Hole Detection and Boundary Recognition in Wireless Sensor Networks

Kun-Ying Hsieh
Dept. of Computer Science and Information Engineering
National Central University
Jhongli, 32054, Taiwan
E-mail: rocky@axp1.csie.ncu.edu.tw

Jang-Ping Sheu
Dept. of Computer Science
National Tsing Hua University
Hsinchu, 30013, Taiwan
E-mail: sheujp@cs.nthu.edu.tw

Abstract—**Coverage holes may exist in wireless sensor networks (WSNs) due to presence of obstacles or invalid sensor nodes in the sensing field. Normally, the holes make the data routing failure when the nodes transmit their data back to the sink. In this paper, distributed protocols are developed to identify the boundary nodes surrounding the holes of the sensing filed in WSNs without using any location information. Experimental results demonstrate that our algorithm can precisely and correctly identify the boundary nodes even in sparsely sensors deployed regions. Besides, our algorithm can give better performance in terms of control packet overhead and simulation time as compared to previous work.**

Keywords: Boundary recognition, distributed algorithms, holes detection, wireless sensor networks

I. INTRODUCTION

Wireless sensor network (WSN) is composed of several sensor nodes deployed and scattered over a specific monitoring region for collecting sensed data. In general, sensor nodes are not equipped with GPS as it is expensive and energy-consuming. Hence, sensor nodes normally have no location information in WSNs. Most of the applications in WSNs require sufficient sensing coverage and connectivity. However, holes may exist within the network due to obstacles such as ponds or small hills that cause the network partitioned and uncovered. Moreover, the holes may make the routing failure when a node transmits sensing data back to the sink. As mentioned above, all we concern is to obtain the boundary information of the holes and network. This boundary information can support the routing protocol such as GLIDER [3] to avoid the holes and increase the routing performance, or to promote the performance of sensor network applications or the implementation of networking functionalities.

In this paper, we study the problem of discovering the nodes on the boundaries which may be inner that encircles the holes and outer that surrounds the network boundaries. Those selected boundary nodes are connected and could form the meaningful boundary cycles. Here, we assume that each sensor node has a unique ID but without having location information, and its communication graph is a unit disk graph. In the network, the sensor nodes are randomly and densely deployed, and some irregular huge holes exist. The main contribution of this paper is to develop a simple and practical distributed algorithm for discovering boundary nodes with less communication overhead and time. All of nodes equally share the energy cost and only few nodes in the border area of holes and network need to pay much more energy to discover the boundary nodes. Accordingly, the network lifetime will be increased. Besides, our algorithm can precisely identify the boundary nodes even in sparsely deployed environment. Furthermore, the scheme of maintaining the information of boundary nodes is also proposed in this paper.

The remainder of this paper is organized as follows. Section II reviews some related works. Our holes detection and boundaries detection algorithm are proposed in Section III. The performance evaluation of our algorithms is presented in Section IV. Conclusion is made in Section V.

II. RELATED WORKS

Normally, the boundary detection algorithms can be classified into three categories: geometric, statistical and topological methods. The approaches in geometric methods [2, 8], rely on the nodes having geographical locations. Although, the geometric method can find more accurate boundary nodes than other two methods in a WSN, each node has to equip extra device such as GPS to obtain the geographical locations. Unlike geometric methods, the statistical methods [1, 4, 5] without having location information usually assume the sensor nodes are uniformly distributed on the sensing field. Thus, we can probabilistically identify the boundary nodes based on some statistical properties under certain network conditions. The major weakness of the statistical methods is that the criteria for identifying the boundary nodes acquired from the statistical characteristics cannot guarantee to find out the boundary nodes precisely. The topological methods [6, 7, 9, 10] use the topological properties such as the information of connectivity to identify the boundary nodes. Normally, topological method has higher packet control overheard than previous two methods due to having to collect information form neighboring nodes; however, it does not need location information and has better accuracy of finding boundary nodes than statistical method. The approach presented in this paper belongs to the topological methods. The algorithm only exploit the collected local information to identify the boundary nodes of holes with unknowing nodes location.

III. DISTRIBUTED BOUNDARY RECOGNITION ALGORITHM

We first outline the algorithm and then explain details of each step as follows. Our Distributed Boundary Recognition Algorithm (DBRA) consists of four phases. The first phase is to select *Closure Nodes* (*CNs*) which roughly enclose the holes and frontier of the sensing field. It is to be noted that, for simplification, throughout this paper we are using the term obstacles instead of the holes and frontier of sensing field. In the second phase, those closure nodes are connected with each other to form *Coarse Boundary Cycles* (*CBCs*) for identifying each obstacle. The third phase is proposed to discover the exact *Boundary Nodes* (*BNs*) and connect them to refine the *CBCs* to be final boundaries. The last phase is proposed to maintain the integrity of *BN*s while the boundary is broken due to nodes failure.

A. Closure nodes selection

In this phase, we select the *CN*s, which enclose the obstacles. We construct a *Virtual Hexagonal Landmark* (*VHL*) by selecting some specific nodes to be the *Landmark Nodes* (*LNs*) within the network. Here, the *LNs* are regularly hexagonal and spread over the network even when *VHL* is constructed on irregular network shapes. Then, each *LN* identifies itself to become a *CN* or still remains as an *LN*.

This work was supported by the MOEA and NSC of R.O.C. under grants 98-EC-17-A-19-S2-0052 and NSC 97-2221-E-007--039-MY3.

The *VHL* construction protocol runs as follows. Firstly, a specific node such as sink node is assigned as the first starting landmark node (*LN*) with unique ID (default value is 0) of the network and broadcasts an *LN_create*(*LN_ID*, *TTL*) packet within *r* hops. Here, the packet includes the *LN's* unique ID and the default value of *TTL* (Time to live) with *r* hops. Upon receiving the packet, each node stores the *LN_create* packet and then forwards it to the other nodes within its communication range. After the *LN_create* packet is flooded over the network with *r* hops, the nodes have (*r* - *n*) to *r* hops distance to the *LN* shall form a ring-shaped area. Here, *r* indicates the radius of broadcasting range and *n* indicates the width of the ring-shaped area.

If the nodes within ring-shaped area receives *LN_create* packet and find the *LN's* ID equals to zero, then the nodes are aware in the first ring-shaped area and need to elect one node within the ring-shaped area as the second *LN*. We assume that each node's ID in the network is unique. The node with maximum ID in the ring-shaped area centered on the first *LN* will be elected as the second *LN*. The second *LN* is selected from the nodes whose distance to the first *LN* is between (*r* - *n*) to *r* hops. The nodes having maximum ID within its one-hop neighbors will broadcast a packet with its ID in the ring-shaped area. Each node within the ring-shaped area that receives a broadcast packet will forward it if the ID in the packet is larger than its own ID. Finally, the packet with the maximum ID will go around the entire ring-shaped area and back to the initiated node, and this node will become the second *LN*. For example, in Fig. 1, assuming node *A* is the first *LN*, and node *B* has the maximum ID within the ring-shaped area, node *B* will become the second *LN* of the network. Note that, the previous procedure of electing the second *LN* will be only executed once.

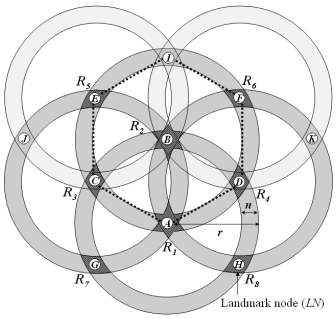


Figure 1: The construction of *VHL* by electing the *LNs* from the intersecting regions of the ring-shaped areas.

Like the first *LN*, the second *LN* also broadcasts the *LN_create* packet including its ID. Each node receives the *LN_create* packet, stores the received *LN's* ID, and forwards the packet with a decreasing *TTL* to other nodes in its communication range. Only the nodes within two intersection regions of two rings centered at the first and second *LNs* will elect the *LNs*. For example, in Fig. 1, assuming that the first *LN A* is located in region R_1 and second *LN B* is located in region R_2, R_3 and R_4 will be the intersection regions of those two ring-shaped areas. Two nodes *C* and *D* having maximum ID within the regions R_3 and R_4 will be elected as the third and fourth *LNs*, respectively. An *n*-hop local flooding can be used to elect the *LN* in each region. Similarly, we can get the third and fourth ring-shaped areas centered at the third and fourth *LNs C* and *D*, respectively. However, we ignore the intersection regions R_1 and R_2 which have elected *LNs* before. The above procedure continues until any new *LN* in the network cannot be found. Then the construction of *VHL* is completed. It is to be noted that, each *LN* waits a predefined threshold time T_c to gather other six different *LN_create* packets from its six neighboring *LNs* after broadcasting the *LN_create* packet. When an *LN* cannot receive six *LN_create* packets within time T_c, it sets itself as a *CN*; otherwise, it still remains as a normal *LN*. Once *LN* changes its role to a *CN*, each *CN* broadcasts the *CN_Info*(*CN_ID, TTL*) packet including its unique ID to inform its neighboring *CNs*. After all the *LNs* have determined their roles, all the *CNs* will enclose the holes and frontier of the sensing field, as shown in Fig. 2, and each *CN* will know its neighboring *CN*s.

B. Coarse boundary cycles identification

In the second phase, we connect the *CNs* to form the rough boundaries enclosing the obstacles. These rough boundaries are named as Coarse Boundary Cycles (*CBCs*) and each of them is assigned a unique ID (i.e. *CBC_ID*). Before each *CN* determines whether it belongs to a *CBC* or not, there are two cases when creating the *CBCs*, as shown in Fig. 2. The first case is *CBCs* are separated by *LNs* and each *CN* in *CBCs* has only two adjacent *CNs*. Note that, to one *CN*, its adjacent *CN* means the closest *CN* to it where the distance between them is within (*r* - *n*) to *r* hops and this adjacent *CN* is also one vertex of virtual hexagon centered at the *CN* in *VHL*. In second case, the *CBCs* are adjacent to each other without separated by *LNs* and each *CN* in *CBCs* has more than two adjacent *CNs*. The Area 1 and Area 2 of Fig. 2 show these two corresponding cases. When creating the *CBC*, the main goal is each *CN* only connects to two adjacent *CNs* and the created *CBCs* do not cross each other. Next, we describe the algorithm of creating *CBCs* according to the different cases.

Upon receiving the *CN_Info* packets from the adjacent *CNs* as described in Section III-A, each *CN* saves these packets and counts the received number of *CN's* ID. After waiting for a predefined time period, any *CN* has not joined the *CBC* and only has two adjacent *CNs* will check whether its ID is larger than other two adjacent *CN's* IDs. If it is true, it will start to create *CBC* and assign its ID to be the *CBC's* ID. However, if it is false, the *CN* will wait to join a *CBC* without initiating to create the *CBC*. Next, the *CN* broadcasts a *CBC_create*(*CN's ID*, *CBC's ID*, *CBC_list*) packet. Here, the *CBC_list* includes this *CN's ID*, the adjacent *CN's* ID that *CN* wants to connect to and the accumulated list of *CN's* ID. Upon receiving the *CBC_create* packet, if *CN's* two adjacent *CNs* has not joined the *CBC* (that is the *CN* has not saved *CBC_ID* yet), they will agree to join this *CBC* and reply *CBC_create_reply*(*CN's ID, CBC_ID*) packets back to the *CN* and also save the *CBC_ID* they belong to. Then, the connections of the *CN* between two adjacent *CNs* are created. After the *CN* is connected to one adjacent *CN*, according to the number of adjacent *CNs* of the *CN*, there are two cases corresponding to the aforementioned two cases of *CBCs* when this *CN* connect to other adjacent *CN*.

Case 1: The *CN* has only one adjacent *CN* that can connect to. If this adjacent *CN* has not joined to any *CBC* (that is adjacent *CN* has not received any *CBC_create* or *CBC_create_reply* packet from its adjacent *CN* yet), the *CN* will follow the previous procedure to connect to this adjacent *CN*. However, if this adjacent *CNs* has joined to different *CBC* with different *CBC_ID*, the *CN* will join the *CBC* with largest value of *CBC_list*. Because the larger *CBC_list* value the *CN* joins to, the less number of remaining *CNs* it has to reversely re-connect and the fewer overhead is wasted.

Case 2: The *CN* has more than one adjacent *CNs* that can connect to. The *CN* will wait enough *CN's* ID messages without doing anything until it can identify which adjacent *CN* should connect to. Once the *CN* can determine which adjacent

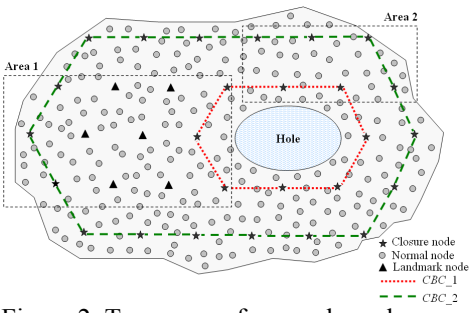


Figure 2. Two cases of coarse boundary cycles.

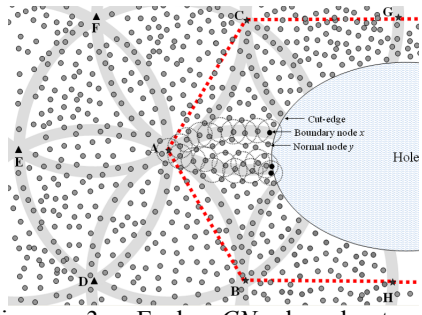


Figure 3. Each *CN* broadcasts a *BN_create* packet flooding along the ring-shaped areas.

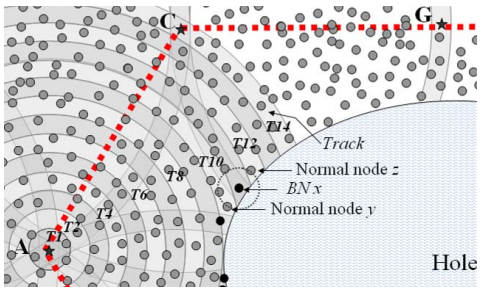


Figure 4. The *Tracks* of the *CN A*.

CN it should connect to, it follows the same procedure of Case has joined to the *CBC_1*. However, it has another two adjacent *CNs*, *H* and *C*, can connect to. Therefore, *CN B* will wait for other messages until it can identify to connect to *CN H* or *C*. As time goes on, *CN G* connects to the *CN H* with *CBC_2*. At this time, after receiving the *CBC_create_reply* packet from *CN H*, *CN B* can eliminate to connect to the *CN H* but *CH C*. Hence, *CN B* follows the same procedure of Case 1 to connect to the *CN C*. Apparently, *CN H* will face the same situation like the *CN B* having three adjacent CNs *B*, *C* and *I*. Since, *CN H* has received the *CBC_create_reply* packet from *CN B* when *CN B* agrees to connect to the *CN A*. Additionally, upon receiving more connection messages, *CN H* can determine to link to *CN I*. Similarly, upon receiving connection messages from *CN H* and *I*, *CN C* can determine to connect to the *CN D* as *CN I* has joined to the *CBC_2*.

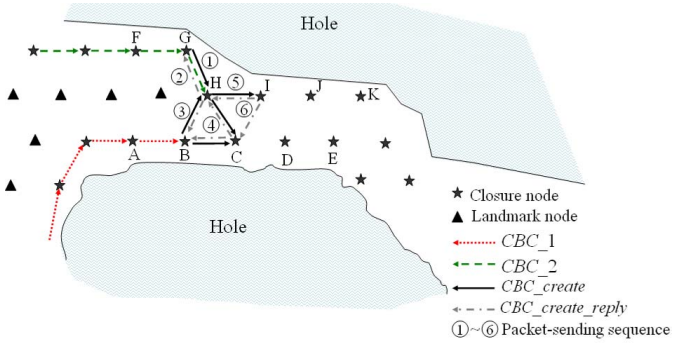


Figure 5. Each *CN* has more than one adjacent *CN* will wait enough messages until it can identify which one adjacent *CN* it should connect to.

C. Discover exact boundary nodes

In this section, we find out the exact *Boundary Nodes* (*BNs*) those who tightly surround the obstacles for creating tight boundaries. At first, some *BNs* near the obstacles are selected to initiate the procedure. Each *BN* connects to its two adjacent *BNs* closest to the obstacle on its two-side, separately. Eventually, the *BNs* connect with each other one by one enclosing the obstacles to form the complete boundaries.

According to the definition of ring-shaped area in Section III-A, each *CN's* ring-shaped area must pass through its two adjacent *CNs*. Similarly, each *CN* is also passed through by its two adjacent *CNs'* ring-shaped areas. For example, in Fig. 3, *CN A* must pass through by its two adjacent *CNs C* and *B's* ring-shaped areas. Additionally, some *CNs'* ring-shaped areas are cut off by obstacles; the flooding of packets along these ring-shaped areas must be stopped by the boundaries of obstacles. Hence, the main idea of selecting the initiated *BNs* is let each *CN* flood the packets along its two adjacent *CNs*' ring-shaped areas and then the nodes on that areas having maximum hop counts will be selected as the initiated *BNs*. For

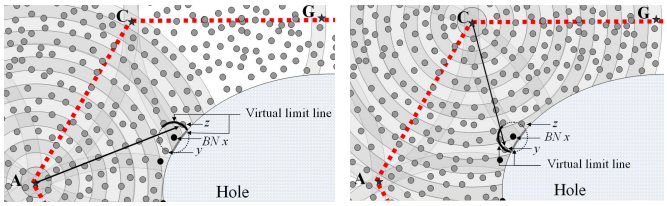
l to connect to that adjacent *CN*. For example, in Fig. 5, *CN B* example, in Fig. 3, *CN A* floods a *BN_create*(*CN_ID*, *adj_CN_ID1*, *adj_CN_ID2*, *CBC_ID*, *TTL*) packet along its two adjacent *CNs B* and *Cs'* ring-shaped areas. Here, *adj_CN_ID1* and *adj_CN_ID2* means the *CN_ID* of *CN's* two adjacent *CNs*. Since, in Section III-A, each *CN* broadcasts the *CN_info* packet to inform its adjacent *CNs* and the node within this broadcasting range, who receives this *CN_info* packet, will also save the *CN_ID* from this packet. Therefore, the nodes located only on the ring-shaped areas and have received three *CN_ID* such as *CN_ID A*, *B* and *C* will flood the *BN_create* packet. The flooding of *BN_create* packets along the ring-shaped areas will continue until it reaches at the end-node. Here, the end-node is the node that receives the *BN_create* packets with maximum hop counts compared to its neighboring nodes and must be located around the cut-edge of ring-shaped area. For example, in Fig. 4, the normal node *x* will receive a packet with maximum hop counts and then it will be selected as an initiated *BN*. Eventually, all the end-nodes will be selected to be the initiated *BNs*.

Based on these initiated *BNs*, then we discover the remaining *BNs*. Each initiated *BN* will select two adjacent nodes individually along its left and right hand side as *BNs*. Here, the selected *BNs* are within initiated *BNs'* one-hop communication range and as close as to the obstacle as possible. Based on the connection procedure of each *BN* to its left and right hand side *BN*, the *BN*s will connect one by one to form a complete boundary. Firstly, each *BN* has to collect the information from its one-hop neighboring nodes for selecting the *BNs*. Each *BN* broadcasts a *BN_select_req*(*ID*) packet to its one-hop neighboring nodes. The node that receive this packet will randomly reply a *BN_select_reply*(*ID*, *Is_BN*, *adj_CN_ID1*, *CN_ID1_hops*, *adj_CN_ID2*, *CN_ID2_hops*, *adj_CN_ID3*, *CN_ID3_hops*) packet back to the *BN*. Here, the *Is_BN* represents whether the node is already a *BN*. The *adj_CN_ID1*and *CN_ID1_hops* represents the node has received the *CN_info* packet from which *CN* and its corresponding hop counts, respectively. Each node may receive at most three *CN_info* packets from its three different adjacent *CNs*. For example, node *y* in Fig. 3 can receive three *CN_info* packets from the adjacent *CNs A*, *B* and *C*, separately.

Obviously, for each *BN*, the selected two *BNs* have to locate separately along its left and right hand side without having them both on the same side. However, each *BN* has no location information to distinguish whether the selected two *BNs* are individually on its two-side or not. Here, we use the *Tracks* which are the rings with different hop counts to assist the *BN* to distinguish. Since each *CN* broadcasts the *CN_info* packet with *R* hops in Section III-A, each node receives this packet with the same hop counts will form a ring. For example, in Fig. 4, *CN A* broadcasts a *CN_info* packet with 14 hops, and then there must exist 14 rings centered at the *CN A* in the

network. Here, we name the rings as the *Tracks* and the distance between each *Track* is one hop. To each *CN*, the sequence of *Tracks* is continuous. For a *BN*, if its selected two *BNs* have different order of *Tracks*, then they must on its two-side. Therefore, *BN* can distinguish whether the two selected *BNs* on the same side by checking whether their *Track* orders are different, even unknown they are located on the left or right hand side.

Now the initiated *BN* can start to select two new *BNs* on its two-side depending on the aforementioned collected information from one-hop neighbors and technique of *Tracks*. The goal of each *BN* is to select the two new *BNs* on its two-side as close to the boundary of the obstacle and *BN's* communication range as possible. Obviously, in Fig. 6(a) and (b), it seems that there are two virtual limit lines made by the boundary of the obstacle and *BN's* communication range that the position of selected new *BN* will not above it. Under this constrain, the best selected new *BN* is located on the intersection point of this two virtual limit lines as it is very close to the boundary of the obstacle. In fact, we only can select the new *BN* as close as to this intersection point as possible, but exactly located on the intersection point. Additionally, in Fig. 6(a), if we assume *CN A* as the starting point and the intersection point as the end point, then the farthest node to the *CN A* must also be the nearest node to the intersection point. Therefore, the main idea of each *BN* to select two *BNs* on its two-side is that each *BN* firstly chooses two different adjacent *CN* on its two-side as reference *CNs*, separately. Then, two nodes having maximum distance to the reference *CNs* on their opposite side will be selected as new *BNs*, separately. For example, in Fig. 6(a), the *BN x* first selects the *CN A* as the first reference *CN* as *BN* is located on the 11th track and its two one-hop adjacent nods *y* and *z* separately are located on the 10th and 12th track of the reference *CN*. Then, node *z* having maximum distance to the reference *CN A* will be selected as the new *BN*. Next, in Fig. 6(b), *BN x* follows the same procedure to select the *CN C* as the second reference *CN* and then select the node *y* as the other new *BN* on the opposite side.



(a) (b)

Figure 6. The *BN x* referring to (a) the reference *CN A* to select node *z* as the new *BN* and (b) the reference *CN C* to select node *y* as the new *BN*.

D. Boundary maintenance

As time goes on, some sensor nodes along the boundaries may exhaust their energy and may die. That means the boundaries will be fragmented without integrity. In order to detect the fragmentation of boundary, each *BN* periodically exchanges beacon messages with its two adjacent *BNs*. Once the *BN* cannot receive the beacon message after waiting for a predefined time period, it assumes that the adjacent *BN* is invalid and the boundary is fragmented. In our protocol, *BN* selects a normal node having maximum adjacent broken nodes as the new *BN* as compared to its one-hop neighboring nodes.

When *BN* finds more than one adjacent nodes as invalid, it broadcasts a *BN_repair_req*(*BN's* ID) packet to its one-hop neighboring nodes. The node, except the *BN*, that receives this packet will broadcast the *BN_Test*(Node's ID) packet randomly to test whether its one-hop neighboring nodes are available. These one-hop neighboring nodes receive the *BN_Test* packet also randomly reply a *BN_Test_Ack*(Node's ID) packet back to the initiating node. After waiting a predefined time, the node compares the received information of *BN_Test_Ack* packets with its neighbor table to check whether the node is available and then obtains the number of broken nodes. Next, the node sends *BN_repair_reply*(Node's ID, *Num_broken_node*) packet back to the *BN*. The *BN* selects a node having maximum value of *Num_broken_node* from the received *BN_repair_reply* packets as a substitute *BN*. If there is more than one packet with the same value of *Num_broken_node*, *BN* selects the node with maximum ID to be a substitute *BN*. Eventually, *BN* sends the *BN_Info*(*BN's ID*, *new BN's ID*) packet to inform the selected node to be a substitute *BN* and then the substitute *BN* replies the *BN_Reply*(*BN's ID*) back to the *BN*.

IV. Simulation and Performance Analysis

To evaluate the performance of the proposed Distributed Boundary Recognition Algorithm (DBRA), it is implemented in NS-2 with the latest version 2.33 on Linux platform of Fedora version 9. The proposed algorithm is compared with TBRA (Topological Boundary Recognition Algorithm) present in [10]. More detail simulation parameters are shown in table I. Our DBRA outperforms TBRA in terms of accuracy ratio especially when node degree is low, and has less control packet overhead and simulation time when lots of holes are within the network. Note that, the DBRA is a fully distributed algorithm; but, the procedure of removing the *cuts* in TBRA is manual.

The metrics for comparing performance are list below:

- *Accuracy ratio*: It is the value of total number of correct *BNs* selected by the proposed algorithms divided by the value of total number of *BNs* should be selected.
- *Control packet overhead*: It is concerned with the total number of packets exchanged.
- *Simulation time*: It is concerned with the total execution time of finding all *BNs*.

Table I: The simulation parameters in NS-2

Simulation parameters	Initial values
Number of nodes	3500
Shape of sensing filed	Square
Size of sensing field	500m × 500m
Communication range	13m, 15m, 17m, 20m
Node degree	7, 10, 13, 16
Shape of holes	Circle
Number of holes	1, 2, 3, 4, 5, 6, 7, 8
r value	6
n value	1

A. Effect of node degree on percentage of accuracy ratio

In this experiment, as shown in Fig. 7, the number of nodes, size of sensing field, *r* value and *n* value adopt the initial values. However, there is only one circle hole with the radius of 90m exists in the sensing filed and the node degree is varied from 7 to 16 with interval of 3 by adjusting the communication total number of *BNs*. Additionally, the correct *BN* means the same *BN* selected by the CBNS and DBRA (or TBRA). Then, we separately experiment the DBRA and TBRA to obtain the individual values of total number of *BNs*. In our simulations, when node degree is 7, both of the DBRA and TBRA have low accuracy ratio as small communication range causes insufficient network connectivity. Especially in TBRA, some *extremal nodes* are selected to be *BNs* and exist around the

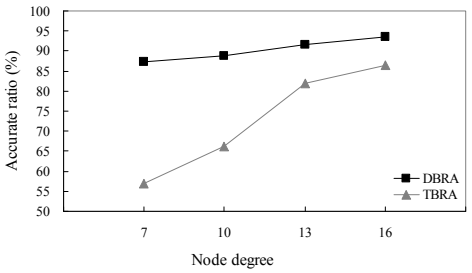


Figure 7. Percentage of accurate ratio for different node degree.

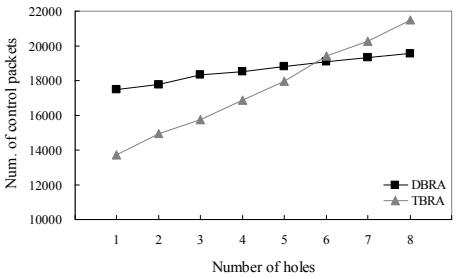


Figure 8. Number of control packets for different number of holes.

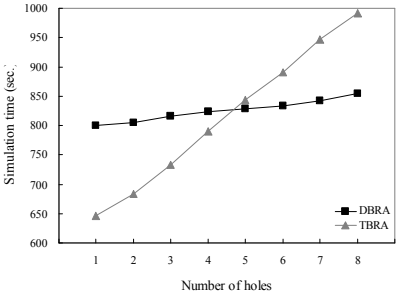


Figure 9. Simulation time for different number of holes.

sparse regions due to insufficient connectivity. However, these sparse regions may not be located at the border area of the hole or network. Therefore, these mistaken *extremal nodes* affect the algorithm to find out the correct *BNs*. Contrarily, our DBRA always selects the *BNs* on the border area of holes or connectivity. Thus, our DBRA can have higher accuracy ratio than TBRA even in low node degree.

B. Effect of number of holes on control packet overhead

In this experiment, as shown in Fig. 8, most of simulation parameters adopt the initial values, but the node degree and communication range are fixed to 7 and 13m, separately. The number of holes is also varied from 1 to 8 with interval of 1 and the radius of each hole is 50m. When constructing the *VHL*, several nodes have to help 7 *LNs* to rebroadcast *LN_Create* packets 7 times. Therefore, the DBRA always has higher control packet overhead due to disseminating fundamental information throughout each node. Contrarily, the required flooding in TBRA over the entire network is done only in two phases, building a shortest path and finding *extremal nodes*. Therefore, when there is only one hole within the sensing field, the essential control packet overhead of TBRA is lower than that of DBRA. However, in TBRA, the more holes within the sensing filed, the more nodes will participate in the phases of refining the coarse inner boundary and restoring the inner boundary. Therefore, the control packet overhead of TBRA is directly proportional to and also rapidly increases with the number of holes. In DBRA, there are only few nodes enclose each hole. Therefore, the number of control packets required to select *BNs* for each hole is also few. Although the number of holes is increased, the control packet overhead of DBRA will not be increased rapidly as compared to the TBRA. Thus, the control packet overhead of TBRA will exceed that of DBRA when the number of holes is more than 6 holes.

C. Effect of number of holes on simulation time

We utilize the same simulation parameters of section IV-B in this experiment. Similar to the pervious experiment, the simulation time is directly proportional to the number of holes especially to the TBRA. In the beginning, the DBRA costs lots of time to construct the *VHL* and *CBC*. Therefore, TBRA has smaller simulation time than DBRA. However, in TBRA, the more holes within the sensing filed, the more nodes participate in the phases of refining the coarse inner boundary and restoring the inner boundary. In DBRA, only nodes located on the border area of the holes and sensing field are required to select *BNs*. In addition, the procedure of selecting *BNs* in each hole can be executed simultaneously. The simulation time in DBRA only increases a little with different number of holes. Similar to the previous experiment, as the number of holes is increased and exceeds a certain threshold, the simulation time of TBRA will more than that of our DBRA. In this experiment, the threshold is 5 holes, as shown in Fig. 9.

V. CONCLUSION

In this paper, we have proposed a distributed protocol to find the boundary nodes enclosing the holes and the frontier of the network by utilizing only connectivity information and without any location information. We develop four phases in our protocol. They are the closure nodes selection phase, coarse boundary cycles identification phase, exact boundary nodes finding phase and maintenance of boundaries phase. Simulation results show that our protocol has higher accurate ratio on selecting correct boundary nodes than previous work. Besides, our protocol has less control message overhead and simulation time than previous work when number of holes are larger than 6.

REFERENCES

[1] K. Bi, K. Tu, N. Gu, W. Dong, and X. Liu, "Topological Hole Detection in Sensor Networks with Cooperative Neighbors," in *Proc. of International Conference on Systems and Networks Communications*, pp. 31-31, Polynesia, Oct. 2006.

[2] Q. Fang, J. Gao, and L. J. Guibas, "Locating and Bypassing Routing Holes in Sensor Networks," *Journal of Mobile Networks and Applications*, Vol. 11, No. 2, pp. 187-200, March 2006.

[3] Q. Fang, J. Gao, L. J. Guibas, V. de Silva, and L. Zhang, "Glider: Gradient Landmark-based Distributed Routing for Sensor Networks," in *Proc. of* INFOCOM, Vol. 1, pp. 339-350, USA, March 2005.

[4] S. P. Fekete, M. Kaufmann, A. Kröller, and N. Lehmann, "A New Approach for Boundary Recognition in Geometric Sensor Networks," in *Proc. of 17th Canadian Conference on Computational Geometry*, pp. 82-85, Canada, Aug. 2005.

[5] S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann, "Neighborhood-based Topology Recognition in Sensor Networks," in *Proc. of 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks,* Vol. 3121, pp. 123-136, Finland, July 2004.

[6] S. Funke and C. Klein, "Hole Detection or: "How much Geometry hides in Connectivity?" in *Proc. of ACM Annual Symposium on Computational Geometry*, pp. 377-385, USA, June 2006.

[7] R. Ghrist and A. Muhammad, "Coverage and Hole-detection in Sensor Networks via Homology," in *Proc. of 4th International Symposium on Information Processing in Sensor Networks*, pp. 254-260, USA, April 2005.

[8] P. K. Sahoo, K.–Y. Hsieh, and J.-P. Sheu, "Boundary Node Selection and Target Detection in Wireless Sensor Network," in *Proc. of the IFIP International Conference on Wireless and Optical Communications Networks*, Singapore, July 2007.

[9] O. Saukh, R. Sauter, M. Gauger, P. J. MarrÓn, and K. Rothermel, "On Boundary Recognition without Location Information in Wireless Sensor Networks," in *Proc. of 7th International Symposium on Information Processing in Sensor Networks*, pp.207-218, USA , April 2008.

[10] Y. Wang, J. Gao, and J. S. B. Mitchell, "Boundary Recognition in Sensor Networks by Topological Methods," in *Proc. of MobiCom*, pp. 122-133, USA, Sept. 2006.