An Improved Connectivity-based Boundary Detection Algorithm in Wireless Sensor Networks

Baoqi Huang, Wei Wu*
College of Computer Science, Inner Mongolia University
Hohhot, 010021, China
Email: cshbq@imu.edu.cn and cswuwei@imu.edu.cn
*Corresponding author

Tao Zhang
Neimenggu Mobile Communication co., LTD.
Hohhot, 010090, China
Email: zhangtao3@nm.chinamobile.com

***Abstract*—This paper tackles the problem of boundary detection by proposing a simple, distributed and connectivity-based algorithm. Our algorithm examines the 2-hop iso-contour of each node, and outperforms existing algorithms examining iso-contours. Specifically, the proposed algorithm makes a rough decision on a suspected boundary node by examining its 2-hop iso-contour, and then refines the decision based on a heuristic operation, which significantly reduces the size of the suspected boundary node set. More importantly, boundary cycles corresponding to inner and outer boundaries are identified and provide valuable knowledge to various applications. A thorough evaluation shows that our algorithm is applicable to dense as well as sparse WSNs.**

I. Introduction

In WSNs (wireless sensor networks), nodes on or near boundaries (termed *boundary nodes*) normally play a more important role than the other nodes. On the one hand, boundary nodes not only directly interact with the outside environment (such as events entering or leaving the region monitored by the WSN, any communication with the outside environment and so on), but also help to extract further information about the WSN structure (which is useful for routing, guiding and management purposes). On the other hand, due to the correspondence between the boundaries of a WSN and its physical environment, such as a building floor plan, a map of a transportation network, terrain variations, and obstacles (buildings, lakes, etc), boundary nodes are important for keeping track of the WSN shape which indicates significant features of the underlying environment. In contrast to the outer boundary of a WSN, boundaries of inner holes (termed *inner boundaries*) are critical indicators of the general health of a WSN, such as insufficient coverage and connectivity. Therefore, boundary detection is of great importance for various WSN applications.

Considerable efforts have been invested in developing boundary detection algorithms, resulting in three categories of methods: geometric, statistical and topological methods. Though the geometric methods can find more accurate boundary nodes than the other two categories, the requirement of node location information limits its application, especially in large scale WSNs. The statistical methods usually make assumptions about the probability distribution of the node deployment, and then probabilistically identify boundary nodes based on some statistical properties under certain network conditions. One major weakness of the statistical methods is the unrealistic requirement on node distribution and density (e.g., the average degree needs to be 100 or higher in [1]). The topological methods [2]–[4] uses topological properties such as connectivity information to detect boundary nodes. Normally, the topological methods have higher packet control overheads than the other two categories due to having to collect connectivity information from neighboring nodes, but do not require node location information and generally outperform the statistical methods.

As a typical topological method, the Funke's algorithm [2] constructs iso-contours[1] of one common node, examines where the contours are broken and outputs boundary nodes with certain guarantees. But, only nodes near the boundaries are identified, and the density requirement is rather high. Khan et al. [3] proposed a variant of Funke's algorithm by examining the 2-hop iso-contour of each node, and argued that one node is an interior node if a closed cycle exists in its 2-hop contour and encloses this node. Another similar algorithm proposed by Chu et al [4] relaxes the condition of distinguishing boundary nodes from interior nodes and identifies more nodes as boundary nodes. Though the latter two algorithms are as simple as the Funke's, they have common weakness, namely requiring a high node density, identifying far more than actual boundary nodes and not showing the identified boundary nodes in a meaningful way.

In this paper, we focus on recognizing accurate boundaries using low-cost connectivity information in WSNs. Like the approaches in [3], [4], our approach is also based on local iso-contours. Specifically, our algorithm utilizes only 3-hop neighboring node information of each node to construct its 2-hop iso-contour, and identifies a set of nodes as suspected boundary nodes. Then, a heuristic operation is introduced to refine the suspected boundary node set. Lastly, by searching the refined set of suspected boundary nodes, our algorithm selects most probable boundary nodes and connect them into boundary cycles. These boundary cycles correspond to both inner and outer boundaries and provide much valuable knowledge to WSN applications. We carry out a thorough

[1]The i-hop iso-contour of a root node contains all the nodes whose minimal hop count to the root is identically i.

978-1-4799-0537-9/13/$31.00 ©2013 IEEE

evaluation and show that our algorithm is applicable in both dense and sparse WSNs.

II. Proposed Boundary Detection Algorithm

Suppose a large number of nodes are scattered in a 2-dimensional geometric region to form a WSN. The wireless channel satisfies the UDG (unit disk graph) model where two nodes are connected if and only if their distance is at most 1. The WSN can be modeled as a graph $G(V, E)$ where the vertex set V denotes nodes and the edge set E denotes communication links between pairs of nodes. Without loss of generality, we assume the graph $G(V, E)$ is connected. Besides, we use the following mathematical notations throughout this paper: the i-hop neighbor nodes of node n_x is denoted $N_i(n_x)$; $|\cdot|$ computes the cardinality of a set.

For clarification, we call the nodes on geometric boundaries as *actual boundary nodes*, and accordingly call the other nodes as *actual interior nodes*.

The basic idea of the algorithm is firstly identifying a set of suspected boundary nodes, then refining it and lastly searching it for boundary cycles. The outline is listed below.

1) Each node, say node n_0, maintains a local variable, denoted f_t, which is initialized based on the 2-hop iso-contour extracted from the subgraph of $G(V, E)$ induced by n_0, $N_1(n_0)$ and $N_2(n_0)$. The value of f_t roughly decides whether this node is a boundary node: $f_t = 1$ indicates that the associated node is likely to be a boundary node and is termed a *suspected* boundary node, while $f_t = 0$ indicates that the associated node is identified to be an interior node, as shown in Fig. 1(b).
2) Each node exchanges its f_t with its 1-hop neighbor nodes, and suspected boundary nodes locally performs a heuristic operation to refine the decision on f_t according to the ratio of identified interior nodes in their neighborhoods, as illustrated in Fig. 1(c).
3) Each suspected boundary node, say n_0, maintains a local variable, denoted f_d, differentiating itself from the other suspected boundary nodes in $N_1(n_0)$ in terms of the proximities to geometric boundaries. The value of f_d of node n_0 is evaluated according to the minimal hop counts from node n_0 and suspected boundary nodes in $N_1(n_0)$ to all the identified interior nodes.
4) The suspected boundary nodes cooperatively search for boundary cycles based on their values of f_d. As a result, a set of boundary cycles corresponding to tight inner and outer boundaries are obtained, as shown in Fig. 1(d)

In what follows, we shall describe the critical steps of the proposed algorithm in detail.

A. Initializing f_t

In this step, node n_0 initializes its local f_t using the return value of a key function $INITIALIZE(n_0)$. According to the initial value of f_t, a coarse judgement as to whether node n_0 is likely to be a boundary node is made, so that the nodes in the WSN are classified into suspected boundary nodes with $f_t = 1$ and identified interior nodes with $f_t = 0$.

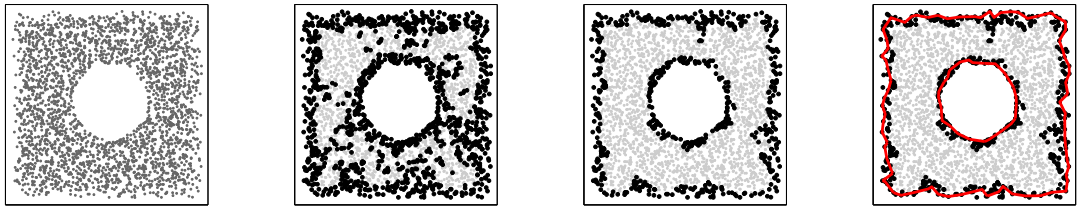
procedure $Initialize(n_0)$

Construct $G_2(V_2, E_2)$ based on $G(V, E)$, where $V_2 = \{i : i \in N_2(n_0)\}$ and $E_2 = \{e_{ij} : i, j \in V_2 \bigwedge e_{ij} \in E\}$
Index nodes in V_2 by $1, 2, \cdots, |V_2|$
$roadmap \leftarrow$ Integer array of size $|V_2|$ with all zeros
$V_c \leftarrow V_2$
$expanding_path \leftarrow$ Empty Stack
while $|V_c| > 0$ **do**
  $n_r \leftarrow$ An arbitrary node in V_c
  $V_c \leftarrow V_c - n_r$
  $Push(expanding_path, n_r)$
  $roadmap(n_r) \leftarrow 1$
  $step \leftarrow 2$
  while true **do**
    $candidate_nodes \leftarrow V_2 \bigcap N_1(n_r)$
    $n_m \leftarrow$ The node in $candidate_nodes$ and satisfying $roadmap(n_m) \leq roadmap(candidate_nodes)$
    if $roadmap(n_r) - roadmap(n_m) > L_{th}$ **then**
      return 0;
    end if
    $candidate_nodes \leftarrow V_c \bigcap N_1(n_r)$
    if $|candidate_nodes| > 0$ **then**
      $V_c \leftarrow (V_c - candidate_nodes)$
      $roadmap(candidate_nodes) \leftarrow step$
      $step \leftarrow step + 1$
      $n_r \leftarrow$ An arbitrary node in $candidate_nodes$
      $Push(expanding_path, n_r)$
    else if $step > 2$ **then**
      $step \leftarrow step - 1$
      $recover_nodes \leftarrow \{n : roadmap(n) = step\}$
      $roadmap(recover_nodes) \leftarrow 0$
      $V_c \leftarrow (V_c + recover_nodes - n_r)$
      $Pop(expanding_path)$
      $n_r \leftarrow$ The top node in $expanding_path$
    else
      break
    end if
  end while
  return 1
end while

Fig. 2. The function $Initialize(n_0)$. $Push(expanding_path, n_r)$ pushes n_r into the stack $expanding_path$ and $Pop(expanding_path)$ pops the top node from the stack $expanding_path$.

The function $INITIALIZE(n_0)$ identifies node n_0 as a suspected boundary node if a closed path enclosing node n_0 is not found in its 2-hop iso-contour, similarly to [3], [4]; for simplicity, we term this condition as the CLOSED-CYCLE condition. Since the method for examining the CLOSED-CYCLE condition in [3] randomly chooses a node for node expanding, there is a possibility that an eligible closed path is not found, while the function $INITIALIZE(n_0)$ considers every possible situation.

The pseudo-code of the function $INITIALIZE(n_0)$ is described in Fig. 2. The parameter L_{th} denotes the minimal



(a) (b) (c) (d)

Fig. 1. Snapshots of each step during running the proposed algorithm in a WSN with the average node degree of 18.8. (a) The node deployment in the WSN; (b) After initializing f_t at each node; (c) After performing the erosion operation; (d) After searching for boundary cycles. Dark black points denote nodes with $f_t = 1$, grey points denote nodes with $f_t = 0$, and red curves denote boundary cycles.

length of all possible closed paths that satisfy the CLOSED-CYCLE condition. Under the UDG model, it is straightforward that $L_{th} = 7$. Obviously, nodes with $f_t = 0$ are definitely actual interior nodes, but nodes with $f_t = 1$ are only probable to be actual boundary nodes, and thus are named suspected boundary nodes. From another point of view, one can readily deduce that the initial values of f_t for actual boundary nodes are always 1, whereas take either 0 or 1 for actual interior nodes, implying that a portion of actual interior nodes are improperly discovered as suspected boundary nodes. Hence, the CLOSED-CYCLE condition is just sufficient.

The successful rate of the function $INITIALIZE(n_0)$ reduces with decreasing the node density. In order to enhance the performance of the proposed algorithm in sparse WSNs, we can enlarge the neighborhood of each node to examine the CLOSED-CYCLE condition, intuitively, by replacing the original graph $G(V, E)$ in the execution of the function $INITIALIZE(n_0)$ by its power graph, denoted $G^2(V^2, E^2)$. As a result, the 2-hop neighbor nodes of each node in the graph $G^2(V^2, E^2)$ are actually the 3- and 4-hop neighbor nodes in the original graph $G(V, E)$, and each node needs to collect all the neighbor nodes within 5 hops to accordingly construct the subgraph $G_2(V_2, E_2)$ (See Fig. 2) of the power graph $G^2(V^2, E^2)$.

B. Performing the Erosion Operation

After initializing f_t, the number of suspected boundary nodes is multiple times of that of actual boundary nodes, and we have to refine the decision as to f_t at nodes with $f_t = 1$.

Observations indicate that, a majority of 1-hop neighbor nodes of actual boundary nodes have $f_t = 1$, but some actual interior nodes, which are adjacent to actual boundary nodes or which have relatively sparse neighborhoods, often turn out to have $f_t = 1$, and a majority of 1-hop neighbor nodes of these nodes have $f_t = 0$. In view of this feature, for any node with $f_t = 1$, if the ratio of its 1-hop neighbor nodes with $f_t = 0$ is above a threshold, denoted E_{th}, this node is highly probable to be an actual interior node, so that its f_t should be accordingly set to 0. As such, we design a heuristic operation, which is like the erosion operation in image processing, to shrink the size of the suspected boundary node set.

C. Determining f_d

To present our result in a meaningful way, we strive to obtain boundary cycles corresponding to both inner and outer geometric boundaries of the WSN. Prior to this, we define a new local variable, denoted f_d, at each suspected boundary node. The value of f_d at node n_0 is computed as below.

$$f_d = h(n_0) + \frac{|\{n_x | n_x \in N_1(n_0) \bigwedge h(n_x) = h(n_0) - 1\}|}{|N_1(n_0)|}, \quad (1)$$

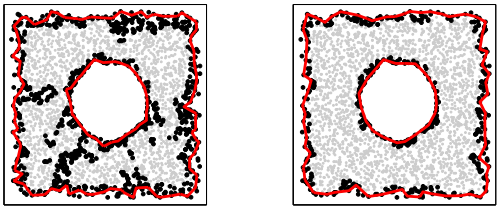
where $h(n_x)$ returns the minimal hop count from a suspected boundary node n_x to all the identified interior nodes, i.e., nodes with $f_t = 0$. To compute $h(n_x)$, the identified interior nodes are supposed to be synchronized among themselves by using the method e.g. in [5], and as a "root node" start to flood the WSN at roughly the same time; then, each suspected boundary node records the minimal hop count to the root node, namely the value of $h(n_x)$. During the network flooding, each suspected boundary node also records the minimal hop count from each of its 1-hop neighbor nodes to the root node, which are useful in the determination of f_d by (1).

It is evident taht, the greater is f_d, the further is the corresponding node away from the actual interior nodes, and in other words, the closer is this node to the geometric boundaries. Therefore, f_d differentiates each node with $f_t = 1$ from its 1-hop neighbor nodes in terms of the proximities to the geometric boundaries, and is much helpful for finding boundary cycles in the next step.

D. Searching for Boundary Cycles

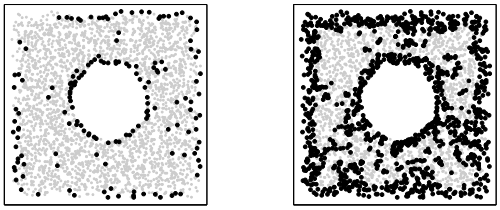
The proposed algorithm searches all the suspected boundary nodes for boundary cycles based on the same mechanism (i.e., node expanding) as in the function $INITIALIZE(n_0)$ except for the following two aspects.

First, unlike the function $INITIALIZE(n_0)$ which only involves the nodes in $N_2(n_0)$ and is realized in a centralized fashion, searching for boundary cycles deals with a relatively large number of nodes and is thus suitable for a distributed implementation. To do so, each node with the locally minimal f_d will serve as a starting node to launch a search process. To reduce the risk of having two search processes to conflict, a random waiting time is assigned to each starting node. However, if two searching processes unfortunately conflict,



(a) (b)

Fig. 3. Results of applying the proposed algorithm in a WSN with a circular hole: (a) the average degree is 15.6; (c) the average degree is 22.4.



(a) (b)

Fig. 4. Results of applying existing boundary detection algorithms. (a) The algorithm in [2]; (b) The algorithm in [4]. The average node degree is 18.8.

namely that one node is involved in two search processes, the one whose starting node's ID is smaller will stop, while the other one will continue.

Second, node is arbitrarily expanded in the function $INITIALIZE(n_0)$, but is expanded in a weighted way in the process of searching for boundary cycles. Specifically, the node with the maximal f_d among all expandable nodes is preferentially expanded, which is the key factor for obtaining tight boundary cycles.

Like L_{th} in the function $INITIALIZE(n_0)$, we define a similar parameter C_{th} for specifying the boundary cycle's length of the minimal boundary hole that can be recognized by the proposed algorithm. If C_{th} is set to a small number, the searching processes might be terminated too early so as to fail to find boundary cycles. C_{th} is dependent on the deployments of WSNs and is suggested to be greater than 10.

With boundary cycles, it is straightforward to distinguish an outer boundary from each inner boundary by comparing the lengths of their corresponding boundary cycles or by comparing the length of each boundary cycle with the perimeter of the WSN which can be roughly estimated based on the deployment information. Moreover, we can further refine the set of suspected boundary nodes by excluding the nodes which are more than 1-hop to boundary cycles.

III. Performance Evaluation

We conduct extensive simulations to evaluate the performance of the proposed algorithm with respect to the node density and existing boundary detection algorithms.

In this simulation, we assume the UDG model and deploy 2700 sensor nodes according to a uniform and random distribution in a square region of size 240 m with one circular hole of radius 50 m, which is the same as the WSN shown in Fig. 1. The average node degree of the graph is varied by adjusting the communication radius of the UDG model. The choices of C_{th} and E_{th} are 15 and 0.8 respectively.

As illustrated in Fig. 3 and Fig. 1(d), the larger is the average node degree, the more accurate are suspected boundary nodes, the tighter are the boundary cycles, and hence, the better is the performance of the algorithm. The underlying reason is that, a larger node density increases the possibility of forming a closed path enclosing each interior node due to having more 2-hop neighbor nodes, and thus lowers the chance of mistakenly identifying interior nodes as suspected boundary nodes.

Furthermore, we implement another two typical connectivity-based boundary detection algorithms [2], [4] for comparison. As can be seen in Fig. 4, the algorithm in [2] can only identify a small portion of actual boundary nodes, but cannot correctly find boundaries of the WSN; the algorithm in [4] can identify all the actual boundary nodes, but mistakenly include many actual interior nodes. In contrast, the proposed algorithm obviously deliver superior results, as shown in Fig. 1.

IV. Conclusions

In this paper, we reported a simple and distributed boundary detection algorithm for WSNs, which utilizes only local connectivity information at each node and is able to produce boundary cycles corresponding to tight inner and outer boundaries of WSNs. We carried out a thorough evaluation and showed that our algorithm is applicable in both dense and sparse WSNs.

Acknowledgements

This work is supported by the Program of Higher-level talents of Inner Mongolia University under 125124 and the Inner Mongolia Autonomous Region Science and Technology Innovation Guide Reward Fund Project under 20121317.

References

[1] S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann, "Neighborhood-based topology recognition in sensor networks," in *First International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, vol. 3121, Turku, Finland, Jul. 2004, pp. 123–136.

[2] S. Funke, "Topological hole detection in wireless sensor networks and its applications," in *Joint Workshop on Foundations of Mobile Computing*, 2005, pp. 44–53.

[3] I. Khan, H. Mokhtar, and M. Merabti, "A new self-detection scheme for sensor network boundary recognition," in *Proceedings of the IEEE 34th Conference on Local Computer Networks (LCN)*, Zurich, Switzerland, Oct. 2009, pp. 241–244.

[4] W.-C. Chu and K.-F. Ssu, "Decentralized boundary detection without location information in wireless sensor networks." in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Paris, France, Apr. 2012, pp. 1720–1724.

[5] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys)*, Los Angeles, California, USA, Nov. 2003, pp. 138–149.