FDRA: Fault Detection and Recovery Algorithm for Wireless Sensor Networks

Chafiq Titouna[1(✉)], Ado Adamou Abba Ari[2,3], and Hamouma Moumen[1]

[1] Department of Computer Science,
University of Batna 2, Batna, Algeria
c.titouna@univ-batna2.dz
[2] Department of Mathematics and Computer Science,
University of Maroua, Maroua, Cameroon
[3] LI-PaRAD Laboratory,
University of Versailles, Versailles, France

Abstract. Failures are inevitable in wireless sensor network, and it is important to detect and recover faulty nodes. In this paper, we present an algorithm to recover faulty nodes called Fault detection and Recovery Algorithm (FDRA). The performance evaluation is tested through simulation to evaluate some factors such as: Packet delivery ratio, control overhead, memory overhead and fault recovery delay. We compared our results with referenced algorithm: Fault Detection in Wireless Sensor Networks (FDWSN), and found that our FDRA performance outperforms that of FDWSN.

Keywords: Wireless sensor networks · Fault tolerance
Connectivity restoration

1 Introduction

A wireless sensor network (WSN) consists of a possibly large number of wireless devices able to take environmental measurements. Typical examples include temperature, light, sound, and humidity. These measurements are transmitted over a wireless channel to a base station (BS) that makes decisions based on these data. WSNs have infiltrated our daily life, such as medical monitoring [1], military surveillance [2, 3], vehicle monitoring [4], home automation monitoring [5], weather monitoring [6], building structures monitoring, and industrial plant monitoring [7–9]. Some of these WSN's applications were deployed in remote and hostile surroundings, and none can attend nodes in such environment. In addition, some nodes are failure due to energy depletion, hardware failure, communication link errors, intrusion by attackers and so on. These unattended nodes cannot be replaced or repaired. They may generate a faulty data and even cannot respond to any request. These problems may lead to network partition which decreases the cover ratio, reduces the availability of the WSN and even produces network partition. Therefore, WSN should possess a mechanism of fault tolerance. It can be defined as the ability of a system to deliver a desired level of functionality in the presence of faults [10]. Since the nodes are prone to failure, fault tolerance should

M. Younas et al. (Eds.): MobiWIS 2018, LNCS 10995, pp. 72–85, 2018.
https://doi.org/10.1007/978-3-319-97163-6_7

be seriously considered in many sensor network applications. Actually, extensive work has been done on fault tolerance and it has been one of the most important topics in WSNs [11, 12].

In this paper, we extend our preliminary work proposed in [13]. We improve the proposal algorithm which called: Fault Detection Scheme (FDS), and we present a novel one called Fault Detection and Recovery Algorithm (FDRA). The first one ensures only a detection of faulty nodes without any recovery. However, in the present scheme FDRA, we improve the previous work by ensuring this task.

The rest of the paper is organized as follows: Sect. 2 presents some related work. In Sect. 3, we describe our recovery algorithm and illustrate it through example. We provide in Sect. 4 performance results and in Sect. 5 we conclude the paper.

2 Related Works

Several works are proposed to detect and recover the faulty nodes in wireless sensor networks [14]. In [15], the authors proposed a detection technique to eliminate all erroneous sensed data generated by faulty nodes. Wang et al. [16] have proposed an approach based on cascaded movement to replace a faulty node by a nearby node, which in turn gets replaced with another and so on until reaching a redundant node. The authors in [17], proposed a few simple algorithms for achieving the baseline graph theoretic metric of tolerance to node failures, namely, biconnectivity. They formulated an optimization problem for the creation of a movement plan while minimizing the total distance moved by the robots. However in DARA [18], the main idea was to detect the failure of an actor and replace the failed actor in a cascaded manner. The previous work was enhanced in [19]. They use the connected dominating set (CDS) of the whole network in order to detect the cut-vertex node. After detecting these nodes, every node picks the appropriate neighbor to handle its failure in the case of failure in future. The objective is to choose a neighbor that may not partition the network again. In [20], the replacement of the failed node is done only by its direct neighbors. Akkaya et al. [21] presented the new distributed partition detection and recovery algorithm (PADRA, PADRA+) to handle the connectivity problem through detection of possible partitions after the failure of the cut-vertex node is observed in the network and restores the network connectivity through controlled relocation of the movable nodes. Younis et al. [22] proposed a localized distributed algorithm called recovery through inward motion (RIM) for the network partition recovery. The main idea is to move the entire neighbor node(s) towards inward direction of the failed node so that nodes can discover each other and recovery can take place. In [23], the authors presented an algorithm called Connectivity Restoration with Assured Fault Tolerance (CRAFT) to solve the problem of simultaneous sensor failures. This algorithm forms a backbone polygon around failure region. Then, CRAFT tries to fix partition problem by connection the disjoint paths to each other. A distributed fault detection algorithm for WSNs named FDWSN has been proposed in [24]. Every node discerns its own status in view of local comparisons of its sensed data with the data of neighboring nodes for q times to detect transient fault. The authors used a redundancy matrix to save all results of comparison. After, the status of the node is declared as good

if the sensed data are similar. Finally, each sensor node with a defined status will broadcast its status to its neighbors to facilitate them for determining their own status. This scheme can detect and isolate faulty nodes with high detection accuracy. Transient faults are also tolerated by using time redundancy.

Contrary to the ideas proposed in the above reviewed works in which movement of sensor nodes or exchanging Hello message are required, our proposal mainly based on stationary sensor nodes that do not need any mobility.

3 Fault Detection and Recovery Algorithm

In this section, we present our proposed technique, named Fault Detection and Recovery Algorithm (FDRA). We begin first by defining some assumptions. We then provide details on the mechanism used for recovering faulty nodes. Our algorithm operates in two phases: (a) First, a tree-forming and selection of sleeping nodes are executed. (b) Second, selection of recovering nodes to recover the faulty nodes and updating connectivity table.

3.1 System Assumptions

A WSN is typically consisting of a large number of nodes scattered over a region of interest to monitor a particular physical phenomenon. Some assumptions, complying with practical aspect, have to be considered in our algorithm. The first assumption is that all sensed data are forwarded from sources to a central node called Base Station (BS), where data processing occurs. The second is that, all nodes are stationary and its batteries cannot be recharged. We recognize that local processing may occur to reduce overall communication costs. The next assumption we make is that all nodes are homogeneous in terms of energy, communication and processing capabilities, and they are assigned a unique identifier (ID). Finally, we also assume that we do not have malicious attacks on the network.

3.2 Algorithm Design

Our algorithm consists of two phases. In the following sub-sections, we describe each phase in details.

Preliminary Phase

Tree-Forming and Creation of Connectivity Table Step

Tree-forming is an efficient and effective technique for recovery process. In this paper, we are interesting to BS-based trees where it uses the BS as the core of the network. This is optimum in WSN, since the BS does not suffer from battery depletion. In Fig. 1, the initial connectivity of the network is presented, while Fig. 2, describes a BS-based tree applied on the initial topology (Fig. 1). The process of creation of BS-based tree is analytically demonstrated in [25, 26]. It is clear that this transformation leads to a creation of connectivity table which presented in Table 1. Consider the scenario shown in

Fig. 2, the table of connectivity will be presented as in Table 1. The columns describe the levels of the tree and the rows represent the ID of sensor nodes belong to each level. We consider the same example, the level 1 (L = 1) contains the following IDs: 1, 2 and 3. These nodes belong to the level 1.

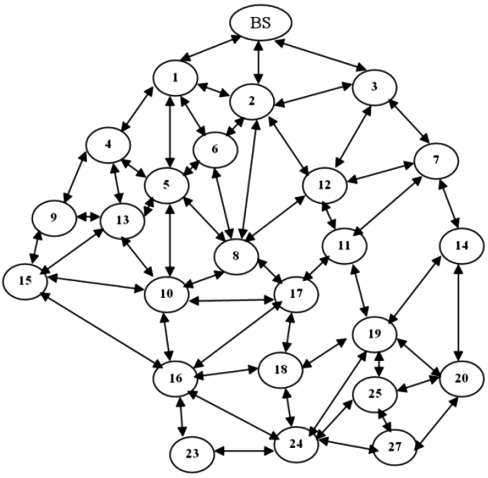

Fig. 1. Initial connectivity of the network.

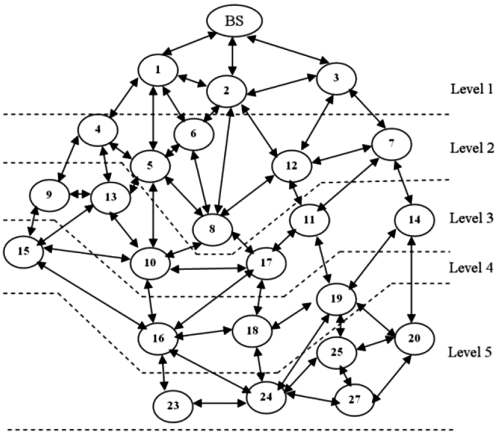

Fig. 2. Connectivity of the network after applying BS-based tree algorithm.

Selection of Sleeping Nodes Step

The BS chooses which nodes should be in sleep mode. The BS can take into account the energy remaining of nodes, the geographic position of nodes, total number of nodes presents in levels or the degree of connectivity of a node. In this paper, we are based on the degree of connectivity of nodes, i.e., the BS selects the node that has fewer neighbors. To start the selection of sleeping nodes, the BS broadcasts a *Req_nn* message to know

the number of neighbors of each node. When a node receives a *Req_nn* message, it will respond with *Resp_nn* message to inform the BS the number of neighbors in its transmission range. When the *BS* receives all *Resp_nn* messages, the selection of sleeping nodes is launched by selecting only nodes with number of neighbors is less than α threshold (α can be equal to the average number of each level). The BS sends a *Select_Sleep_msg* message to all selected nodes. A response *Resp_Sleep_msg* message, it will be sent by the selected nodes to indicate to the BS that they will switch to sleep mode. E.g., consider Fig. 1. The BS selects in the 1st level the node 3. The nodes with ID 4, 6 and 7 in the second level. The nodes 9, 11 and 14 in the third level. The nodes 15, 18 in the fourth level and nodes with ID 23 and 27 in last level. The IDs of selected nodes are presented in bold in the Table 1. Figure 4 represents a message exchanging required to select sleeping nodes in level 2.

Update Connectivity Table Step

This step allows updating the connectivity table. After a phase of detection of faulty nodes (see more details in [13]) the cluster-head transmitted to the BS a black list (BL) contains all the faulty nodes detected. The *BS* updates the connectivity table, so it will proceed to an elimination of all faulty nodes'ID from the table. E.g., consider the Table 1. We suppose that nodes 5, 12 and 16 are in BL. So they should be removed from the Table 1 (we just highlighted in Table 1). The next subsections describe the steps required for the recovery phase.

Table 1. Connectivity table.

L = 1	L = 2	L = 3	L = 4	L = 5
1	**4**	**9**	**15**	**23**
2	5	13	16	24
3	**6**	10	**18**	25
	8	17	19	**27**
	12	**11**	20	
	7	**14**		

Recovery Phase

Principle

FDRA recognizes two transition states for the nodes, active and sleeping. Initially all nodes in the network are in active state. This means that all radio's nodes are on until receiving a message like "*Select_Sleep_msg*". This message comes from BS to order preselected nodes to switch their radio off and move to sleep mode. Returns back to the active state happen when the BS selects a sleeping node to recover a faulty node. To do that, BS will send a wakeup message to these nodes. Therefore, the transition between the two states is ordered by the BS and is performed by sending messages.

Selection of Recovery Nodes Step

The process of selection of recovery nodes implies the reactivation of some sleeping nodes. The BS sends a *Wake_up_msg* message to all sleeping nodes of the same level of the faulty nodes. However, the sleeping nodes which are in level that does not contain

any faulty nodes, they will not receive this message. E.g., consider the previous example, in Table 1, the sleeping nodes which receive the *Wake_up_msg* are: The nodes 4, 6 and 7 of the 2nd level and the nodes 18 and 15 of the 4th level (i.e., The message concern two levels (2 and 4), because we supposed in previous subsection that there are only three faulty nodes in BL (5, 12 and 16) which they are located in 2nd and 4th level respectively).

At receiving the *Wake_up_msg* message, the sleeping nodes turn their radio on and send back a wakeup acknowledgement message (*Wake_up_ack*) to the BS to indicate its new state (Active state). The BS sends now a request (*Req_hop_reqr*) to know the number of hops required to reach the faulty nodes from the sleeping nodes of the same level. The sleeping nodes update their routing table and respond to the BS using a simple packet (*Resp_hop_reqr*). The structure of this packet is described in Fig. 3.

When all *Resp_hop_reqr* are received, the BS creates a Hop Required Table which helps it to choose the appropriate recovery node RN. E.g., the Table 2(a) and (b) summarize the number of hops required for the nodes 4, 6 and 7 to reach the faulty nodes 5 and 12.

srcid	desid	FN(1)_id	hops_nd	FN(2)_id	hops_nd	...	FN(n)_id	hops_nd

Fig. 3. Packet format.

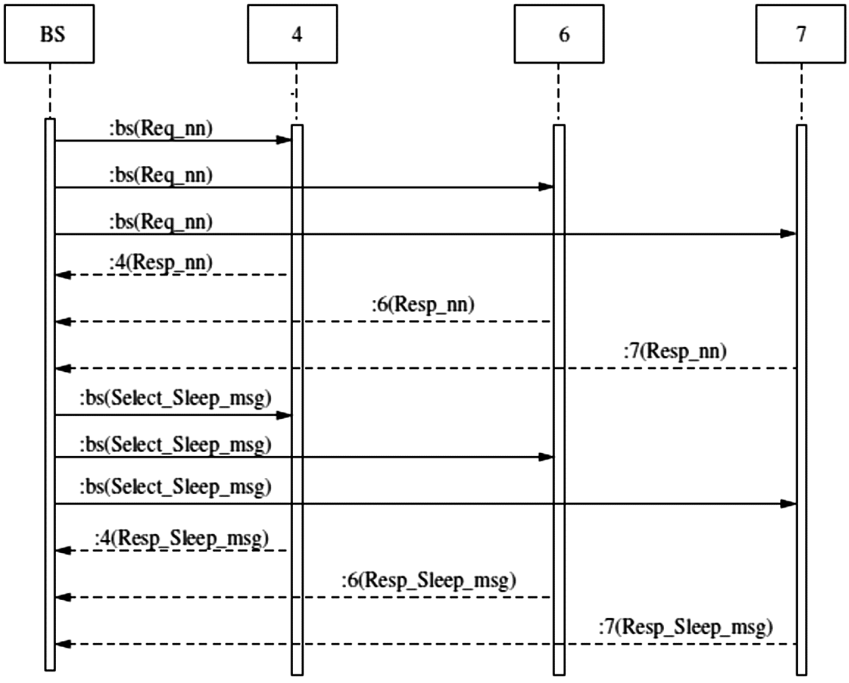

Fig. 4. Message exchanging between BS and selected sleeping node (level 2).

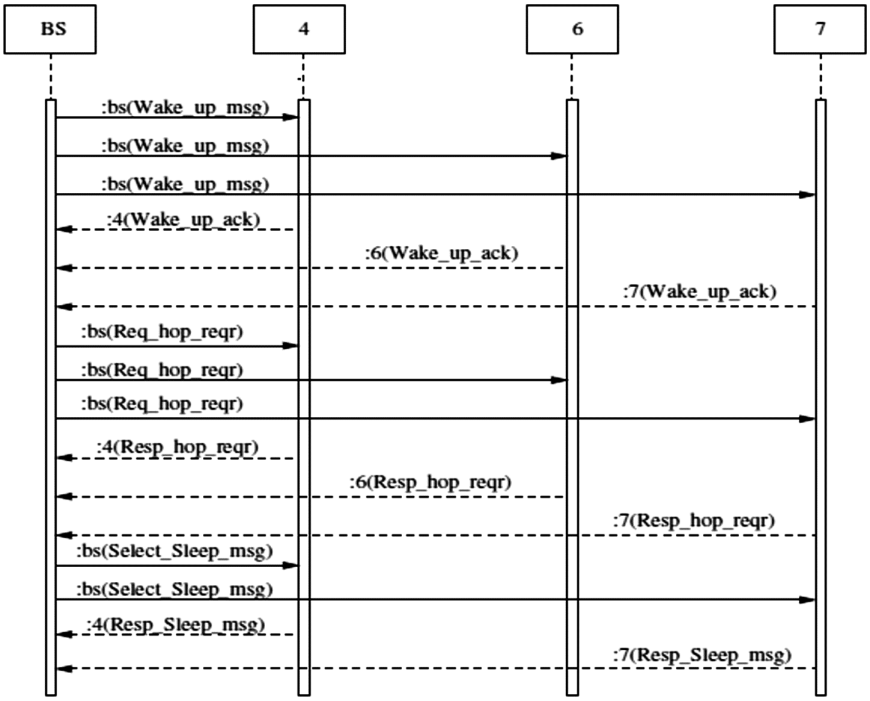

Fig. 5. Message exchanging required for selecting RN (level 2).

The BS chooses the one which requires fewer hops to reach the faulty nodes FN. E.g., consider the Table 2, the node with ID 6 need 2 hops to reach node 5 and 12. So, the BS selects it as RN. A *Select_Sleep_msg* message is sent to the not selected nodes (e.g., 4 and 7) to go back in sleep mode. A response *Resp_Sleep_msg* message, it will be sent by the selected nodes to the *BS*. Figure 5 shows the previous exchange messages.

Table 2. (a):Hop required table (FN = 5) (b)Hop required table (FN = 12).

(a)		(b)	
Sleeping node ID	Hop required	Sleeping node ID	Hop required
4	1	4	3
6	1	6	2
7	3	7	1

Update Connectivity Table Step

The BS should update the connectivity table after each recovery phase. All sleeping nodes that are selected for covering the erroneous nodes must be mentioned in the connectivity table. E.g., consider the example in previous section, the node with ID 6 will be considered as an active node.

4 Performance Evaluation

We have conducted several series of simulations using the TOSSIM simulator [27] in order to evaluate the performance of our proposed algorithm. For comparison purposes, our proposal FDRA and FDWSN protocol are evaluated under the following metrics: (1) the packet delivery ratio (PDR); (2) the control overhead (CO); (3) the memory overhead (MO); and (3) the fault recovery delay (FRD). The key simulation parameters are summarized in Table 3.

4.1 Analysis of Packet Delivery Ratio (PDR)

Packet delivery ratio is calculated as the number of packets received by the receiver divided by the number of packets sent by the sender. This metric characterizes the percentage of successful source data packet delivery; ideally, this should be 100%. Figure 6 shows the total number of data packets received by the BS over the number of nodes (with node transmission power (TP) set to 2 mW). From this figure, we see that the amount of data collected by the *BS* from each source is much more important with our FDRA algorithm then the FDWSN protocol. This can be explained by the fact that FDWSN uses transient phase for fault detection and it does not consider the other causes of node's failures. The Fig. 7 shows the improvement in packet delivery ratio when node transmission power is equal to 4 mW. We explain this improvement by the fact that the number of neighbors of recovery node increases when we increase the transmission power. However, for FDWSN, we noticed that there is a less in packet delivery ratio because FDWSN requires a set of iterations that consume battery power which increases faulty nodes (battery depletion).

4.2 Analysis of Control Overhead (CO)

Since, the detection and recovery of faulty node is an additional task in WSN. It requires extra control packets. This metric computes the additional control packets needed to perform the recovery process. In Fig. 8, we noticed that for both FDRA and FDWSN increase with the increase of the number of nodes.

Table 3. Simulation parameters.

Parameter	Value
Area size	1000×1000 ms
Number of nodes	20, 40, 60, 80, 100, 200
Transmission power	2mW, 4mW
Transmission channel	Wireless channel
Propagation model log	Normal path loss model
Data packet size	32 bytes
Bandwidth	200 Kilobytes/second
Radio layer	CC2420 radio layer
Queue size	50 packets

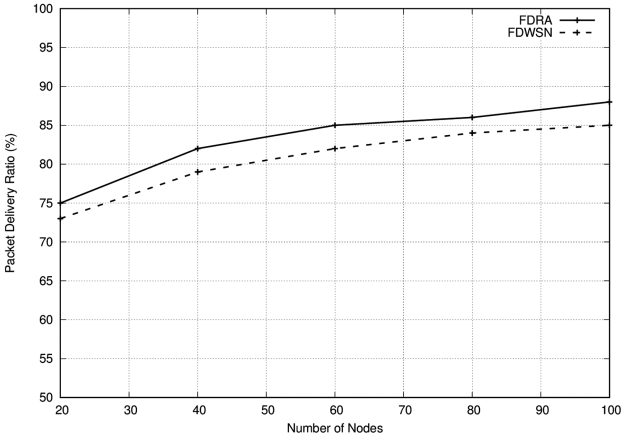

Fig. 6. PDR vs. number of nodes, TP = 2mW.

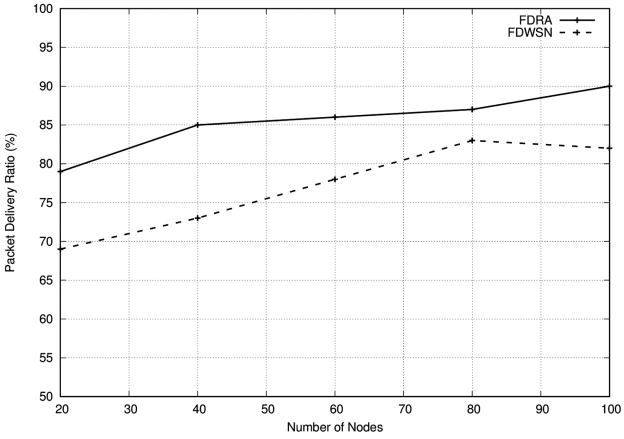

Fig. 7. PDR vs. number of nodes, TP = 4mW

More nodes require more control packets to achieve the recovery process. However, the FDWSN protocol uses more control packet comparing with our FDRA since the later does not need any iteration for the detection process. The FDWSN's detection faulty nodes process is very complicated and it is based on collecting neighbors information to detect the faulty nodes. When we increase the node transmission power to 4 mW (see Fig. 9), FDWSN needs more packet of control. However, our algorithm performs better with high transmission power because the recovery node can reach more nodes of the same level.

4.3 Analysis of Memory Overhead (MO)

This metric represents the average number of bytes needed to be stored in the memory of all nodes that are implied in recovery process. We compute for FDRA and FDWSN, the additional memory space needed to ensure the all process. We plot the result in Fig. 10. During the simulation, we notice a change in the quantity of bytes required for both algorithms. This instability in memory overhead is due to the mechanism deployed on nodes. We observe that the memory overhead in FDRA is less than in FDWSN because the later requires more memory to store transient fault matrix and other parameters. In Fig. 11 (node transmission power = 4 mW), we observe better results comparing with previous curves (transmission power = 2 mW) of two algorithms with FDRA less memory overhead.

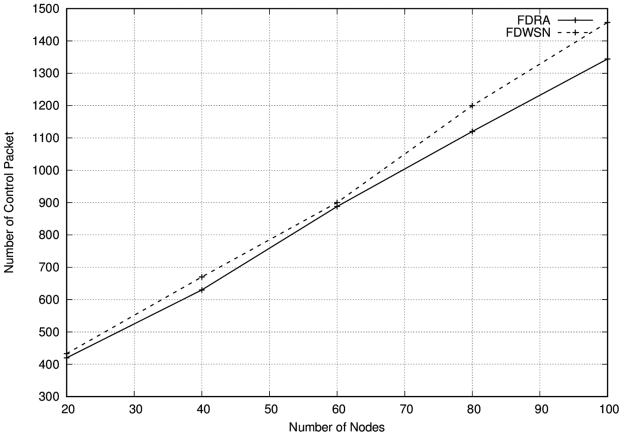

Fig. 8. CO vs. number of nodes, TP = 2mW.

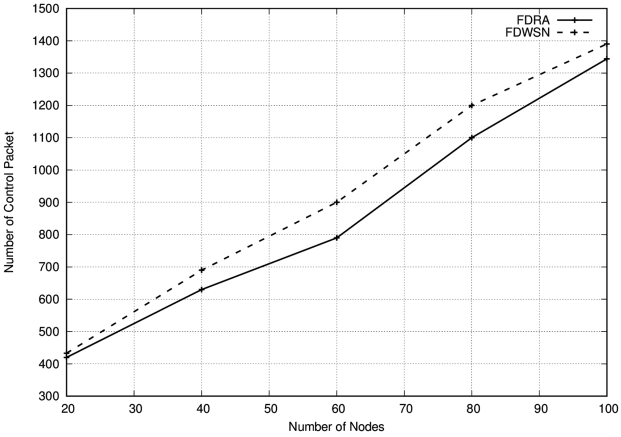

Fig. 9. CO vs. number of nodes, TP = 4mW.

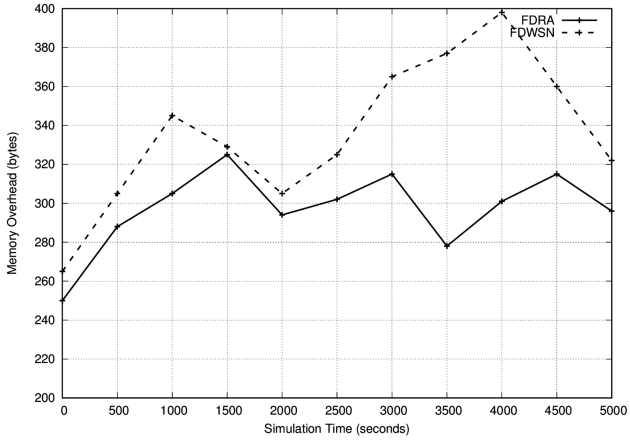

Fig. 10. MO vs. simulation time (TP = 2mW, number of nodes = 100).

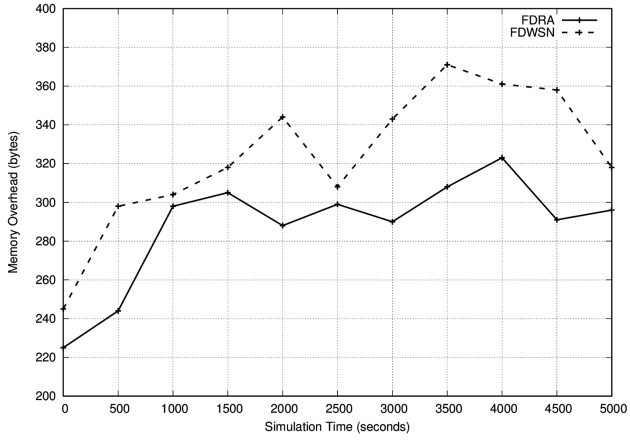

Fig. 11. MO vs. simulation time (TP = 4mW, number of nodes = 100).

4.4 Analysis of Fault Recovery Delay (FRD)

The fault recovery delay is a very important metric in the conception of a fault tolerance protocols. It is defined as the average time taken to recover from the effect of faulty node. From the Fig. 12, it is clear that our FDRA outperforms FDWSN. This is due mainly to the fact that FDWSN requires more time to create and compare transient fault matrices for each faulty node. However, the FDRA can recover a multiple faulty nodes if the position of recovery node selected is close from the faulty nodes (i.e., one sleeping node can recover multiple faulty nodes). When we increase the number of nodes in network to 200 (Fig. 13), we observe a small increase for FDRA compared with FDWSN. The later requires more time to recover from faulty nodes because it needs to compare transient fault matrices.

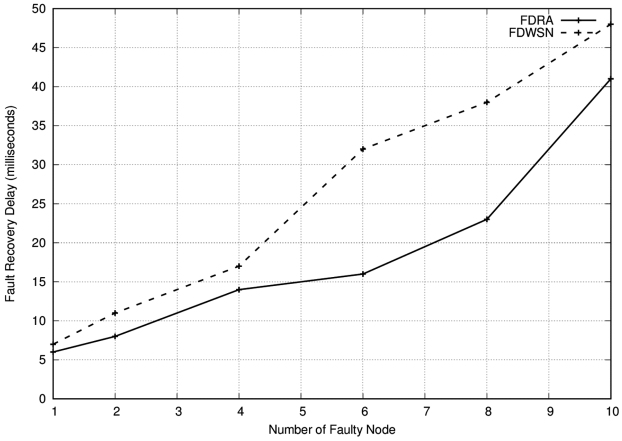

Fig. 12. FRD vs. number of faulty node (number of nodes = 100).

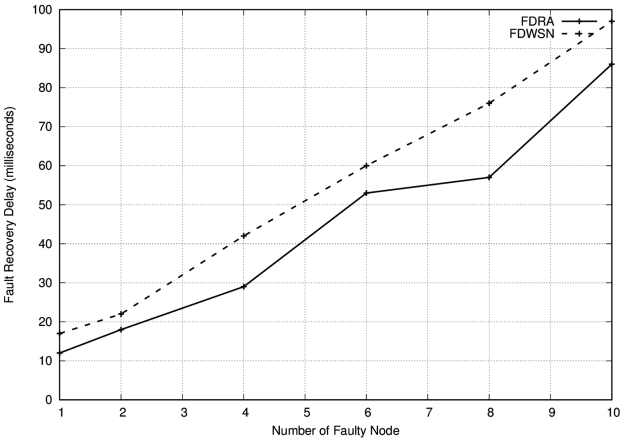

Fig. 13. FRD vs. number of faulty node (number of nodes = 200).

5 Conclusion

This paper focused on recovery faulty nodes within WSN. The main goal was to use sleeping nodes to recover faulty nodes after detection phase. In this paper we have improved our previous work to ensure a recovery process. The proposal algorithm is divided into two phases, a preliminary phase and a recovery phase. These phases are ordered by the BS and they are launched just after the detection's process of faulty nodes. The extensive simulations have demonstrated that the proposed solution permit an efficient recovery. The obtained results have confirmed also that the proposed algorithm outperforms compared to the results of the comparative algorithm FDWSN.

As a future work, we plan to formally prove the correction of the proposed algorithm. More specifically, prove that (1) it recovers all faulty nodes, and (2) it eliminates network partition.

References

1. Nourizadeh, S., Deroussent, C., Song, Y.Q., Thomesse, J.P.: Medical and home automation sensor networks for senior citizens telehomecare. In: IEEE International Conference on Communications Workshops, pp. 1–5. Dresden, Germany (2009)
2. He, T., et al.: Vigilnet an integrated sensor network system for energy-efficient surveillance. ACM Trans. Sens. Netw. **2**(1), 1–38 (2006)
3. Vicaire, P., et al.: Achieving long term surveillance in vigilnet. In: 25th International Conference on Computer Communications, pp. 1–12. Barcelona, Spain (2006)
4. Song, H., Zhu, S., Cao, G.: SVATS: a sensor-network-based vehicle anti-theft system. In: 27th Conference on Computer Communications, pp. 2128–2136. Phoenix, USA (2008)
5. Yang, L., Ji, M., Gao, Z., Zhang, W., Guo, T.: Design of home automation system based on zigbee wireless sensor network. In: 1st International Conference on Information Science and Engineering, pp. 2610–2613, Nanjing, China (2009)
6. Szewczyk, R., Mainwaring, A., Polastre, J., Anderson, J., Culler, D.: An analysis of a large scale habitat monitoring application. In: 2nd International Conference on Embedded Networked Sensor Systems, pp. 214–226. Baltimore, USA (2004)
7. Toll, G., et al.: A macroscope in the redwoods. In: 3th International Conference on Embedded Networked Sensor Systems, pp. 51–63. San Diego, USA (2005)
8. Sikka, P., Corke, P., Valencia, P., Crossman, C., Swain, D., Bishop, H.G.: Wireless adhoc sensor and actuator networks on the farm. In: 5th International Conference on Information Processing in Sensor Networks, pp. 492–499. Nashville, USA (2006)
9. Werner, A.G., Swieskowski, P., Welsh, M.: Real-time volcanic earthquake localization. In: 4th International Conference on Embedded Networked Sensor Systems, pp. 357–358. Boulder, USA (2006)
10. Demirbas, M.: Scalable design of fault-tolerance for wireless sensor networks. Ph.D thesis, The Ohio State University, USA (2004)
11. Ramanathan, N., Kohler, E., Girod, L., Estrin, D.: Sympathy: a debugging system for sensor networks. In: 29th IEEE International Conference on Local Computer Networks, pp. 554–555. Florida, USA (2004)
12. Ringwald, M., Römer, K., Vitaletti, A.: SNIF: Sensor network inspection framework. Technical report 535. ETH Zurich, Zurich, Switzerland (2006)
13. Titouna, C., Aliouat, M., Gueroui, A.M.: FDS: fault detection scheme for wireless sensor networks. Wirel. Pers. Commun. **86**(2), 549–562 (2016)
14. Mohamed, Y., Izzet, F.S., Kemal, A., Sookyoung, L., Senel, F.: Topology management techniques for tolerating node failures in wireless sensor networks: a survey. Comput. Netw. **58**, 254–283 (2014)
15. Titouna, C., Aliouat, M., Gueroui, A.M.: Outlier detection approach using bayes classifiers in wireless sensor networks. Wirel. Pers. Commun. **85**(3), 1009–1023 (2015)
16. Wang, W., Srinivasan, V., Chua, K.C.: Using mobile relays to prolong the lifetime of wireless sensor networks. In: 11th Annual International Conference on Mobile Computing and Networking, pp. 270–283. Cologne, Germany (2005)
17. Basu, P., Redi, J.: Movement control algorithms for realization of fault-tolerant ad hoc robot networks. IEEE Netw. **18**(4), 36–44 (2004)

18. Abbasi, A.A., Akkaya, K., Younis, M.: A distributed connectivity restoration algorithm in wireless sensor and actor networks. In: 32nd IEEE International Conference on Local Computer Networks, pp. 496–503. Dublin, Ireland (2007)
19. Akkaya, K., Thimmapuram, A., Senel, F., Uludag, S.: Distributed recovery of actor failures in wireless sensor and actor networks. In: IEEE Wireless Communications and Networking Conference, pp. 2480–2485. Las Vegas, USA (2008)
20. Tamboli, N., Younis, M.: Coverage-aware connectivity restoration in mobile sensor networks. J. Netw. Comput. Appl. **33**(4), 363–374 (2010)
21. Akkaya, K., Senel, F., Thimmapuram, A., Uludag, S.: Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility. IEEE Trans. Comput. **59**(2), 258–271 (2010)
22. Younis, M., Lee, S., Gupta, S., Fisher, K.: A localized self-healing algorithm for networks of moveable sensor nodes. In: IEEE Global Telecommunications Conference, pp. 1–5. New Orleans, USA (2008)
23. Sookyoung, L., Mohamed, Y., Meejeong, L.: Connectivity restoration in a partitioned wireless sensor network with assured fault tolerance. Ad Hoc Netw. **24**, 1–19 (2015). Part A
24. Lee, M.H., Choi, Y.H.: Fault detection of wireless sensor networks. Comput. Commun. **31**(14), 3469–3475 (2008)
25. Sergiou, C., Vassiliou, V.: Source-based routing trees for efficient congestion control in wireless sensor networks. In: 8th International Conference on Distributed Computing in Sensor Systems, pp. 378–383. Hangzhou, China (2012)
26. Sergiou, C., Vassiliou, V.: Tree-forming schemes for overload control in wireless sensor networks. In: 11th Annual Mediterranean Ad Hoc Networking Workshop, pp. 61–66. Ayia Napa, Cyprus (2012)
27. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: accurate and scalable simulation of entire TinyOS applications. In: 1st International Conference on Embedded Networked Sensor Systems, pp. 126–137. Los Angeles, USA (2003)