Efficient Adaptive Failure Detection for Query/Response based Wireless Sensor Network

Fatima Zohra Benhamida, Yacine Challal, Mouloud Koudil

Efficient Adaptive Failure Detection for Query/Response based Wireless Sensor Networks

Fatima Zohra Benhamida
Laboratoire de Méthode de Conception de Systèmes
Ecole Nationale Supérieure en Informatique
Algiers, Algeria
f_benhamida@esi.dz

Yacine Challal
Laboratoire Heudiasyc, UMR CNRS 6599
Université de Technologie de Compiègne
Compiègne, France
ychallal@hds.utc.fr

Mouloud Koudil
Laboratoire de Méthode de Conception de Systèmes
Ecole Nationale Supérieure en Informatique
Algiers, Algeria
m_koudil@esi.dz

***Abstract*—we consider the problem of failure management in wireless sensor networks (WSN). In such networks, nodes can be subject to frequent failures due to energy depletion and the hostile deployment environment. Our work focuses on local crash process detection in WSN considering intermittent failures due to lossy radio links. As a part of this problem, we introduce a new type of Adaptive Neighborhood Failure Detection mechanism (ANFD) that relies on adaptive timers. Then we analyze properties of the introduced failure detection algorithm and determine how and when timers' update is necessary. We carried out intensive simulations using TinyOS-2 and evaluated the performance of our solution. Simulation results show that our solution improves failure detection rate and delay, and reduces fault positive detections and packet loss ratio. Moreover, we analyzed energy consumption using PowerTossimZ and results demonstrate that our solution does not induce extra energy consumption compared to other solutions in the literature.**

Keywords- failure management, wireless sensor networks, lossy links, process crash.

I. INTRODUCTION

Wireless sensor networks (WSN) are prone to a variety of failures due to energy limitations, radio transmissions and close coupling with environment where network components may be not physically protected. Therefore, fault management is one of the critical issues in WSN. It requires new communication protocol considering the ability to cope with node failures. However, the design of these new approaches must respond to the following WSN constraints and properties: (1) a node does not necessarily know all the nodes of the network due to storage capacity limitations (2) the network is not fully connected because of radio range limitation, which means that a message sent by a node might be routed through a set of intermediate nodes until reaching the destination; (3) communication is data-centric: in most WSN applications, data is requested by a collector by sending a query based on certain attributes. Then, source nodes respond with packets containing required information. This Query/Response mechanism (denoted here by Q/R) is periodically invoked during the lifespan of the network; (4) links are prone to failures and may momentarily drop messages during transmission; (5) nodes are equipped with limited energy batteries: communication protocols have to consider this limitation to maintain the network alive as long as possible.

Failure detection has been deeply investigated in the literature of distributed systems [5]. However, the proposed failure detectors do not meet WSN constraints and requirements and rely on assumptions hardly verified in WSN. Indeed, most of proposed failure detection classes are based on an all-to-all communication approach where each process periodically sends a heartbeat message to all processes [6][11][13]. As they usually consider a fully connected set of known nodes, these implementations are not adequate for dynamic and partially connected environments, such as WSN. Furthermore, failure detectors are usually based on packet acknowledgment [7] where the receiver sends back an acknowledgment to the source after every packet reception. Otherwise, the sender suspects the receiver of having crashed. Both heartbeat and acknowledgment mechanisms are not suitable for WSN where processes are limited in energy and communication bandwidth. In [14], a timer-free asynchronous failure detector has been proposed. It is based on an exchange of messages and assumes the knowledge of two values: f (the maximum number of processes that can crash) and n (the number of nodes in the system). Moreover, the computation model consists of a set of initially fully connected known nodes. Yet, this timer-free approach is not applicable to a partially-connected network such as WSN. Some works have been proposed which deal with the scalable nature of dynamic systems. Nonetheless, few of them tolerate links failures [1][10]. In most of works, they only considered systems where process crashes are permanent and links are reliable (i.e., they do not lose messages). This may increase the risk of mistakes, when the failure detector suspects a process of having crashed while the packets were lost because of intermittent transmission failure due to unreliable links.

In this paper, we define a new failure detection class tailored to the constraints and requirements of WSN, called ANFD (Adaptive Neighborhood Failure Detection). Indeed, our solution relies on adaptive timers to detect nodes crash while considering intermittent failures of radio links. The detection of process failures is based only on a local perception that the node has on the network and not on global exchanged information. The exchanged lists of failure suspicions and mistakes are piggybacked into

Query/Response messages of the WSN application. Then we analyze the properties of the defined class in terms of completeness and accuracy of failure detection in typical WSN. Furthermore, we carry out simulations using TinyOS2 to evaluate the performance of the proposed solution. We particularly evaluate the induced energy consumption overhead using PowerTOSSIM-Z and demonstrate that in addition to the benefits of the proposed solution in terms of failure detection and recovery, and reducing packet loss ratio and fault positive detections, it does not introduce extra energy overhead compared to other solutions.

The paper is organized as follows: after the introduction and state of the art presented in this section, we present an overview and motivation of the proposed solution in section II. In section III, we present the different models assumed for the operation of the proposed solution. Then we present a detailed description of the solution and its properties in section IV. We present simulation results using TinyOS-2 and PowerTOSSIM-Z and performance evaluation analysis and comparison in section V. We end up this paper with conclusions in section VI.

II. OVERVIEW

Let us consider the following simple configuration to illustrate the main issues and motivation behind our proposal. Consider a system of two processes (i.e. a network of two sensor nodes): a data collector P_i and a data source P_j. Process P_i wants to inquire regularly some information from the network; it sends periodically a query to process P_j. P_j responds with a data packet containing required sensed data. This is known as a Q/R mechanism. Let us assume first that links are reliable which means that messages sent from correct processes are successfully transmitted to their destination. For process crash detection, we use a *Failure Detection Timeout* (FDT_{ij}): if FDT_{ij} runs out before receiving the expected response, P_i suspects that P_j has crashed. Each FDT_{ij} value is initiated according to application parameters; such as Q/R round trip delay, 1-hop transmission latency, etc.

However, the situation changes if, in addition to process crash, link failures may also occur. In fact, with the above *FDT*, P_i may make some mistakes, when the process is still correctly working but some messages are lost because of lossy links (intermittent failures). For this reason, we need to update FDT_{ij} value following the lossy pattern of radio links.

In this paper, we explore the use of adaptive *FDT* to circumvent this obstacle: we use a stochastic approach to determine *FDT* taking into consideration loss ratio, links state, transmission delay, etc. It is important to notice that *FDT* is defined between two consecutive neighbor sensors instead of any couple of nodes. This allows more accurate calculation of *FDT* that relies then on local predictable transmission delay without requiring knowledge about the entire system composition and network topology. Nevertheless, local suspicions and mistakes will propagate throughout the network and hence taken into consideration in route maintenance by the overall nodes of the WSN. We present thereby, a new failure management class for WSN, where each process can query a failure detection module that provides information about which process of its neighbors has crashed. This information is typically given in a form of a list of suspects ($Suspected_i$, is the list of process P_i containing its suspected neighbors). However, mistakes can be made: a process may be suspected though it has not crashed. Hence, in addition to the $Suspected_i$ list, each module has a list of $Mistake_i$ where it notifies previous mistaken suspicions and retrieves correct processes from $Suspected_i$ list. All these notifications are piggybacked on Q/R messages using neighborhood interaction to avoid extra overhead that would have been induced if dedicated signaling packets were used.

In ***Erreur ! Source du renvoi introuvable.***we illustrate the interactions between the different modules with our failure detection module: ANFD uses the characteristics of the data gathering protocol (e.g. round delay) and the pattern of packet losses to determine a timer *FDT* for each neighbor. The neighboring list is given by the network module. Furthermore, the data gathering protocol sends (resp. receives) Q/R messages to (resp. from) ANFD module in order to update (resp. deliver) the lists of suspicions and mistakes. Thus, every Q/R message piggybacks information about nodes' crashes and mistaken suspicions. Moreover, both lists updates will lead to a new *FDT* refresh in order to give more accurate notifications in next rounds.

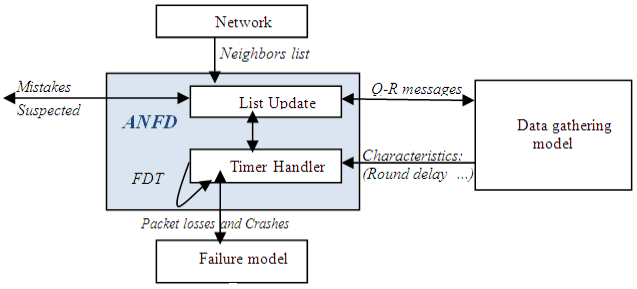

Figure 1. Different modules interaction.

III. SYSTEM MODEL

A. Network model

We consider a sensor network consisting of a finite set V of $n>1$ processes, namely, $V= \{P_1,...,P_n\}$. There is one process per node and they communicate by sending and receiving messages via a wireless network. Processes have no knowledge about V or n; but, they know a subset of V, composed of nodes with whom they previously communicated. The system can be represented by a communication graph $G(V,E)$ in which V represents the set of nodes and E represents the set of radio links. If $(P_i,P_j) \in E$ then P_i and P_j are considered neighbors. Moreover, we suppose that the network is *f-covering*; i.e. the graph $G(V,E)$ is $(f+1)$ connected. This property ensures that there is always a route between any two nodes of the network in spite of f faults.

B. Failure model

We assume that sensor nodes of the WSN may crash, as it is commonly assumed in the literature [?, ?, …]. Moreover, links between two sensor neighbors are considered lossy: they may drop messages during transmission. This is commonly due to transmission link failure; which is intermittent by nature. Subsequently, if a process crashes, it will never belong to the network anymore; and has to be notified to all its neighbors. However, if it fails to send some packets, this loss should be considered as intermittent failure. Hence, even if ANFD may suspect a process of having crashed while actually the link is down it will be able to detect mistaken suspicions once the link is set back.

C. Data gathering model

We consider data-centric Q/R based routing protocols. In this kind of data gathering models a collector node (also known as a sink) periodically broadcasts a data *Query* to all nodes using a multi-hop routing scheme. Each intermediate node receives the query from an upstream neighbor, and then diffuses it to all its neighboring nodes. When a source observes a matching event, it sends back a *Response* to nodes which it should reply to. In the same way, intermediate nodes forward the *Response* until it reaches the sink. Without loss of generality, Directed Diffusion [12] is a scheme that matches this data gathering model, and we will show further how ANFD can improve DD to cope with failures and route recovery.

IV. ADAPTIVE NEIGHBORHOOD FAILURE DETECTION

We present hereafter ANFD algorithm for process P_i. We suggest including ANFD as a module between the layer that manages the Q/R messages and the beneath layer. ANFD can interact with both layers using four main primitives, and a set of variables and procedures introduced in TABLE I.

TABLE I. NOTATIONS AND TERMINOLOGY FOR ANFD.

Variable/ Procedure	Presentation
$Neighbors_i$	List of P_i's 1-hop neighbors updated outside our algorithm[1].
$Counter^j$	Round counter at node P_i. $Counter_i^j$ is the round counter attributed to node P_j at node P_i. The round counter is used as a stamp for generated information.
$Suspected_i$	Set of processes suspected by P_i of having crashed. Each element consists of a tuple $<P_x, counter_i^x>$; P_x is the suspected process at time $counter_i^x$.
$Mistake_i$	Set of nodes which were previously suspected of having crashed but such suspicions are currently considered to be false. Each element consists of a tuple $<P_x, counter_i^x>$; where P_x is a previously suspected process.
QRLayer.Receive (msg)	Delivers messages to the layer managing Q/R messages
DLayer.Send (msg)	sends messages to downward layer to be transmitted to its destination
Piggyback(*msg, suspected, mistake*, P_i)	adds to *msg*, both of *suspected* and *mistake* set, and the source P_i
Retrieve(*msg, suspected, mistake, source*)	reads and returns both *suspected* and *mistake* sets from *msg* and their *source*
Add(*set*, <*id, ctr*>)	Inserts the new process *id* in the list *set* or updates the attributed *ctr* if it already exists with a smaller counter.
SetFDT(P_j)	evaluates and sets or updates the failure detection timeout for process P_j.
Timer.schedule (P_j, FDT_{ij})	triggers a timer FDT_{ij} for process P_j.
Timer.stop(P_j)	stops timer FDT_{ij} for process P_j.
Actual (<P_l, $counter_l^j$>)	Statement that returns true if: P_l does not exist in the caller's (P_i) suspect and mistake lists, or P_l exists in P_i's lists but with smaller counter.

[1] We suppose that external module updates $Neighbors_i$ list each time P_i receives a new message from unknown node P_j.

A. ANFD algorithm

We recall that we consider a Q/R data gathering model. The algorithm proceeds by rounds. At each Q/R round, a node (generally the collector) launches a query broadcast. Each node forwards the received Q/R message to the nodes in its range until it possibly crashes. The basic principle of our approach is to piggyback failure suspicion on the Q/R messages, and dispatch this information hop by hop. In addition to application data, we add up to each message two sets of nodes: $Suspected_i$ (the set of nodes that are suspected of being faulty), and $Mistake_i$ (the set of the mistakes; the nodes that were previously erroneously suspected of being faulty). Each node keeps a counter, which is incremented at every round. Any new information that is generated by this node about failure suspicions or correction of false suspicions (mistakes) within a round is tagged with the current value of such a counter. This tag mechanism avoids obsolete information to be taken into account by nodes.

ANFD module for process P_i is described in Figure 2. It is composed of two main primitives and one procedure: The primitive *ANFD.Send(msg)* includes both of $Suspected_i$ and $Mistake_i$ sets in the message *msg* originated from the upper layer (line 4). After piggybacking the information, P_i sends the new message to the downward layer (line 5) in order to forward it to the destination node. In addition, if the original message is a query, P_i initializes a timer for every correct known neighbor; say FDT_{ij} for node P_j awaiting for a response to its query (lines 6-9). At the end of *ANFD.Send*, $counter_i$ is incremented by one. If any FDT_{ij} runs out (line 19), *Timer.Fire*(P_j) is triggered; that is, P_i suspects P_j of being faulty. First, it checks if $Mistake_i$ list contains P_j, so it is removed from it. The new suspicion information is then included in $Suspected_i$ with a tag which is equal to the current value of $counter_i$ or with a greater tag than the one associated with P_f in $Mistake_i$ set if it was previously inside (lines 20-23). This suspicion will be included in the next Q/R message when *ANFD.Send* is executed again.

The second primitive *ANFD.Receive(msg)* aims to update *suspected* and *mistake* sets according to those piggybacked on the received message delivered from the downward layer. First, P_i retrieves information, namely, both *suspected* and *mistake* sets with their source node P_j (line 12), then it stops FDT_{ij}, since the message was successfully delivered (line 13). P_i checks then if P_j (the source) was recently suspected in order to delete it from $Suspected_i$ set and insert it into $Mistake_i$ with current $counter_i$ (lines 14-16). After that, P_i calls the *UpdateLists* procedure in order to treat the received

information about suspicions and mistakes in P_j's message (line 17). Finally, *QRLayer.Receive* sends the new message *msg* to the upward layer (line 18). The two loops of procedure *UpdateLists* handle information about suspected (respectively erroneously suspected) processes. Thus, for each node P_l included in $Suspected_j$ (resp. $Mistake_j$) set, P_i includes P_l in its $Suspected_i$ (resp. $Mistake_i$) set only if it received more recent information about P_l status (faulty or mistaken of being faulty). In such a case, P_i removes P_l from its lists to insert the given more recent information in the corresponding set (lines 25-28& 29-32).

```
1   Init :
2     suspected_i ← Ø; mistake_i ← Ø; counter^i ← 0
3   ANFD.Send(msg):
4     Piggyback(msg, suspected_i ,mistake_i, P_i)
5     DLayer.Send(msg)
6     if type(msg) = QUERY then
7      for all P_j ∈ neighbors_i\suspected_i do
8          FDT_ij=setFDT(P_j)
9         Timer.schedule (P_j,FDT_ij)
10    counter_i = counter_i + 1
11  ANFD.Receive(msg):
12   Retrieve(msg, suspected_j, mistake_j, P_j)
13   Timer.stop(P_j)
14   if <P_j,counter^i_j>∈suspected_i or
     counter^i_j≤counter^i then
15           Add(mistake_i, <P_j,counter^i>)
16           suspected_i = suspected_i - <P_j,..>
17    UpdateLists(P_j, suspected_j, mistake_j);
18    QRLayer.receive(msg)
19  Timer.Fire(P_j):
20    if <p_j ,counter^i_j> ∈ mistake_i then
21       counter^i = max(counter^i, counter^i_j + 1)
22       mistake_i = mistake_i - <P_j ,..>
23    Add(suspected_i, <P_j, counter^i>)
24  UpdateLists(P_j, suspected_j, mistake_j) :
25    for all <P_1,counter^j_1 >∈ suspected_j do
26     if(actual(<P_1,counter^j_1>) then
27        mistake_i = mistake_i - <P_1,..>
28        Add(suspected_i,<P_1,counter^j_1>)
29    for all <P_1, counter^j_1> ∈ mistake_j do
30     If (actual(<P_1,counter^j_1>) then
31         suspected_i = suspected_i \ <P_1,..>
32         Add(mistake_i, <P_1, counter^j_1>)
33  end
```

Figure 2. ANFD algorithm.

B. How to determine FDT (SetFDT function)

The Failure Detection Timeout (FDT) should be dynamically updated while taking into consideration transmission failures, communication protocol behavior and network dynamics in general. Notice that there is no general formula since it depends on the choice of the routing protocol and its parameters. We illustrate by an example in section V how to set its value for a specific protocol, namely Directed Diffusion. In general case, we need to consider the following factors:

- Transmission delay between 1-hop neighbors: to estimate the required time for a packet to be transmitted from source to destination in a direct link (i.e. 1-hop neighbors).
- Round delay for the Q/R mechanism: ANFD should wait at least a round trip of the Q/R messages before suspecting any neighbor failure.
- Tolerated Burst Loss (*TBL*) in link failure: to postpone the suspicion in order to tolerate intermittent failures due to packet loss (not process crash). Indeed, it has been demonstrated [7][9] that losses in wireless networks follow a bursty pattern. Therefore, *FDT* must be at least greater than the average value of the *TBL*.
- Network activity statistics: network traffic may influence the effective delays for packet transmission, and packet losses. *FDT* has to take into consideration such actual network traffic and its impact on packet transmission delays and losses. Knowing that *FDT* is updated for 1-hop neighbor, it is then easy to use such parameter.

In the beginning, since there is no former communication to make statistics from, the first *FDT* value is initially set according to hypothetical values of transmission interval and latency. After which, *FDT* is updated using the effective loss rate regarding received Q/R messages. Moreover, after each mistaken suspicion, *FDT* must be increased in order to adapt the failure detection mechanism to the network activity and avoid notifying link failures (i.e. packet loss) as crashes.

C. Properties of the new failure detection class for WSN

Any failure detector FD is characterized by two properties: completeness and accuracy [5]. Completeness requires that FD eventually suspects every process that actually crashed, while accuracy limits the mistakes FD can make. Since we have considered network and failure models tailored to WSN, we define new completeness and accuracy properties that better suite our network, failure and data gathering models:

Definition 1 (1-hop-Completeness):There is a time after which every process P_i that crashes is suspected by all its correct neighbors.

$$\forall P_i \text{ crashes}, \forall P_j \in Neighbors_i, \exists \tau \in T: \forall t > \tau, P_i \in Suspected_j$$

This property satisfies strong completeness limited to 1-hop neighborhood.

Definition 2 (1-hop-Accuracy):There is a time after which some correct processes are never suspected by any correct neighbor.

$$\exists P_i \text{ correct}, \forall P_j \in Neighbors_i, P_j \text{ correct}, \exists \tau \in T: \forall t > \tau, P_j \notin Suspected_j$$

This property defines the eventual weak accuracy limited to 1-hop neighbors.

The merit of these new definitions is to provide adequate measurement tools of completeness and accuracy of failure detection tailored to the limitations and requirements of

WSN. Indeed, if strong completeness and accuracy are mandatory in conventional distributed systems, in WSN 1-hop-completeness and 1-hop-accuracy are enough for route update provided that the network remains connected (f-covering property). Therefore, due to energy, storage and bandwidth limitations, every node monitors only its direct neighbors, even though the information (Query resp. Response) is initiated by a further node (sink resp. source). Later, suspicion notifications sent initially to neighbor nodes will reach on-route nodes to the sink and hence update their routes consequently. Finally, notice that, using this local interaction is energy efficient as will be demonstrated in performance evaluation section through simulations using TinyOS2 and PowerTOSSIM-Z.

Proposition: *ANFD guarantees the following properties:*
- *1-hop-completeness*
- *1-hop-accuracy*

Proof:

V. PERFORMANCE EVALUATION

In this section, we present simulation results with respect to the adaptive *FDT* used in the new proposal ANFD. We will focus mainly on the outcome of using adaptive failure detection timer that allows, as we will see, to enhance failure detection with the presence of intermittent failures due to packet losses. We carried out the simulations using TinyOS2 [?] augmented with PowerTOSSIM-Z [?] for energy consumption profiling.

A. Simulation model

We will consider Directed Diffusion (DD); a Q/R based data gathering protocol, as a basis for our study. DD does not use any failure management mechanism; it relies instead on periodic diffusion of exploratory data to circumvent eventual faulty nodes. We have implemented DD in TOSSIM/TinyOS [14] environment. We consider a network composed of 50 to 100 nodes randomly deployed. We modeled packet loss using a two-state Markov chain [7][9]. We simulate node crashes by running a python script that selects a subset of nodes to turn off (node *i* crashes at time *t*) following an exponential law. In order to illustrate the outcome of using adaptive failure detection, we considered several simulation scenarios using original DD without failure detection, Fat2D [?] (DD with constant *FDT)* and the proposed solution: DD augmented with ANFD (DD equipped with adaptive *FDT*). We were interested in four main metrics:

- the failure detection time which is the required period between failure occurrence and its detection.
- The packet loss rate, defined by the proportion of lost data due to route breakdown after some node crashes. The performance of ANFD is greater when it can decrease the amount of lost data.
- The mistaken suspicions rate or false positive rate defined by the proportion of falsely suspected nodes due to intermittent failures. This allows evaluating the accuracy property.
- Energy consumption which provides hint on the network lifespan and the overhead induced by introducing ANFD.

B. Simulated protocols parameters

In this section we present the considered configuration of the three simulated protocols using TinyOS2 that we analyze and compare at the end of this section.

1) Directed Diffusion (DD) [?]

Recall that in DD a source sensor sends sensed data each I_r (reinforcement interval) time units to the upward node toward the sink. Besides, each I_e (exploratory interval) time units, the source sensor broadcats data in order to discover eventual new routes and hence circumvent failed ones.

2) Fat2D [?]

in this protocol, DD is augmented with a static failure detection timeout (FDT) calculated as follows:

$$\mathbf{FDT} = \begin{cases} \mathbf{2 \times I_r} & \mathbf{if\ TBL} \leq \mathbf{2 \times I_r} \\ \mathbf{TBL} & \mathbf{if\ 2 \times I_r} < \boldsymbol{TBL} \leq \mathbf{I_e} \\ \mathbf{I_e} & \mathbf{if\ TBL} > \mathbf{I_e} \end{cases} \quad \ldots\ldots 1$$

Where I_r, I_e are reinforcement (resp. exploratory) intervals, and TBL (Tolerated Burst Loss) is the average length of the burst loss due to intermittent radio transmission failure,

Formula (1) means that FDT equals to the average length of a burst loss with lower and upper limits: the lower limit is the exploratory interval where DD naturally discovers new routes and circumvent failed ones, and the upper limit is two times the reinforcement interval beneath which we cannot suspect a loss since the source is not supposed has sent a new message.

3) ANFD

For ANFD, the initial FDT_0 is defined by formula (1). Then, for *FDT* update, we check the mistaken suspicions rate[2] after each round (cf. algorithm in fig. ?, line 3). If it exceeds a fixed threshold τ, we increase the TBL to tolerate more burst loss. For this reason, we increase the percentage of accepted burst loss (say by 10%), and then update TBL by using the cumulative distribution function (cf. algorithm in figure ?, lines 4&5). Once TBL is updated, we use formula (1) to get the new FDT.[3]

```
1 FDT←FDT0; threshold←τ; Ie; Ir; TBL; TBL_rate;
2 For each round:
3     if mistake_rate > τ then
4        TBL_rate← TBL_rate+10%
5        TBL= -1/λ*log(1- TBL_rate)
6        if TBL< 2*Ir then
7             FDT = 2*Ir
8        else if TBL< Ie then
```

[2] Mistaken suspicions rate is the amount of suspicions regarding total packets loss number.

[3] Notice the choice of threshold and increasing percentage can be fixed according to protocol and application parameters.

```
9              FDT = TBL
10       else  FDT = I_e
```

Figure 3. *FDT* update function for Directed Diffusion.

C. Simulation results

In this section we analyze the impact of implementing ANFD mechanism on DD, with respect to the following scenarios, and compare it to DD and Fat2D:

1) Impact of failure detection timer on detection period: we compare DD, Fat2D (equipped with a static failure detection timer) and ANFD that uses adaptive *FDT*. We vary the number of crashes randomly inserted during the experiment. Results in **Erreur ! Source du renvoi introuvable.**show that implementing a failure detection timer in DD has considerably decreased the required period for failure detection. Moreover, we clearly notice that using the new approach ANFD with adaptive timer results in a faster detection mean time.

Figure 4. Impact of failure detection timers on detection time.

2) Impact of adaptive FDT on packet loss rate: the second measurement was performed to evaluate the impact of failure detection on data loss. Since adaptive FDT allows a prompt detection and then a fast recovery, the new approach obviously decreases data loss. This is shown by the graph on **Erreur ! Source du renvoi introuvable.**where the packet loss rate in DD increases proportionally to the number of crashes. Notice that ANFD has considerably decreased this rate. Moreover, it gives the best results comparing to static *FDT* approach implemented in Fat2D.

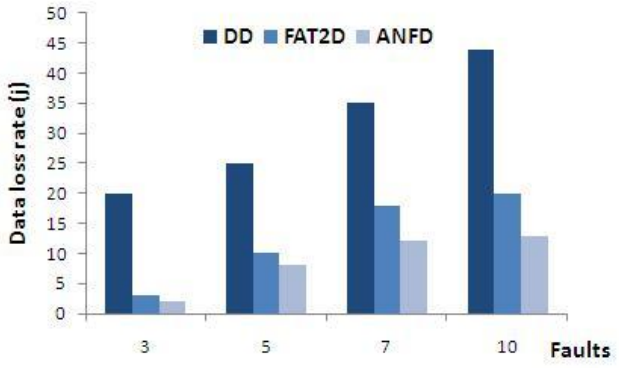

Figure 5. Impact of failure detection timers on packet loss rate.

3) Impact of failure detection timers on false positive suspicions: this measurement was performed on Fat2D vs. ANFD in order to evaluate the accuracy property. In this scenario there is no process crash (only intermittent failures are injected using a burst loss model based on a two state Markov chain). We vary simulation time in order to allow more intermittent failures for longer periods. For each experiment, the rate of false suspicions was measured. We notice in **Erreur ! Source du renvoi introuvable.**that the rate of mistaken suspicions is slightly greater in the case of constant timer (Fat2D), compared to the case of adaptive timer (ANFD). Actually, *SetFDT* function makes *FDT* updates after each round in order to tolerate more intermittent failures and prevent mistaken suspicions. This dynamic increasing offers better accuracy comparing to constant timer.

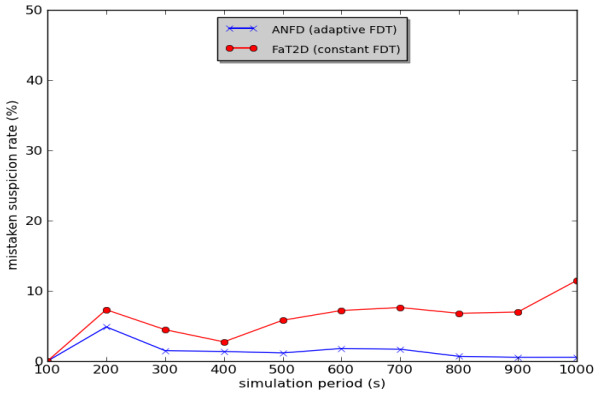

Figure 6. Impact of failure detection timers on accuracy property.

4) Impact of failure detection on energy consumption: the final measurement was performed for the battery use. As energy consumption is a critical constraint for WSN, it's important to evaluate the impact of implementing a failure detection mechanism on the amount of battery usage. We carried out the measurements using PowerTOSSIM-Z [?]. Results in **Erreur ! Source du renvoi introuvable.**show

that, even though ANFD (or even Fat2D) implements an additional mechanism for failure detection, they actually consume energy less than DD. Indeed, while DD wastes a great amount of energy to send lost data on faulty routes (after nodes' crashes), ANFD detects failures sooner and hence saves energy consumption by prompt path recovery. Yet, even if the difference of battery use is small, this result is promising since the overhead due to implementing an additional technique for failure detection compensate the overhead that would have been induced because of useless transmissions after node failures. However, our adaptive failure detection technique has the merit to save sensed data through fast route recovery. Without failure detection mechanisms, DD either looses data which decreases the quality of decision making at the application level or relies on higher transport protocol for eventual retransmission which would induce great overheads (energy, bandwidth, storage, computation, etc.).

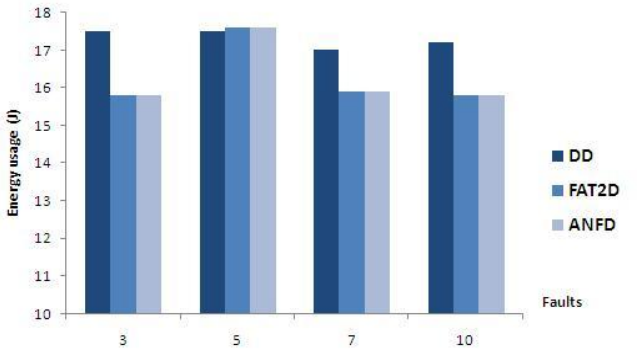

Figure 7. Impact of failure detection on energy consumption.

VI. CONCLUSION

In this paper, we have presented a new failure detection solution for WSN (ANFD); where processes interact only with their 1-hop neighbors using a Q/R scheme. The algorithm of ANFD is based on adaptive failure detection timer for crash suspicion. The proposed technique insures 1-hop completeness and 1-hop accuracy, two properties that we have introduced in this paper to take into consideration requirements and constraints of WSN. We carried out simulations using TinyOS2, and energy profiling using PowerTOSSIM-Z. The simulation results show that our solution decreases the failure detection time, packet loss ratio and suspicion fault positives. Moreover, we demonstrated that augmenting directed diffusion with our adaptive failure detection scheme allows not only to save valuable sensed data through fast route recovery, but also does not induce extra energy consumption overhead.

REFERENCES

[1] M. K. Aguilera, W. Chen, M. Kawazoe, and A. Wei, S. Toueg. Heartbeat: A Timeout-Free Failure Detector for Quiescent Reliable Communication. 1997.

[2] F. Z. Benhamida and Y. Challal: FaT2D: Fault Tolerant Directed Diffusion. The Fifth International Conference on Availability, Reliability and Security "ARES 2010–The International Dependability Conference". Poland, February 2010.

[3] F. Bonnet and M. Raynal: Looking for the Weakest Failure Detector for k-Set Agreement in Message-Passing Systems: Is πk the End of the Road? In Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems. (SSS '09). Springer-Verlag, 149-164, France, 2009.

[4] A. Boukerche, R.W.N. Pazzi, and R. B. Araujo. A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications. In Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM '04). ACM, New York, NY, USA, 2004. 157-164.

[5] T.D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. Journal of the ACM, 1996.

[6] B. Devianov and S. Toueg. Failure detector service for dependable computing. In Proc. of the First Int. Conf. on Dependable Systems and Networks, pages 14–15, juin 2000.

[7] E. O. Elliott. "estimates of error rates for codes on burst-noise channels". The Bell System Technical Journal, 42:1977–1997, Sept 1963.

[8] C. Fetzer, M. R,aynal and F. Tronel. An Adaptive Failure Detection Protocol. In Proceedings of the 2001 Pacific Rim international Symposium on Dependable Computing. Page 146. PRDC. IEEE Computer Society, Washington, DC. December 17 - 19, 2001.

[9] E. N. Gilbert. "Capacity of a burst-noise channel". The Bell System Technical Journal, 39:1253–1265, Sept 1960.

[10] I. Gupta, T. D. Chandra, and G. S. Goldszmidt. On scalable and efficient distributed failure detectors. In Proc. of the twentieth annual ACM symposium on Principles of distributed computing, pages 170–179. ACM Press, 2001.

[11] W.R. Heinzelman, A. Chandrakasan and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In proceedings of 33rd annual Huwaii international Conference. IEEE Proc. - Maui : Hawaii Int, Janvier 2000. pages. 1–10.

[12] C. Intanagonwiwat, R. Govindan, and D.Estrin. "Directed diffusion: a scalable and robust communication paradigm for sensor networks". In ACM Mobicom, pages 56–67. ACM, Aug 2000.

[13] M. Larrea, A. Fernàndez, and S. Arévalo: Optimal implementation of the weakest failure detector for solving consensus. In Proc. of the 19th Annual ACM Symposium on Principles of Distributed Computing, pages 334–334, NY, ACM Press, July 16–19 2000.

[14] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: "accurate and scalable simulation of entire tinyos applications". In Proceedings of the 1st international conference on Embedded networked sensor systems, November 05-07. 2003.

[15] A. Mostefaoui, E. Mourgaya, and M. Raynal. Asynchronous implementation of failure detectors. In Proc. of Int. Conf. on Dependable Systems and Networks, June 2003.

[16] A.T. Mzrak, Y. Cheng, K. Marzullo and S. Savage. Detecting and Isolating Malicious Routers. IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 3, july-september 2006.

[17] P. Sens, L. Arantes, M. Bouillaguet, V. Martin and F. Greve. Asynchronous Implementation of Failure Detectors with partial connectivity and unknown participants. Report n°6088. INRIA.December 2007.