A Distributed Fuzzy-based Failure Management for Wireless Sensor Networks

Shahram Nourizadeh, Ye-Qiong Song, Jean-Pierre Thomesse

A Distributed Fuzzy-based Failure Management for Wireless Sensor Networks

Shahram Nourizadeh [1,2] Y.Q. Song[1] J.P. Thomesse[1]
[1]LORIA research laboratory - INPL, Nancy - France [2]MEDETIC Association, Colmar - France
{Shahram.Nourizadeh, Song, Thomesse}@loria.fr Shahram.Nourizadeh@medetic.com

***Abstract*— In addition to limited energy resources in wireless sensor networks, failure of the nodes is a constraint to provide a reliable communication. As the failure of a node may have many reasons like mobility of the node and node or link failure, it is clear that in a real environment, we cannot control or reduce the number of failure in a network but it's possible to manage it. In this paper, we propose a technique to improve routing protocols by considering both energy and failure constraints and managing them. By using this technique, the routing protocols dynamically adapt to nodes' failure. The simulations results show the efficiency of this technique in two sample routing protocols.**

Keywords- Failure management, Wireless sensor networks, distributed algorithm, Fuzzy logic

I. INTRODUCTION

Wireless Sensor networks have recently emerged as a premier research topic. Generally, the communication in sensor networks, with battery powered nodes, suffers from failure, low transmission power even more so than in regular wireless networks. Despite of many research projects, failure management in sensor networks is an open research challenge. Nodes' mobility, communication problem and link failure may be detected as a failure, on the other hand, in some networks, like healthcare networks, failure of a sensor can be detected as a health anomaly. These 2 cases can cause false alarms.

Let's continue with our last example, healthcare networks. A healthcare network is resource-constrained medical sensor network which is deployed in an environment like patient's home or a hospital, in order to collect data about some vital signs. The main objective of a healthcare network is the gathering the health information of the patient and delivering them to a server in which the information will be saved in a medical record and also will be processed to detect any abnormally in health state of the patient. Node's or link's failure is one of the common problems in healthcare networks that causes faulty alarms. If the network doesn't be able to grantee a robust data delivery, the healthcare system will have a lot of fault alarm.

For these reasons, it is essential to provide failure management techniques for distributed ad-hoc networks, and particularly healthcare sensor networks. Many recent studies in this area take drastically different approaches to addressing the fault tolerance issue in routing, transport and/or application layers. In addition, in the ad hoc networks, due to the mobility of the nodes and we have faulty alarm problem.

Implementing failure detectors over local networks is, by now, a rather well-known issue, but it is still far from being a solved problem with wireless sensor networks. But traditional solutions fail to solve important problems such as the potentially large number of monitored processes, the higher probability of message loss, the ever-changing topology of the system, and the high unpredictability of message delays [1].

A number of high profile applications for wireless sensor networks have been envisioned [2][3][4]. Fault tolerance in measurements by a group of sensors, was first studied by Marzullo [5]. Marzullo proposed a flexible control process program that tolerates individual sensor failures. Issues addressed include modifying specifications in order to accommodate uncertainty in sensor values and averaging sensor values in a fault-tolerant way.

[6] developed an algorithm that guarantees reliable and fairly accurate output from a number of different types of sensors when at most k out of n sensors are faulty. The results of the scheme are applicable only to certain individual sensor faults and traditional networks. They are not extendable to the reliability needs in complex network levels and most importantly; they do not address the reliability issues that are induced by the ad-hoc nature of the wireless sensor networks.

Multi-sensor data fusion is a problem that recently has attracted a great deal of attention in a number of scientific and engineering communities [7][8][9]. Majority of these works are restricted to sensor fusion of sensors of the same modality.

The remainder of this paper is structured as follows: section II discusses challenges. Our proposal is described in section III, while our simulations are resented in section IV and section V provides concluding remarks.

II. CHALLENGES

As we said, the traditional failure management systems fail to address the needs of distributed systems because they are simplify made for wired local area networks and limited-scale systems.

In particular, traditional implementations are not designed to cope with a large number of processes, a high message loss rate, a dynamic network topology, and the unpredictability of wireless sensor networks.

Message loss, Message saving, Mobility and Topology change are some basic problems that failure detection protocols must address in a distributed system. Because of mobility, topology changes and also limited energy resources, in WSNs, a node's failure that sometimes causes network partitions occurs significantly more frequently than in wired LANs.

Network dynamics resulted by node mobility and node state transitions due to the use of power management or energy efficient schemes may be detected as node failures or wireless link failures. Such a highly dynamic network greatly increases the complexity of failure management.

Also, with bandwidth limitation in a sensor network the failure detector must generate a minimum number of control messages.

However, the traditional failure detectors and management systems assume that all of the nodes of the network are synonymous, that means there is no difference between a node that was crashed *n* times in *t* hours, with a node that was crashed *m* times (m>n) in the same period of time. That will causes an unclear decision making and electing between different nodes.

In addition, in the traditional failure detectors, when a node failed, it will be assumed as a dead node and we don't have a return of the node. That will be a restriction, for example, when a node is in maintenance.

For these reasons, we propose an adaptive and dynamic algorithm for failure management in a mobile network. As the failure of a node may have many reasons like mobility of the node and node or link failure, it is clear that in a real environment, we cannot reduce the number of failure but it's possible to manage it, and this is the main idea of our proposal. In our algorithm, the failure history of the nodes is a parameter to differentiate between them.

But, how we can do this? Can we compare the nodes just with their failure history? Which are the best parameters to do this evaluation? Next sections will answer all these questions.

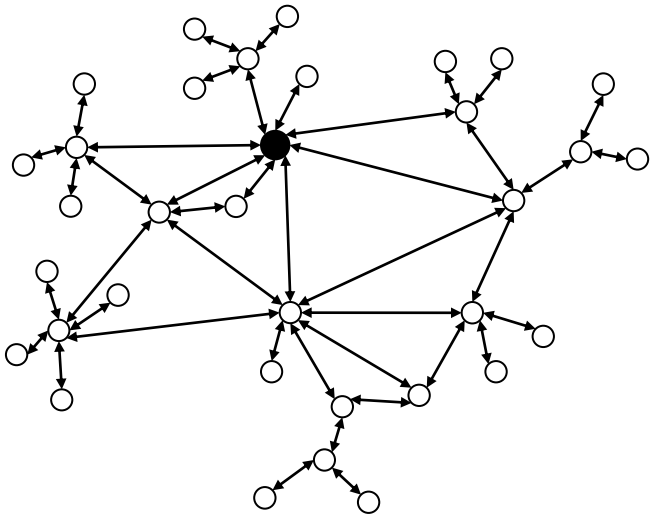


Figure 1. A sample random deployed network

III. PROPOSITION

We assume that all nodes are integrated with a failure detection system and are able to detect the failure. First, in section A, we give the definitions used in the algorithm and in section B we describe the algorithm.

A. *Some definitions*

- *confidence*: This parameter shows the level of confidence that we give to a node, and has three possible values: *High*, *Medium*, *Low*. This parameter shows the priority of nodes for taking part of routing.
- *failure*: This parameter shows history of the failure of a sensor detected by its neighbor. This parameter will be computed by using the number of sensor's failures during its lifetime. It is also a fuzzy variable that has 3 levels: *High*, *Medium* and *Low*.
- *energy*: this is the battery charge of the node. Like the other parameters, *energy* is a fuzzy variable with 3 levels: *High*, *Medium* and *Low*.

B. *How does it work?*

Figure 1 shows as an example a mesh network. The black node is the sink and the others are the wireless mobile nodes. As shown in this figure, the nodes are connected together and we have a P2P connection between them.

By using a traditional failure management algorithm, in this network, all the nodes are equal for us. That means we have not an idea about their failure rate; a node that crashed 10 times and a node that crashed 5 times in the same period, are equal. But by using our algorithm, we will color the network with the *confidence* level of the nodes.

In our algorithm, each node of the network has a neighbor table which has one row for each neighbor. In a dynamic network, the number of rows changes because of the mobility of the nodes. In this table we keep four parameters for each node: *energy*, failure, *Received Signal Strength Indicator (RSSI)* and *confidence*.

To complete the neighbor table, each node runs a function to compute the *confidence* parameter of its neighbors. This function uses fuzzy logic [11][12]. We use fuzzy logic because it can manipulate the rules in a natural way, it can be used in context by blending different parameters – rules combined together to produce the suitable result. Fuzzy logic is capable of making real time decisions, even with incomplete information.

Note: Instead of fuzzy logic, any another technique or logic can be used.

The confidence parameter is a dynamic parameter which means it can change from *Low* to *High* and also *High* to *Low*, because the failure parameter is dynamic. Each time the failure of a node change, we will update confidence of the node also. In each node A, failure of its neighbor B will be computed by: failure = fuzzy (n / L); where fuzzy is a function to convert decimal value to fuzzy value, n is number of B's failures and L is the life time of A.

To re-compute the *failure* and so updating of confidence, we have 2 methods:

1. Periodical update: In each network, due to the mobility or failure frequency of the nodes, base station will define an update period, in which the *failure* of the neighbors will be recomputed.
2. Event-based update: Each time that a node detects a failure in its neighbor, it re-computes its *failure* parameter and updates confidence of the node.

Figure 2 shows an example of *failure* re-computing and *confidence* updating methods. As we can see in this figure, computed *confidence* for B, in node A, change by frequency of B's failure. If number of B's failure remains constant, its *confidence* in A will start to increase.

Figure 3 shows an example of neighbor table in a sample network. We have the nodes A and B, and their common neighbors, x and y. We can see in this figure that x and y, the common neighbors of A and B, have different levels of *confidence* in neighbor table of A and B. But how can we explain this?

As we said, we use 3 parameters to compute *confidence* of each node: its *energy* level, its *RSSI* and its *failure*. The *energy* level of x will be same in neighbor table of A and B, as the energy level of y. But, *RSSI* and *failure* history will be different. A, and B may detect different number of failures in x, and the *RSSI* may be different because of difference of environmental conditions between the nodes. Therefore in our proposal, even a common neighbor of two nodes, may have different level of *confidence* in each of them.

Figure 4 shows the fuzzy decision making process. As the *energy*, *RSSI* and *failure* have each of them, 3 possible values, in Inference state, we have in total 27 rules to evaluate for each node.

IV. EVALUATION

This section presents the evaluated performance of our proposition. We integrated OPNET simulators to implement the physical and MAC layer, with an application developed in C to implement Fuzzy rules. Our simulation focuses on the delivery ratio of the network packets as a performance metric and in each step of the simulation; number of faulty nodes is variable.

A. Some assumptions

To evaluate our protocol, and to reduce simulation complexity, to have clear results, we have these assumptions:

- The network consists of several wireless nodes and a base station and nodes have a low mobility.
- To focus on the assessment of the performance of the proposed algorithm, we do not generate any user data traffic during a simulation.
- All the nodes are able to detect correctly the failure of the other nodes. They can use heartbeat or any other failure detection.
- When a node fails or crashes, it is not dead; it will be return to the network after a variable time, $t \neq \infty$.

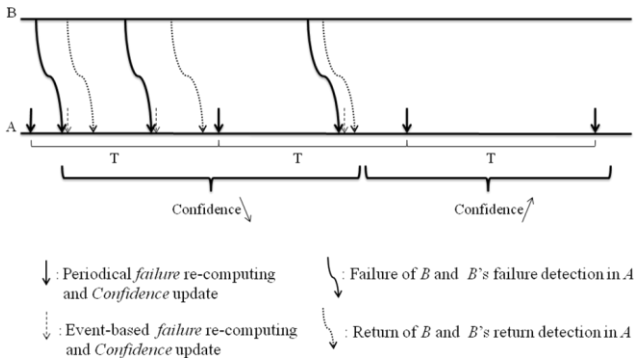
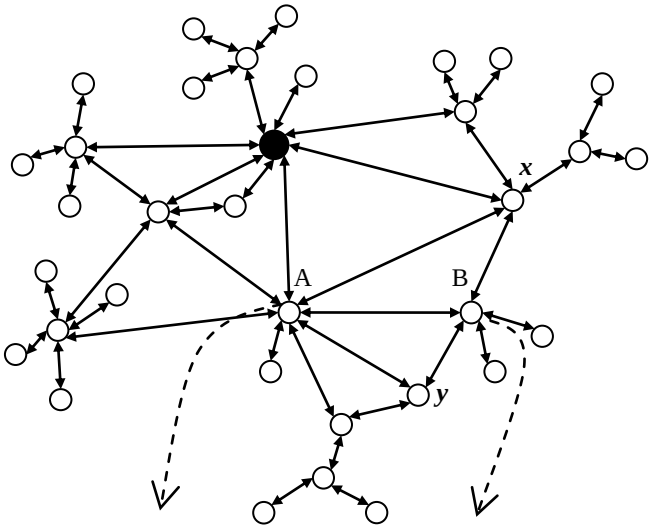


Figure 2. Updating *Confidence* parameter



Neighbor		*confidence*
…		…
x		High
…		…
y		Medium

Neighbor Table of A

Neighbor		*confidence*
…		…
x		Medium
…		…
y		Low

Neighbor Table of B

Figure 3. A senario of Neighbor Tables

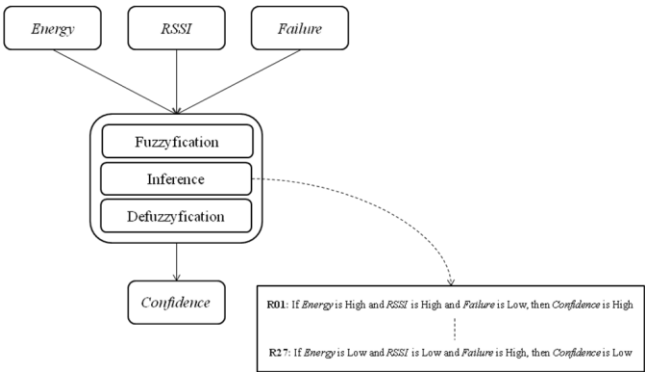


Figure 4. Fuzzy decision making process

TABLE I. SIMULATION PARAMETERS

Node Number	50
Surface	50m x 50m
Transmission range	15m
Data transmission rate	15 packet/sec
Failure model	Random
Packet size	128 bytes
Initial Energy	5J
Energy consumption (Calculation, receive and send)	10 nJ/bit

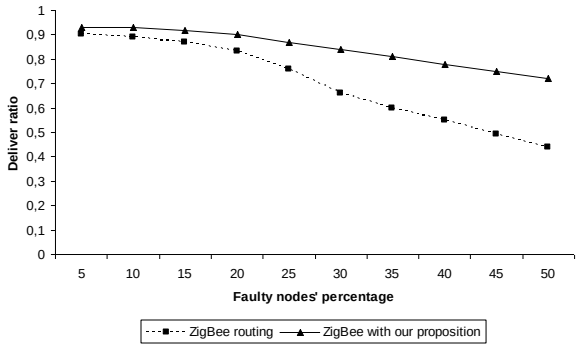


Figure 5. Delivery ratio in ZigBee

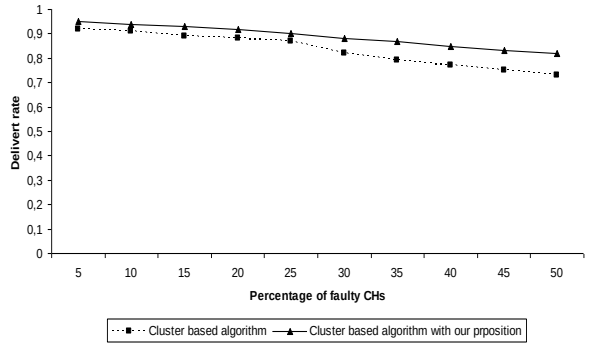


Figure 6. Delivery ratio in [14]

B. Simulation parameters

Table II, shows our simulation parameters. We evaluated our algorithm in a random deployed sensor network, and with 2 different algorithms:

- ZigBee Mesh routing protocol [13]
- a cluster based routing algorithm [14]

C. Simulation results

In the simulation of our algorithm, we focus on the network delivery ratio. Network delivery ratio is defined as the total received packets divided by the total sent packets in the sensor network. We compared this metrics in normal ZigBee mesh routing protocol with the one that is integrated with our failure management algorithm.

Figure 3 shows the result of this simulation. The simulation shows that by applying our algorithm, we can increase the data delivery ratio by 20%. As we can find in this figure, when we have 25% or more of faulty router nodes in a ZigBee network, by applying our algorithm, we have a greater data delivery ratio, compared to a normal ZigBee routing protocol.

We also applied our failure management algorithm, to a clustering protocol [14]. We can find the simulation result in figure 4 that compares 2 versions of [14]: with and without our algorithm. Simulation results show that by using our algorithm, the data delivery ratio in [14] is increased by 5%.

V. CONCLUSION

A fuzzy logic based failure management algorithm was proposed in this paper. Stable route recovery and high data delivery ratio are the main characteristics that the proposed protocol adds to the routing algorithms.

This algorithm is especially effective in networks that use sensor nodes for data collection and in which the data delivery ratio is important.

In all of the data acquisition systems like health monitoring systems, either the data is collected from the network periodically or on the occurrence of an event, the data are highly vital and we must have a minimum number of faulty alerts. This necessitates the use of a protocol for data collection that readily adapts to the failures of the nodes and changes in the data delivery rate.

In such networks health events and information is sensed by several nodes and therefore, this protocol can help the network to deliver these sensed events and avoid data loss in the network. The simulation results show that the proposed protocol is well suited for such applications.

REFERENCES

[1] Naohiro Hayashibara, Xavier Défago, Rami Yared, Takuya Katayama: The F Accrual Failure Detector. SRDS 2004: 66-78

[2] M. Weiser, “The Computer of the Twenty-First Century,” Scientific American, pp. 94-100, September 1991.

[3] D. Tennenhouse, “Proactive Computing,” Communications of the ACM, vol. 43, no. 5, pp. 43-50, 2000.

[4] D. Estrin, R. Govindan, and J. Heidemann. “Embedding the Internet: Introduction,” Communications of the ACM, vol. 43, no. 5, pp. 38-42, 2000.

[5] K. Marzullo, “Tolerating failures of continuousvalued sensors,” ACM Transactions on Computer Systems, vol. 8, no.4, pp. 284-304, November 1990.

[6] D.N. Jayasimha, “Fault tolerance in multi-sensor networks,” IEEE Transactions on Reliability, vol. 45, no.2, pp. 308-15, June 1996.

[7] R. R. Brooks and S. S. Iyengar, “Multi-Sensor Fusion: Fundamentals and Applications With Software,” Prentice- Hall, 1997.

[8] P. K. Varshney, “Distributed Detection and Data Fusion,” N.Y.: Springer-Verlag, 1997.

[9] G.D. Hager, “Task-Directed Sensor Fusion and Planning - A Computational Approach,” Kluwer Academic Publishers, 1990.

[10] J. J. Clark and A. L. Yuille, Data Fusion for Sensory Information Processing Systems, Kluwer, 1990

[11] L.A. Zadeh, Fuzzy Sets, Information and Control, 1965

[12] L.A. Zadeh, Outline of A New Approach to the Analysis of Complex Systems and Decision Processes, 1973

[13] ZigBee routing protocol: Z. Alliance “Zigbee alliance.” [Online]. Available: http://www.zigbee.org/

[14] S. Nourizadeh, Y.Q. Song, J.P. Thomesse, “A Location-Unaware Distributed Clustering Algorithm For Mobile Wireless Sensor Networks Using Fuzzy Logic”, FET2007