*G2*). If *G2* is a ring, i.e. ***IsRing***(*G2*) returns true, then it sets *p.nearBoundaries = FALSE* (*p* is not near any boundary), or it sets *p.nearBoundaries = TRUE* (*p* is near boundaries).

IsRing(*G2*)

- Determines whether *G2* is connected or not by simply applying the coloring algorithm: selects an arbitrary node *k* in *G2*, colors *k* and all nodes in *G2* that are connected to *k*. If there are still uncolored nodes in *G2* then *G2* is not connected, otherwise *G2* is connected.
- If *G2* is not connected, i.e. consisting of several connected components, then returns false.
- Or, *G2* is a connected component,
  - Selects an arbitrary node *t* in *G2* then divides *G2* into sub-graphs g_0, g_1, g_2,… where g_i is formed by nodes at the distance *i* from *t* in *G2* (suppose each edge has the weight of 1) and the connectivity between them. g_0 is a connected graph while g_i, i > 0 may consist of one or two connected components.
  - Examines if g_2 consists of two connected components by applying the coloring algorithm to g_2, if so
    - Let *G'2* be the graph resulting from *G2* by the removal of nodes in g_0 and g_1 and the related links
    - Determines if *G'2* is not connected by applying the coloring algorithm to *G'2*, if so then returns false.
    - Or, *G2* is a "ring", returns true.
  - Or, g_2 is a connected component or contains no nodes, returns false.

Fig. 3. Our Boundary-Finding Algorithm

In our algorithm, *G2* is stored locally and *IsRing*(*G2*) is executed locally as well. After two rounds, each node *p* knows the topology of the local sub-graph that contains *p* and all nodes near (0- or 1-hop away from) *p*. *G2* is extracted from this sub-graph. The function *IsRing* (*G2*) is then called to examine whether *G2* is a "ring". If *G2* is a ring, then *p* is not near any boundary, so *p* sets its *nearBoundaries* variable as FALSE; or *p* is close to boundaries, so *p* sets its *nearBoundaries* variable as TRUE. Because *G2* is constructed and *IsRing*(*G2*) is executed locally, every node in the network will complete the task after two (asynchronous) rounds.

To examine whether the 2-neighbor graph *G2* forms a ring, we first examine if it is a connected graph by simply applying the coloring algorithm: select an arbitrary node *k* in *G2*, color *k* and all nodes in *G2* that are connected to *k*. If there are still uncolored nodes in *G2,* then *G2* is not connected; otherwise, *G2* is connected. Obviously, if *G2* is not a connected graph then it cannot be a ring. In the case where *G2* is a connected graph, *G2* may be a ring or just a segment of a broken ring. To determine whether *G2* is or is not a ring, we "cut" a segment of *G2* that contains node *t*, *t*'s neighbors and *t*'s neighbors' neighbors, and then examine the remaining segment(s). If we cannot cut a large enough segment (g_2 is a connected graph or contains no nodes), then *G2* is actually not a ring. Otherwise, if we have two remaining segments or have one exact remaining segment that does not "fit" with the removed segment (*G'2* is not connected), then *G2* is actually not a ring. *G2* is a ring only when we have one remaining segment that fits with the removed segment. The intuition of cutting one segment of *G2* and examining the remaining segment(s) is shown in Fig. 4.

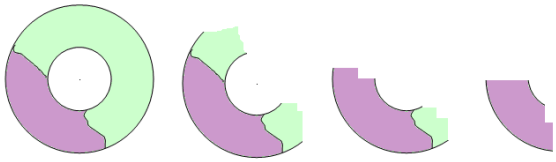


Fig. 4. A ring or a segment of a broken ring after cutting one segment. The removed segment is violet. The remaining segments are green. (a) The remaining segment fits with the removed segment. (b) The two remaining segments. (c), (d) A sufficiently large segment cannot be cut

3. Dealing with Network Dynamics

Network dynamics is caused by node failure and death, or, sometimes, by new node deployment or node mobility. To deal with network dynamics, we can modify our algorithm so that each node *p* resends its list of neighbors if there are sufficient changes in the list, and reconstructs and reexamines its 2NG whenever it receives new lists of neighbors. In this way, our algorithm can respond quickly to topological changes.

4. Experimental Evaluation

In order to evaluate the performance of our algorithm,

we built a simple simulator which generates random node distributions and provides a set of tools for us to view the network, to create holes, and to observe nodes near boundaries. Our algorithm has been evaluated on various network instances.

To evaluate the effect of node density on our boundary detection algorithm, we performed experiments in which the node density and communication range were varied. Our observation was that for node distributions where the average degree in the communication graph was around 7 or above, our algorithm seems to perform reasonably well. For comparison, the algorithm in [7] produces reasonable results only for graph densities with an average degree of around 18 or more.

Also, in order to evaluate the robustness of our algorithm, we performed experiments involving both convex and concave holes which are close to each other. Our observation was that our algorithm performs well in the presence of arbitrary holes.

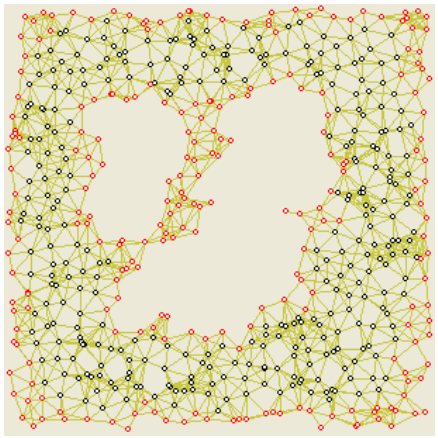
Some experimental results of our algorithm are given in Fig. 5.

5. Comparison with Previous Algorithms

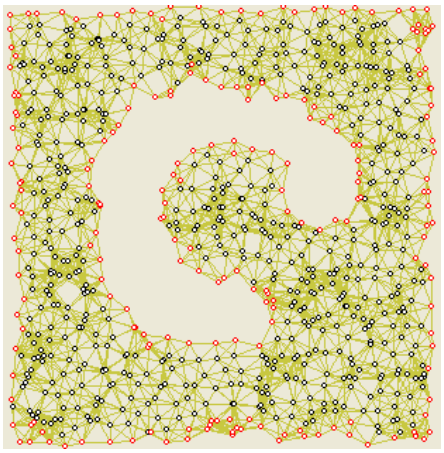
As mentioned in Section 1, to the best of our knowledge, up to now only the algorithms in [5-7] belong to the topological class. Those in [5,6] are applicable only to uniform and dense wireless sensor networks, while that in [7] is not efficient since it has to solve two complex sub-problems, namely that of beacon selection or leader election, and the flooding problem [13]. Another disadvantage of algorithm [7] is that it does not deal well with network dynamics. Also, the experimental results show that the algorithm in [7] does not perform well in network densities with an average degree of less than 18, while our algorithm does. Thus, we believe that our algorithm is the first efficient algorithm for topological boundary detection.

6. Final Remarks

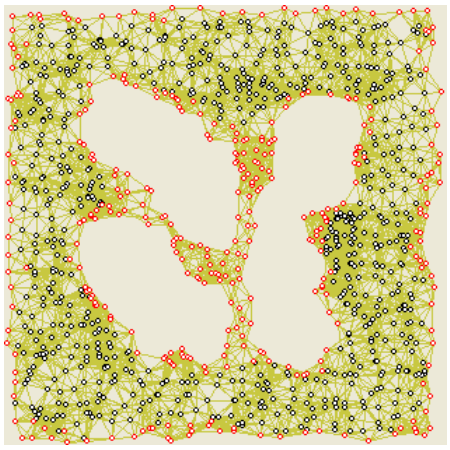
Nodes near boundaries can be classified into two sub-classes named *SB* and *MB*. The former consists of nodes



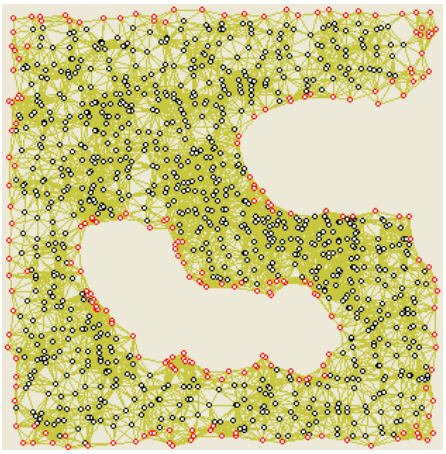
(a) Network composed of 559 nodes with two holes, average degree 7.



(b) Network composed of 836 nodes with one hole, average degree 10.



(c) Network composed of 1203 nodes with three holes, average degree 13.



(d) Network composed of 1697 nodes with one hole, average degree 17.

Fig. 5. Some experimental results of our boundary-finding algorithm.

near only one boundary, whereas the later consists of nodes near at least two boundaries.

Recall that, in our algorithm, each node examines its *2NG* to determine whether it is near any boundaries: If its *2NG* consists of segment(s) of a "broken ring", then it is near boundaries. Our further observation can be approximately stated as: *a node belongs to SB if its 2NG consists of only a single segment of a "broken ring" (see Fig. 2-b), and belongs to MB if its 2NG consists of at least two segments of a "broken ring" (see Fig. 2-c).* With this observation, we can make a bit of a change to our algorithm presented in Fig. 3 to determine whether the boundary nodes belong to *SB* or *MB* without any further cost. Also, we can further classify *SB* into two sub-classes named SB_1 and SB_2, where SB_2 consists of nodes in *SB* that neighbor at least one node in *MB*. This is done simply by forcing each node in *MB* to broadcast a greeting message to all its neighbors. Fig. 6 illustrates our concepts of *MB*, SB_1 and SB_2: The blue nodes belong to *MB*, the red nodes to SB_1 and the green nodes to SB_2.

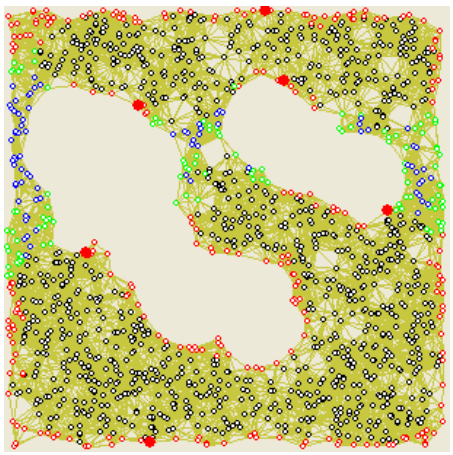


Fig. 6. Subsets of boundary nodes: *MB* consists of blue nodes, *SB1* consists of red nodes, and SB_2 consists of green nodes.

The usefulness of this classification is discussed as follows. Boundary nodes, together with their connectivity, form a graph that is like a road map: Each connected component of nodes in SB_1 serves as a road (in most cases) and each connected component of nodes in the union of *MB* and SB_2 serves as a crossroads or a bridge that links roads together.

Recently, researchers have paid more attention to exploring the geometry information, which is useful in routing and localization [1-3,8,9,12], hidden in the network [4,7,11]. We hope that, with the aid of exploiting this unique map, we can expose more accurately the geometry information hidden in the network.

As previously stated, a hole is formed when nodes in a large area are broken down. The formation of holes is often caused by extreme events such as fire, earthquake, inundation, and so forth. So, in habitat monitoring applications, we need to know if actual holes are formed in order to know whether such an event has happened. In surveillance applications, we may want to know whether an enemy, for example, has left the restricted area or is still in that area but "is hidden" in a hole of the sensor field. We believe that, as far as the development and application of sensor networks go, distinguishing the boundary of the sensor field with those of holes is more beneficial. So, one of our future works will be to distinguish nodes near the (outer) boundary of the sensor field from those near the (inner) boundaries of holes. We hope that this map will also be able to provide us with clues to solve this problem efficiently.

In conclusion, we have proposed a simple, efficient, location-independent algorithm for boundary detection. Routing and localization that make use of the knowledge of boundaries is our future trend of research.

References

[1] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with Guaranteed Delivery in Ad-hoc Wireless Networks", *Wireless Networks*, 2001, 7 (6): 609-616.

[2] J. Bruck, J. Gao, and A. Jiang, "Map: Medial-Axis-Based Geometric Routing in Sensor Networks", *Proc. MobiCom*, 2005.

[3] D. De Couto and R. Morris, "Location Proxies and Intermediate Node Forwarding for Practical Geographic Forwarding", *MIT-LCS-TR824*, 2001.

[4] Q. Fang, J. Gao, and L. Guibas, "Locating and Bypassing Routing Holes in Sensor Networks", *In 23rd Conf. of the IEEE Communications Society (INFOCOM)*, 2004.

[5] S.P. Fekete, Michael Kaufmann, A.Kröller, Katharina Lehmann, "A New Approach for Boundary Recognition in Geometric Sensor Networks", *Proceedings of the 17th Canadian Conference on Computational Geometry*, 2005, pp. 82-85.

[6] S. P. Fekete, A. KrÄoller, D. P. Sterer, S. Fischer, and C. Buschmann, "Neighborhood-Based Topology Recognition in Sensor Networks", *In ALGOSENSORS,* 2004.

[7] Stefan Funke, "Topological Hole Detection in Wireless Sensor Networks and its Applications", *DIALM*, 2005.

[8] Stefan Funke, Nikola Milosavljevi´c, "Guaranteed-Delivery Geographic Routing under Uncertain Node Locations", *In INFOCOM*, 2007.

[9] L. Guibas Q. Fang, J. Gao, V. de Silva, and L. Zhang, "Glider: Gradient Landmark-Based Distributed Routing for Sensor Networks", *In INFOCOM*, 2005.

[10] T. Moscibroda and R. Wattenhofer, "Maximal Independent Sets in Radio Networks", *The 24th ACM Symp. on the Principles of Distributed Computing*

(PODC), 2005.

[11] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic Routing without Location Information", *In Proc. MobiCom*, 2003.

[12] E. Royer and C. Toh, "A Review of Current Routing Protocols for Ad-hoc Mobile Wireless Networks", *In IEEE Personal Communications*, 1999.

[13] H. Attiya, J. Welch, "Distributed Computing: Fundamentals, Simulations and Advanced Topics, Second Edition", *John Wiley & Sons*, 2004.

Thanh Le Dinh

He received his BS and MS degrees in Computer Science from Hongduc Univ. in 2004 and the College of Technology, Vietnam National Univ., Hanoi in 2006, respectively. During 2006~2007, he was a lecturer at Hongduc Univ. Since 2008, he has been an active researcher at the College of Technology, Vietnam National University, Hanoi. His research interests include Distributed Systems, Wireless Networks, P2P Networks and Self-stabilization.