Detection of Failed Boundary Nodes in Wireless Sensor Networks

Farid Lalem*, Ahcène Bounceur*, Rahim Kacimi‡, and Reinhardt Euler*
*Lab-STICC UMR CNRS 6285 - Université de Bretagne Occidentale
20, Avenue Victor Le Gorgeu, 29238, Brest, France
‡University of Toulouse
IRIT-UPS, 118 route de Narbonne, 31062 Toulouse, France
Email: Farid.Lalem@univ-brest.fr

***Abstract*—Wireless Sensor Networks (WSNs) plays an important role for monitoring strategic and dangerous hi security sites. Failure of some sensor nodes can lead to a failure of hole system which causes losses such as economical, equipment damage and even risks for human lives. Moreover, failures are unavoidable in WSNs due to hardware constraints, hostile environment, unattended deployment and limited resources. This paper proposes a fully distributed approach, called Boundary Node Failure Detection (BNFD), for an efficient boundary control based on the determination of the WSN boundary. This one is determined using an algorithm which has the property to determine in each iteration the one-hop neighbor of the current boundary sensor. Hence, each boundary sensor knows its direct next boundary neighbor and can communicate with it in order to periodically test its presence. When a situation of failure is detected, a network restructuring will be launched to find a new boundary and an alarm will be triggered. The proposed approach has been implemented and simulated with the Castalia simulator. The simulation results show that the proposed method is energy efficient.**

***Index Terms*—Failure Node Detection, Frontier, Wireless Sensor Network, Reliability.**

I. Introduction

Applications dedicated to the border surveillance of strategic and dangerous sites (e.g., oil or nuclear sites, frontiers of a country, etc.) are failure sensitive and require less energy consumption. In such applications, the detected event must be notified to the sink urgently, reliably and with low energy consumption.

Wireless Sensor Networks have been emerging in the last decade as a powerful tool for connecting the physical and digital world [1]. Sensor networks often operate in potentially hostile and harsh environments; most of the applications are critical missions such as battlefield surveillance. For instance, critical terrains can be rapidly covered with sensor networks and closely overseen for the activities of the opposing forces. As the operations evolve and new operational plans are prepared, new sensor networks can be deployed anytime for battlefield surveillance [2].

Moreover, WSNs have enabled several advanced monitoring and control applications in environmental, bio-medical, and several other domains. Boundary detection, as a fundamental technique for such applications, has become crucial for the functionality of WSNs [3].

For several years, energy-efficiency was the major design criteria for many proposed WSN solutions. However, there are still important efforts to accomplish especially with respect to the detection of faulty nodes.

In [4], the authors proposed MANNA, a system for fault detection in the network. Every node checks its energy level and sends a message to managers or agents. When there is a state change, the manager which is located at the external side of the WSN performs a centralized fault detection based on the analysis of gathered WSN data. However, the communication cost is too expensive due to the communication between nodes and the managers.

In [5] and [6], the authors proposed an algorithm to estimate the time to failure of a battery by analyzing the battery discharge curve and the current discharge rate.

In the FleGSens project [7], a wireless sensor network for the surveillance of critical areas and properties is developed which incorporates mechanisms to ensure information security. The intended prototype consists of 200 sensor nodes for monitoring a $500m$ long land strip. The system is focused on ensuring the integrity and authenticity of generated alarms as well as the availability in the presence of an attacker which may use an optimal number of sensor nodes. Two protocols were developed and presented: a tracking protocol to provide secure detection of trespasses within the monitored area, and a protocol for a secure detection of faulty nodes which ensures the integrity of the sensor network and avoids any breach in the coverage. This is done by selecting a random number of nodes called buddies to listen to other nodes' heartbeats.

In [8], a method to detect the sensor node failure or malfunctioning with the help of confidence factors is presented. The confidence factor of a roundtrip path in the network is estimated by using the round trip delay (RTD) which is the time required for a signal to travel from a specific source node along a path containing other nodes and back again. Confidence factors of all round trip paths are stored in look-up tables. Then, by analyzing the status of the confidence factor of all paths in the look-up tables, faulty sensor nodes are detected easily.

The technique presented in [9] is based on calculating periodically the throughput of a rectangular zone containing a certain number of sensor nodes. The calculated throughput will

be compared to a predefined threshold. If it is less than this threshold, the zone will then be divided into quadrants. The throughput of each quadrant will be calculated and compared to another threshold. If the throughput of a quadrant is less than its corresponding threshold, it will be divided again into another four quadrants. This process will be repeated until a quadrant that contains only one sensor node is reached. The node enclosed by that quadrant can be identified as a suspect node based on its throughput which is low. After that, the sleeping nodes around and near to the suspect one will be waked up in order to test the suspect node.

In [10], a tool called *Sympathy* is developed for detecting and debugging failures in sensor networks. Sympathy calculates metrics that allow to detect failures efficiently like statistic packets generated by nodes and transmitted to the sink. It includes an algorithm that root-causes failures and localizes their sources in order to reduce overall failure notifications and to point the user to a small number of probable causes. Sympathy detects a failure and triggers localization when a node generates less monitored traffic than expected.

In this paper, a new method for boundary failure detection is presented. After deployment, the boundary nodes will be determined using the algorithm presented in [11], where in each iteration the next boundary node n_{i+1} is determined by the node that has the minimum angle $\varphi_{min}(n_{i-1}, n_i, n_{i+1})$ formed by the edges (n_i, n_{i-1}) and (n_i, n_{i+1}) where n_i is the boundary node found in iteration i and n_{i-1} is the boundary node found in the previous iteration $i-1$.

The reason of choosing this algorithm for this work is based on the characteristic that each boundary node knows its two boundary neighbors. In fact, once the boundary nodes are determined, each such node will periodically send a message to its next neighbor, which should respond. If a response is not received, a situation of failure will be triggered and a network restructuring will be carried out to find a new boundary. The proposed approach requires a limited number of simple local computations and it needs the information on one-hop neighbors. The boundary nodes act as a sentinel with high duty cycle while the nodes that are the neighbors of the boundary nodes can have a low duty cycle. The other nodes are used only to replace the faulty boundary nodes. We note that our approach can be used to enclose a specific region such as frontier nodes covering a pollution or volcanic area or to detect an event such as a fire in a forest, etc.

The remainder of the paper is organized as follows. Section II introduces the boundary determination (BD) algorithm. In Section III, the proposed method of detecting the failure nodes is described. Simulation results and performance analysis are detailed in Section IV. Finally, Section V concludes the paper.

II. Boundary Determination (BD) Algorithm

In this section we will present the main steps of the algorithm to determine the boundary nodes in a WSN. This algorithm is the distributed version of the algorithm presented in [11]. First, we introduce some definitions and primitives. Then we present our algorithm.

A. Definitions and primitives

We assume that any two sensor nodes can directly exchange messages if their Euclidean distance is not greater than their communication range R_c and that a planar area can be covered by a sensor node if their Euclidean distance is not greater than the sensing range R_s. Consequently a WSN can be modeled as an undirected graph $G = (S, E)$, where $S = \{s_1, s_2, ..., s_n\}$ is the set of sensor nodes, $n = \mid S \mid$ is the total number of the sensor nodes and E is the set of the communication links. The link between two nodes can be defined as follows:

$$e_{ij} = \begin{cases} 1 & \text{if the node } s_i \text{ communicates with } s_j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The neighbor nodes $N(s_i)$ of a given node s_i are the nodes that communicate with it:

$$N(s_i) = \{s_j / e_{ij} = 1, j = 1, ..., n \text{ and } j \neq i\} \quad (2)$$

The location of each sensor node s_i is denoted by (x_i, y_i). For a better understanding of the proposed algorithm, we define in Table I some primitives and their definitions. We have chosen to make the following assumptions:

1) *Identical nodes:* All nodes are homogeneous in terms of capacity of computation, memory size and radio range. For the purpose of distinction, each node is attributed a unique identifier id.
2) *Localization:* Every node knows its location in space in term of (x, y) coordinates (we have possibly to use the GPS to get node coordinates) and each pair of coordinates (x_i, y_i) is allocated only once to every node s_i, which means that each node has a distinguished location in the network and also knows the location of the first node (a node with the smallest x-coordinate).
3) *Mobility:* In the beginning, the sensor nodes are assumed to be static during the boundary detection.
4) *Nature of the nodes:* Let us consider the set S of all sensor nodes and let B denote the set of the boundary nodes, where $B \subseteq S$, $|S| = n$ and $|B| = h$. In the following we will define $s_i^b, i = 1, ..., h$ as a boundary node, where s_i^b is designated by the previous boundary node s_{i-1}^b. The next boundary node s_{i+1}^b is chosen by s_i^b from a set of its neighbor nodes $N(s_i^b)$ as the node that forms the minimum angle with s_i^b and s_{i-1}^b.

B. The proposed algorithm

The boundary determination algorithm presented above starts from the node s_0^b having the smallest x-coordinate. The main steps of this algorithm, executed by each sensor node, can be described as follows:

- *Step 1*: This step is not executed by s_0^b. The boundary node $s_i^b, i > 0$, receives a packet from the previous boundary node s_{i-1}^b and updates its boundary node table t_i and the coordinates of the previous boundary node $coordP_i$;

TABLE I
PRIMITIVES AND THEIR DEFINITIONS

Primitive name	Definition
s_i	The i^{th} sensor node of S
s_i^b	The i^{th} boundary node of B
id_i	The identifier of a sensor node
s_0^b	The boundary node that starts the algorithm
$N(s_i)$	The set of neighbors of the node s_i
$coordC_i$	The coordinates of the current sensor node s_i
$coordP_i$	The coordinates of the sensor node previous to the current boundary node s_i
$coordN_j$, $j \in N(s_i)$	The list of the coordinates of the sensor node following the current boundary node s_i
t_i	The table of boundary nodes of the sensor node s_i
AC	Ask for a coordinates packet
CS	Sending of coordinate packet
SN	Select Node packet

- *Step 2*: s_i^b sends a broadcast AC packet to its neighbors $N(s_i^b)$ to ask for their coordinates. This is done by sending a packet formed by its identifier id_i and the AC primitive: $[id_i|AC]$;
- *Step 3*: Each neighbor node $s_j \in N(s_i^b)$, which receives the AC packet, sends a CS packet to the boundary node s_i^b with a CS packet containing its identifier, its CS primitive and its coordinates: $[id_j|CS|coordC_j]$;
- *Step 4*: The boundary node s_i^b will calculate the angle formed by the previous boundary node s_{i-1}^b with itself and with each one of its neighbor nodes s_j. This is done using the coordinates $coordP_i$, $coordC_i$ and $coordN_j$ in order to calculate the angle φ_{min}^i as follows:

$$\varphi_{min}^i = \underset{j \in \{k/s_k \in N(s_i^b)\}}{argmin} \{\varphi(coordP_i, coordC_i, coordN_j)\}$$

  Let us consider s_j^* the neighbor node that forms the minimum angle φ_{min}^i. This node will then be considered as the next boundary node of s_i^b. Therefore, $s_{i+1}^b = s_j^*$. Note, that especially for the starting node s_0^b, its previous boundary node is assumed as a fictive node with an x-coordinate smaller than the x-coordinate of s_0^b and the same y-coordinate.
- *Step 5*: The boundary node s_i^b will send an SN packet to the new boundary node s_{i+1}^b with the updated table of the sensor nodes $t_{i+1} = \{t_i\} \cup \{s_i^b\}$, where $t_0 = \emptyset$. This is done by sending the packet formed by its identifier id_i, the SN primitive, its coordinates and the table of sensor nodes t_{i+1}:$[id_i|SN|coordC_i|t_{i+1}]$;
- *Step 6*: The next boundary node s_{i+1}^b will perform the same procedure from *Step 1* on. The procedure stops as soon as the next boundary node is equal to the starting node s_0^b.

To explain more clearly how this algorithm works, we will use the example of Figure 1 which represents a graph of a WSN with eight sensor nodes. Let us consider the set $S = \{S1, S2, ..., S8\}$ of these nodes and the set B of the boundary nodes which is initially empty.

First (*Step 2*), we start from the node having the smallest x-coordinate, which is $s_0^b = S1$. This node will be considered as the first boundary node. Then, the boundary set is updated to $B = \{S1\}$. The node $S1$ sends a broadcast AC packet to its neighbors $N(S1) = \{S2, S3, S4, S7\}$ to ask for their coordinates. This is done by sending a packet formed by its identifier 1 and the AC primitive: $[1|AC]$ (cf. Figure 1(a)).

Next (*Step 3*), each neighbor node $S2, S3, S4$ and $S7$ which receives the AC packet sends a CS packet to the boundary node $S1$ with a CS packet containing its identifier, the CS primitive and its coordinates (cf. Figure 1(b)): $[S2|CS|2, 4]$, $[S3|CS|4, 8]$, $[S4|CS|7, 7]$ and $[S7|CS|3, 3]$;

Next (*Step 4*), the boundary node $S1$ will calculate the angle formed by the fictive node $S1'$, with itself and with each one of its neighbor nodes $S2, S3, S4$ and $S7$. This is done using the coordinates $coordP_1 = (0, 5)$, $coordC_1 = (1, 5)$ and $coordN_2 = (2, 4)$, $coordN_3 = (4, 8)$, $coordN_4 = (7, 7)$ and $coordN_7 = (3, 3)$ in order to calculate the angle φ_{min}^1 as follows:

$$\begin{aligned} \varphi_{min}^1 &= \underset{j \in \{2,3,4,7\}}{argmin} \{\varphi(coordP_1, coordC_1, coordN_j)\} \\ &= \varphi(coordP_1, coordC_1, coordN_3) \end{aligned}$$

which means that the next boundary node is $S3$. This situation is illustrated by Figure 1(c).

Then we run *Step 5*. As shown by Figure 1(d) the boundary node $S1$ will send an SN packet to the new boundary node $S3$ with the updated table of the sensor nodes $t_1 = T_3 = \{t_0\} \cup \{S1\}$, where $t_0 = \emptyset$ (cf. Figure 1(e)). This is done by sending the packet formed by its identifier 1, the SN primitive, its coordinates and the table of sensor nodes t_1:$[1|SN|1, 5|T_3]$.

After that (*Step 6*), the next boundary node $s_1^b = S3$ will perform the same procedure from *Step 1*. The boundary node $S3$ receives a message from the previous boundary node $S1$ and updates its boundary node table $t_1 = T_3$ and the coordinates of the previous boundary node $coordP_1$ and so on. The procedure stops when the next boundary node is equal to the starting node $S1$. Figure 1(f) shows the final iteration of the boundary determination algorithm BD.

III. FAILURE NODE DETECTION

In this section, we will present the proposed method of Boundary Node Failure Detection (BNFD) using the BD algorithm presented above. The main steps of this method are presented by the flowchart of Figure 3. First, the BD algorithm will be run on a given WSN. As an example, Figure 2 shows a WSN with 11 sensor nodes and its boundary nodes (i.e., $S1$, $S2$, $S3$, $S4$, $S5$, $S6$ and $S7$). During the execution of this algorithm and in each iteration, each boundary node stores locally the id of its direct next boundary neighbor, which represents the node selected to be the next boundary node. Once the border is completely determined, each boundary node S_i starts sending periodically a testing message A "*Are you there?*" in order to test the presence of its boundary neighbor as it is shown by Figure 2(a). Note that in this figure, the A-messages are sent by each node at the same time. However,

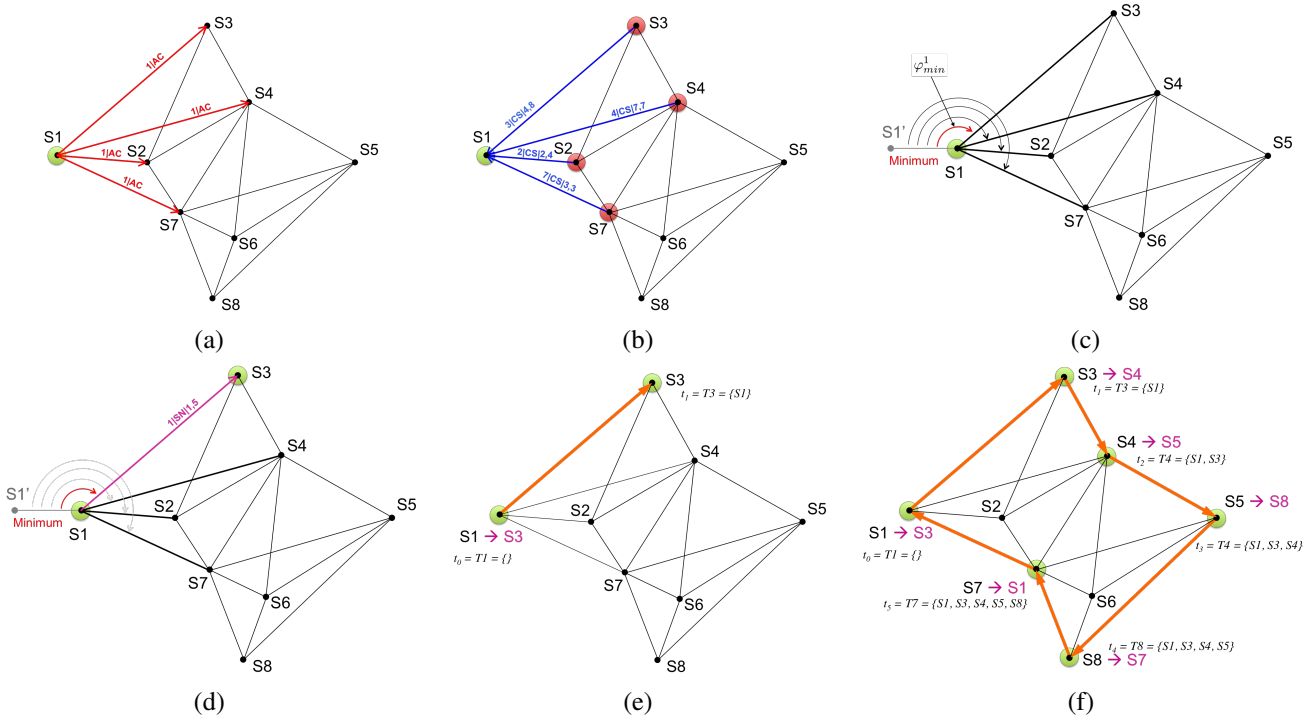

Fig. 1. Boundary Determination (BD) algorithm example.

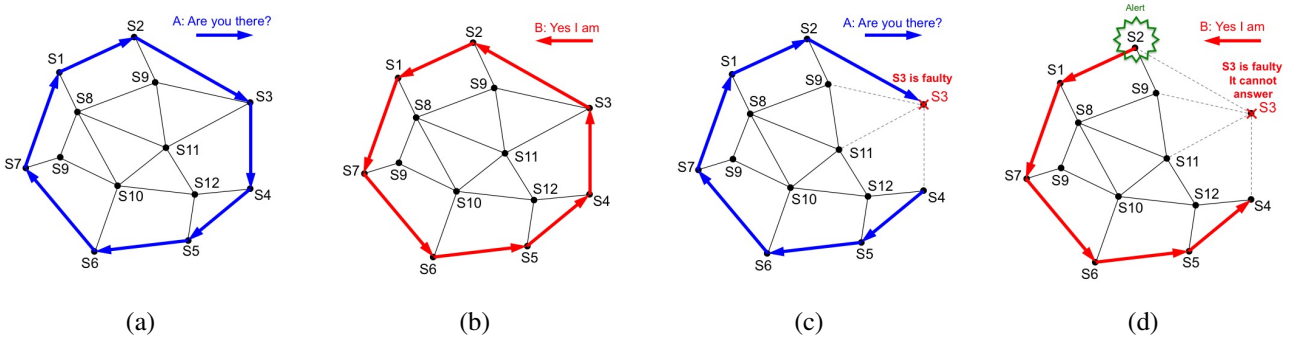

Fig. 2. Border Monitoring Algorithm.

in reality, these messages will be sent sequentially in order to avoid collisions, just as described by the flowchart of Figure 3. Once an A-message is sent, the transmitter will wait, for a limited time, for an answer B-"*Yes, I am*"-message from its next boundary neighbor, which is represented by red arrows in Figures 2(b) and (d). If the neighbor is failing then it cannot answer. Therefore, once the limited time is reached, and no answer is received any more from the neighbor (see Figure 2(c)), then the next neighbor will be declared as a failure node (see Figure 2(d)). In this case, the BD algorithm will be re-executed and an alarm will be triggered.

Algorithm 1 shows the pseudocode of the presence testing procedure, where each node sends the message 'A' to its next border neighbor nid (cf. lines 2 and 3) once the boundary nodes are determined. Then, each node will wait a given time period t_1 for an answer message 'B' from its next neighbor

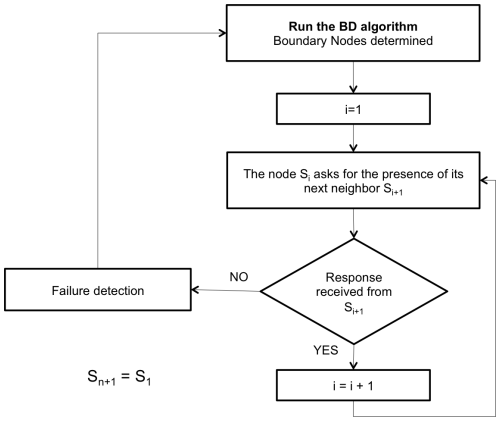

Fig. 3. The BNFD approach.

(cf. line 4). If the node does not receive any message during this period than its neighbor will be declared as a failure node (cf. lines 5 and 6). In this case, the BD algorithm will be re-executed again in order to find the new border node and an alarm will be triggered. Otherwise, if a node receives the message 'A' from its previous neighbor pid than it has to answer by the message 'B' in order to confirm that it is not a failure node (cf. lines 10, 11 and 12). Receiving the message 'B' from its next neighbor nid means that the node nid is not faulty (cf. lines 14 and 15). This algorithm is repeated every t_2 seconds.

Algorithm 1 Presence Testing algorithm

```
1:  while (true) do
2:      p = cid+"|"+"A";
3:      send(p, nid);
4:      n = read(t1);
5:      if (n=="") then
6:          A failure is detected: run the BD algorithm;
7:          break();
8:      else
9:          type = read();
10:         if (type==A) then
11:             p = cid+"|"+"B";
12:             send(p, pid);
13:         end if
14:         if (type==B) then
15:             No failure detected: do nothing;
16:         end if
17:     end if
18:     wait(t2);
19: end while
```

IV. Performance Evaluation

The objective of this section is to evaluate the robustness of the proposed method and its energy efficiency. To do this, we have used the Castalia-3.3 framework [12] which is based on the OMNeT++ 4.6 simulator. It provides advanced channel and radio models and contains a realistic power consumption profiling.

A. The case study

In the considered scenario, N sensor nodes are deployed uniformly so that to obtain the maximum coverage of the considered region. The transmission power level of all the sensor nodes is fixed to -5 dBm, the radio hardware model used by all the sensor nodes is the TelosB equiped with the CC2420 radio modules based on the 802.15.4 standard.

Figure 4 shows the deployment scenario for $N = 9$ sensor nodes in a rectangular area of 30×30 meters. By applying the BD algorithm we have obtained the following boundary nodes: $B = \{N_0, N_3, N_6, N_7, N_8, N_5, N_2, N_1\}$.

Figure 5 and Figure 6 show the energy consumed by each boundary node using the BD and the BNFD algorithm, respectively. The energy consumed by each node is given by the sum of the consumption in transmission, reception, listening and sleeping modes. As we can see, the BD algorithm consumes an average energy of 0.368 Joules, which represents

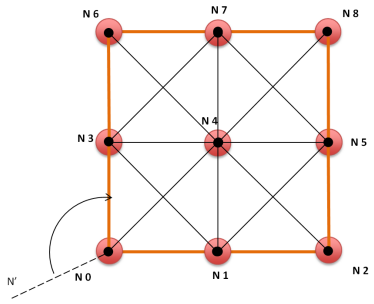

Fig. 4. Network Deployment and Connectivity.

0.002% of the initial battery capacity used in the Castalia framework and which is equal to 18720 Joules (i.e., a capacity of two AA batteries). Therefore, it can be executed around 50869 times until the battery's depletion. If we assume that one execution of the algorithm takes around 30 minutes, then on one day this algorithm can be executed 48 times. Thus, one battery can be used for $50869/48 \approx 1060$ days (i.e., 3 years).

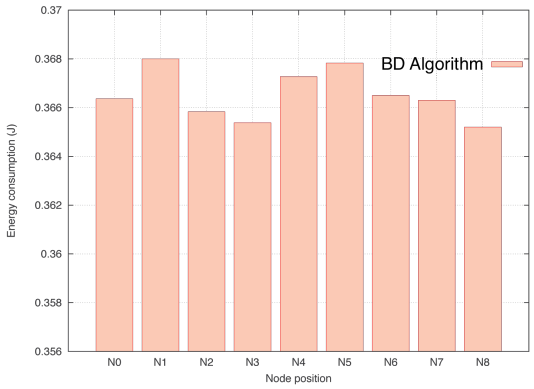

Fig. 5. Energy Consumption per Node (BD Algorithm).

In the same way, for the case of the BNFD algorithm, as shown by Figure 6, each node consumes an average energy of 0.18 Joules, which represents 0.001% from the initial battery capacity. Therefore, it can be executed $104 \cdot 10^3$ times until the battery's depletion. If we assume that one execution of the algorithm takes around 10 minutes, then on one day this algorithm can be executed 144 times. Thus, one battery can be used for $104000/144 \approx 722$ days (i.e., 2 years).

Therefore, as shown by Figure 7, the total mean energy consumed by both BD and BNFD algorithms is equal to 0.548 Joules, which represents 0.003% of the total energy of one battery. Therefore, it can be executed 18720 times until the battery's depletion. Based on assumptions given above, i.e., one execution of the algorithm takes around $40 = (30 + 10)$ minutes, then on one day these two algorithms can be executed 36 times. Thus, one battery can be used for $18720/36 \approx 520$ days (i.e., 1.5 years).

To study the impact of the size of the network on the energy consumption, we vary the number of the sensor nodes from 9

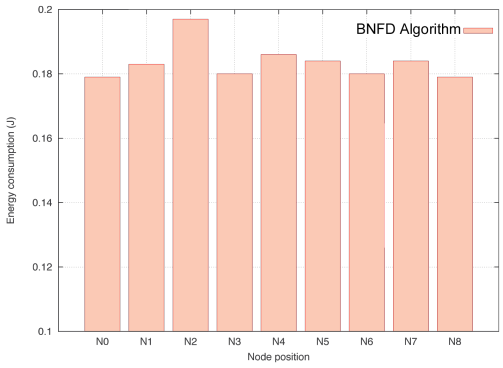

Fig. 6. Energy Consumption per Node (BNFD Algorithm).

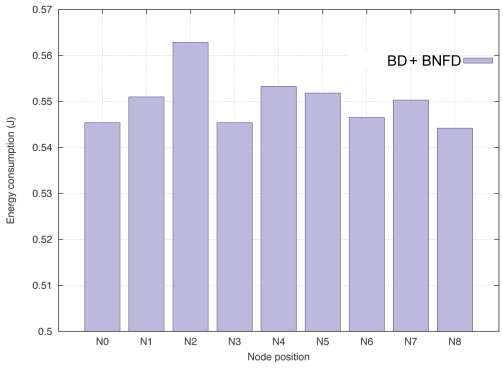

Fig. 7. Energy Consumption per Node (BD+BNFD Algorithms).

to 1000 and measure the average of the energy consumption of the BD and BNFD algorithms. Figure 8 plots the energy consumption under different network sizes. It shows that when the number of nodes deployed in the field of interest increases, the average of energy consumption also increases. In fact, this is an expected result because all nodes are involved in the communication process. Indeed, in our case we are interested in monitoring a sensitive site and in detecting failure nodes on the boundary. Thus, if we deploy sensor nodes only around the boundary, the energy consumption decreases due to the reduction by the number of nodes that are inside the network.

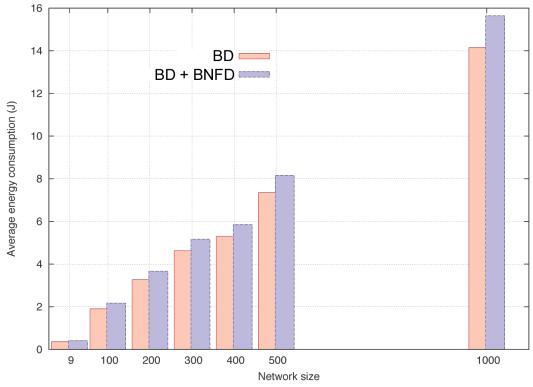

Fig. 8. Energy consumption (BD and BD+BNFD Algorithms).

Finally, it is possible to improve the energy consumption using the proposed approach by executing the BD algorithm only when a node fails instead of executing it periodically as mentioned above.

V. Conclusion

In this paper we have proposed a distributed approach to detect failure nodes in Wireless Sensor Networks (WSN). Our new approach finds the WSN boundary nodes and monitors a sensitive area with the nodes situated on the boundary. Through extensive simulations with the Castalia simulator, we have evaluated the performance of our scheme under various conditions and we could show its energy efficiency. In this approach, the number of exchanged messages to identify the faulty sensors is small and a substantial amount of energy can be saved by the sensor nodes. Moreover, the proposed approach outperforms previous ones by providing high detection accuracy. As future work we intend to analyze our approach in terms of resilience to changes in the set of failures by injecting faults into the network, to study the network performance in this case and how to immediately deliver failure notifications through the network.

References

[1] E. Felemban, "Advanced border intrusion detection and surveillance using wireless sensor network technology," *Int. Journal of Communications, Network and System Sciences*, vol. 6, no. 5, pp. 251–259, 2013.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[3] F. Li, C. Zhang, J. Luo, S.-Q. Xin, and Y. He, "Lbdp: localized boundary detection and parametrization for 3-d sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 2, pp. 567–579, 2014.

[4] L. B. Ruiz, J. M. Nogueira, and A. A. Loureiro, "Manna: A management architecture for wireless sensor networks," *Communications Magazine, IEEE*, vol. 41, no. 2, pp. 116–125, 2003.

[5] D. Rakhamtov and S. Vrudhula, "Time-to-failure estimation for batteries in portable electronic systems," in *International symposium on Low power electronics and design*, pp. 88–91, ACM, 2001.

[6] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "A discrete-time battery model for high-level power estimation," in *Proceedings of the conference on Design, automation and test in Europe*, pp. 35–41, ACM, 2000.

[7] P. Rothenpieler, D. Krüger, D. Pfisterer, S. Fischer, D. Dudek, C. Haas, A. Kuntz, and M. Zitterbart, "Flegsens-secure area monitoring using wireless sensor networks," *Proceedings of the 4th Safety and Security Systems in Europe*, 2009.

[8] R. N. Duche and N. Sarwade, "Sensor node failure or malfunctioning detection in wireless sensor network," *ACEEE Int. J. Commun*, vol. 3, no. 1, pp. 57–61, 2012.

[9] A. A. Taleb, D. K. Pradhan, and T. Kocak, "A technique to identify and substitute faulty nodes in wireless sensor networks," in *Sensor Technologies and Applications, 2009. Third International Conference on*, pp. 346–351, IEEE, 2009.

[10] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pp. 255–267, ACM, 2005.

[11] A. Bounceur, R. Euler, A. Benzerbadj, F. Lalem, M. Saoudi, T. Kechadi, and M. Sevaux, "Finding the polygon hull in wireless sensor networks," in *EURO conference, July, 2015, Glasgow.*, 2015.

[12] G. Fortino, R. Greco, and A. Guerrieri, "Modeling and evaluation of the building management framework based on the castalia wsn simulator," in *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, pp. 668–674, IEEE, 2013.