Adaptive failure detection in low power lossy wireless sensor networks

Fatima Zohra Benhamida, Yacine Challal, Mouloud Koudil

Adaptive Failure Detection in Low Power Lossy Wireless Sensor Networks

Fatima Zohra Benhamida[a], Yacine Challal[a,*], Mouloud Koudil[a]

[a]*Ecole nationale Supérieure d´Informatique, Algiers, Algeria.*

Abstract

In this paper, we investigate the use of failure detectors (FD) in Wireless Sensor Networks (WSN). We provide a classification of FD with respect to some WSN criteria. The focus will be on energy depletion and lossy links. We propose then a new general FD model tailored to WSN constraints, called Adaptive Neighborhood Failure Detector for Low-power lossy WSN (AFDEL). AFDEL provides adaptive local failure detection robust against packet loss (intermittent failures) and saves the use of energy, bandwidth and memory storage. Furthermore, we introduce in AFDEL model an adaptive timer strategy. This strategy offers the possibility to customize the dynamic timer pattern with respect to application requirements in terms of completeness and accuracy. We illustrate the use of this strategy by proposing three failure detection techniques based on AFDEL general model. As a pat of this work, we give a stochastic based approach for timer adaptation. We evaluate

*Corresponding author: Y. Challal, Laboratoire de Méthodes de Conception de Systèmes. Ecole nationale Supérieure d´Informatique, Algiers, Algeria.

Email addresses: f_benhamida@esi.dz (Fatima Zohra Benhamida), y_challal@esi.dz (Yacine Challal), m_koudil@esi.dz (Mouloud Koudil)

all techniques using implementation on MiXim/Omnet++ framework. The overall experiments show that AFDEL achieves better trade-off between accuracy/completeness and detection/recovery period compared to static timer approach FaT2D [1] and HeartBeat timer-free technique [2]. Moreover, using adaptive timer strategy in local interaction/detection makes it possible to reduce energy consumption and the overall exchanged data overhead.
Keywords:
failure detector model, adaptive timer-based techniques, lossy links,
low-power WSN.

---

1. Introduction

The detection of process crash in distributed systems is a critical problem designers have to cope with. Thus, failure detectors have been introduced as a fundamental service able to help in the development of resilient distributed systems. Its importance has been revealed by Chandra and Toueg [3] who proposed the abstraction of unreliable failure detectors in order to circumvent the impossibility of consensus in asynchronous environments. Unreliable failure detectors, namely FD, can informally be seen as a per process oracle which periodically provides a list of probable crashed processes. This concept has been implemented for distributed systems with wired connectivity and abundant energy. In this article, we consider the study of FD in wireless sensor networks, which is a special case of distributed systems with new constraints on resources and radio link quality.

Wireless sensor network (WSN) consists of spatially deployed autonomous

sensors, working cooperatively to monitor observed phenomena. It is an enabling technology for several applications such as surveillance of environmental conditions, battlefield tracking, medical assistance, etc. This kind of networks presents the following properties: (1) a node does not necessarily know all the nodes of the network due to storage capacity limitations. It can only send messages to its 1-hop neighbors, i.e. those nodes that are within its transmission range; (2) the network is not completely 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) links are prone to failures and may momentarily drop messages during transmission; (4) nodes are equipped with very small autonomous batteries. Communication protocols have to consider this limitation for efficient energy consumption; (5) WSNs are prone to a variety of malfunctioning conditions due to the potential deployment in uncontrolled and harsh environment and to the complex architecture and energy depletion.

Fault tolerance is, therefore, one of the critical issues in WSN. This requires a consideration of the ability to cope with node crashes by detecting failed nodes, isolating them and eventually recovering routes to which they belonged. Notice that implementing FD, initially proposed for distributed systems, requires a special design to respond to WSN constraints. Some researchers have proposed few failure detectors tailored to WSN. However, most of them rely on assumptions that are not realistic with respect to WSN constraints and operation environment.

In this paper, we shall survey the main failure detectors proposed for distributed systems and the possibility to apply them to WSN. For this reason,

we introduce a set of classification criteria to compare these different FD. The focus will be on energy depletion and lossy links for the reason that resource limitation and reliable communication represent one of the most relevant constraints for WSN [4] [5] [6]. Furthermore, we study the few proposed FD particularly designed for WSN. As we aim to enhance the failure detection paradigm for WSN, we propose a new FD class $\Diamond S^{al}$. This class adapts FD properties for dynamic systems using local interactions for failure notification. The contributions of our work are manifolds:

1. We introduce failure detection model specifically designed for WSN, respecting WSN constraints and limitations: energy, memory, wireless transmission limitations (packet loss, intermittent connectivity), non complete graph topology, etc.;

2. The message exchange pattern is based on local exchanged information among neighbors and not on global exchanges among nodes in the system. This allows to define efficient energy consumption strategy;

3. It tolerates intermittent failures due to lossy links and adapts crash detection procedure using an adaptive timer;

4. It offers the possibility to customize the timer adaptation technique to application requirements in terms of completeness and accuracy.

The paper is organized as follows: in the next section, we recall failure detectors background. Then, in section 3, we present classification criteria to study FD in the perspective to apply them to WSN. We rely on this classification to present related works. In the 4^{th} section, we introduce a new FD model that fits low-power lossy WSN. The general algorithm and its properties are

given in section 5. We explain its timer adaptation strategy. In section 6, we illustrate the use of this FD model and its dynamic timer by proposing three timer adaptation techniques. In the next section, we present performance evaluation analysis through extensive simulations using Omnet++/MiXim and comparison with respect to existing solutions. We end up this paper with conclusions.

2. Failure detector: definitions

In their seminal paper [3], Chandra and Toueg define unreliable failure detector as a per process oracle which periodically provides a list of probable crashed processes in the system. Authors have proposed eight classes of unreliable failure detectors formally characterized by two properties: Completeness and Accuracy. Completeness requires that FD eventually suspect every process that actually crashed, while accuracy limits the mistakes FD can make. The FD is unreliable in the sense that it may not suspect a crashed process or can erroneously suspect a process of having crashed while it is correctly working. The FD can later on remove the process from its list if it believes that it was a mistaken suspicion. The eight possible classes, given in Table 1, are defined from the following completeness properties:

1. Strong completeness: eventually, every process that crashes is permanently suspected by every correct process.

2. Weak completeness: eventually, every process that crashes is permanently suspected by some correct processes.

The two completeness definitions can be combined with the following four accuracy properties:

1. Perpetual strong accuracy: no process is suspected before it crashes.
2. Perpetual weak accuracy: some correct processes are never suspected.
3. Eventual strong accuracy: there is a time after which no correct process is suspected.
4. Eventual weak accuracy: there is a time after which some correct processes are never suspected.

In what follows, we classify some related works and then investigate the use of FD in WSN with respect to WSN constraints.

Table 1: Unreliable failure detector classes.

		Accuracy			
		Strong	Weak	Eventually strong	Eventually weak
Completeness	Strong	P	S	$\Diamond P$	$\Diamond S$
	Weak	Q	W	$\Diamond Q$	$\Diamond W$

3. Related Works

In this section, we review failure detectors proposed in the literature and analyze the ability to implement them for low-power lossy WSN. Our aim is to analyze how far each FD is suitable for WSN while considering WSN constraints and limitations. In addition to generic FD that have been designed for distributed systems, we consider FD that have been designed specifically

for WSN. Our review will follow classification criteria that we defined after deep analysis of existing solutions with respect to WSN constraints that would influence the design of appropriate FD. Namely, we will consider design decisions regarding detection paradigm, network connectivity, radio link failure, and energy consumption.

3.1. Failure Detection Paradigm

Many failure detection paradigms have been proposed according to system and network models. Some works implement keep-alive methods based on heartbeats, pings or application data messages. Both Aguilera et al. [2] and Rost et al. [14] have proposed FD solutions based on heartbeats. Heartbeat is a message periodically sent from monitored node to the failure detector to inform that it is still alive. If the heartbeat doesn't arrive before timeout expires, the monitored node is then considered as faulty. Besides, Sens et al. have proposed in [10][11] detection mechanism using ping. Ping is a request message continually sent from a failure detector to monitored node. Upon reception of this query, the node responds with an acknowledgment ACK. If the node fails to send the expected ACK, it is suspected of having crashed. Moreover, Memento [14] relies on application data exchange to monitor faults. In this case, no additional messages are handled for crash detection. Failure detectors will rather use specific-application information transmitted through the network to make suspicion. Note that we can classify these paradigms into two types, depending on timers: the timer-based failure detectors handle timeouts to make suspicions. It is generally implemented for synchronous and static systems where timer delays are fairly predictable.

Table 2: Panorama of Failure Detectors.

Failure Detector	**Paradigm**	**Connectivity**	**Links**	**Class**
Aguilera et al. [2]	- Heartbeat - Timer-free	- Partial - Neighbors	- Broadcast - Lossy	$\Diamond P$ adapted
Bonnet et al. [7]	Define 03 new classes based on P, Σ&Ω for anonymous systems	- Partial - anonymous	- Broadcast - Reliable	$A\Sigma$, $A\Omega$ &AP
Bonnet et al. [8]	new quorum-based class	Full	- Broadcast - Reliable	Π_k
Mostefai et al. [9]	- Query-Response - Timer-free	Full	- Broadcast - Reliable	$S/\Diamond S$
Sens et al. [10]	- Ping - Query-Response - Timer-free	- Partial - Neighbors	- Broadcast - Point-To-Point - Reliable	$\Diamond S$
Greve et al. [11]	- Suspicion-flooding - Ping - Timer-free	- Partial - Neighbors	- Lossy	$\Diamond S$
Greve et al. [12]	- Query-Response - Timer-free	- Partial - Neighbors	- Reliable	$\Diamond S^M$
Hayashibara et al. [13]	- Degree of confidence (threshold) - Heartbeat - watchdog	- Partial - Neighbors	Tested on LAN	$\Diamond P$
Rost et al. [14]	- Heartbeat - Application Data	- Partial - Routing Tree	Lossy	NA
Branch et al. [15]	- Streaming data	- Partial	- NA	NA
Chen et al. [16]	- Test on distance and dependency values - Hardware failure	- Partial	- Lossy	NA

Heartbeat paradigms often use timers awaiting the monitored node's message as proposed in [17][18]. However, the majority of failure detectors for asynchronous and dynamic systems use timer-free paradigm since the timeout estimation can't be given from the network model [2][9][10][11][12][19].

3.2. Network Connectivity

The design of FD solutions depends heavily on the considered network model. Each failure detector maintains a list of suspicions as a subset from all monitored nodes known by its own process. When a node is addressed as faulty by any FD module, this information must be propagated to other modules. Nevertheless, failure propagation is time consuming in large scale systems such as WSN. Different approaches were proposed to improve propagation time. Authors in [8][9] consider a complete graph; where every node is directly connected to all nodes in the system. This leads to immediate delivery of failure notifications. On the other hand, other FDs assume a partial connectivity, so that, only a subset of nodes is monitored. Authors in [2][12][18][10][11] monitor 1-hop neighbors. Moreover, Rost et al. propose in [14] a hierarchical failure detector to arrange nodes into a multi-level hierarchy and partition monitoring into small groups. Failures are reported along a tree to improve scalability.

3.3. Link failure

In WSN, communication channels are highly volatile. Link failures are mainly due to radio interference where packets transmission is prone to burst losses. Given that link failures are intermittent, failure detectors should be designed

in a way that allows distinguishing link failure from process crash. Nevertheless, all solutions in [7][8][12][9][16][10][17][11] consider reliable communication channels. Thus, any unsuccessful transmission is considered as a crash suspicion. Notice that, in WSN, link failure tolerance is a leading constraint to accurately identify crashes. Subsequently, when a failure detector considers unreliable communication systems, it must single out process crashes among unsuccessful transmissions due to intermittent faults. For this reason, authors in [2][12][14] consider lossy links in their failure detection design.

3.4. Energy consumption

Failures in WSN cannot be approached in the same way as in traditional distributed systems. Indeed, traditional network protocols are generally not concerned with energy consumption; since wired networks are constantly powered and wireless ad hoc devices can get recharged regularly. Besides, battery depletion is one of the major causes that yield to process crash. Therefore, the implementation of failure detection mechanism must be energy efficient. In [15], Branch et al. consider energy consumption as a vital constraint. Meanwhile, most of presented solutions (see Table 2) do not consider any resource constraint neither in their design nor during performance evaluation.

3.5. Discussion

In Table 2, we present a sample of FD solutions according to classification criteria introduced above. In addition to these factors, we study in Table 3 the advantages and drawbacks of each FD when applied for WSN. We notice

Table 3: Discussion of Failure Detectors in WSN.

Failure Detector	**Advantage**	**Drawback**	**Observation**
Aguilera et al. [2]	- Perfect system model for WSN - Update mistaken suspicion - Extend previous work with lossy links	- Resource demanding - Extra-Data exchange	Quiescent algorithm
Bonnet et al. [7]	- Anonymous - Asynchronous	- Lossy link intolerance	Consensus
Bonnet et al. [8]	Asynchronous	- Lossy link intolerance - Fully connected	k-set agreement
Mostefai et al. [9]	Asynchronous	Lossy link intolerance	f-covering
Sens et al. [10]	- Update mistaken suspicion - Mobility - Anonymous - Dynamicity	- Resource demanding - Lossy link intolerance	- f-covering - responsiveness & membership properties
Greve et al. [11]	- Mobility - Anonymous - Update mistaken suspicion	- Lossy link intolerance - Energy-demanding - Extra-Data	- Require at least d neighbors
Greve et al. [12]	- Query-Response mechanism - Local asynchronous interaction	- No simulation test, no algorithm proof - Generic solution for wireless networks	CH election, consensus
Hayashibara et al. [13]	- 2 new properties (accruement & unknown bound)	- Not designed for WSN (tested on LAN) - No energy-efficiency	Tested on LAN
Rost et al. [14]	- Decrease of false positive rate - Aggregation and incremental difference	- No energy consideration (analysis/evaluation) - Hardly scalable because of the routing tree constraint	
Branch et al. [15]	- Resources management (energy consumption, bandwidth) - WSN model	- Applied for data outliers only - No sensor failure detection	Can be tailored for byzantine failures
Chen et al. [16]	- Accuracy - WSN model	Only sensor hardware failures	

that most of current implementations of FD classes are based on an all-to-all communication approach; where each process periodically sends a heartbeat message to all processes [2][14]. As they usually consider a complete graph network model, these implementations are not adequate for dynamic and partially connected environments, such as WSN. Furthermore, FD are usually based on packet acknowledgment; 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.

Authors in [9] propose a timer-free asynchronous FD. 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 completely connected known nodes. Yet, this timer-free approach is not applicable to partially-connected networks such as WSN. Some works deal with the scalable nature of dynamic systems [2][20]. Nonetheless, few of them tolerate links failures [2][14][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) [7][8][9][10][11]. This may increase the risk of mistakes when FD suspect a process of having crashed while the packets were lost because of intermittent failure due to unreliable links. Even though resource management is a vital constraint in WSN, most of solutions take no consideration for energy consumption, bandwidth or memory storage. Authors in [15] proposed an energy-efficient failure detection mechanism. However,

their algorithm is designed for data outlier detection not for process crash.

Particularly, most general modules designed for distributed systems cannot be applied for WSN without modification. Therefore, it is hardly possible to use these FD given WSN constraints and limits. For this reason, some researchers have developed new FD tailored for WSN. Rost et al. present in [14] a health monitoring system for WSN (Memento). This protocol dedicated for WSN applications, uses a routing tree and heartbeat mechanism. It considers a lossy communication channel, with partial connectivity between neighbors.

After this presentation of related works and discussion, we introduce hereafter a new FD model tailored to WSN applications and constraints. We will present the general algorithm to define the new FD class, adapted to low-power lossy WSN. In addition to this abstract definition, we present three detection techniques based on the proposed model.

4. AFDEL: A New Failure Detector Model for Low-power Lossy WSN

4.1. 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_j must periodically send data message to P_i. Let us assume first that links are reliable which means that messages sent between correct

processes are successfully transmitted. As we want to define a timer-based failure detector, we use a *Failure Detection Timeout* (FDT_{ij}) in process P_i for neighbor P_j: if FDT_{ij} runs out before receiving the expected response, P_i suspects that P_j has crashed. FDT_{ij} value is initiated according to application parameters; such as periodic sending interval, 1-hop transmission latency, etc.

However, the situation changes if, in addition to process crash, packet loss and 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 intermittent failures (lossy links, congestion...). Moreover, the pattern of packet loss is not stable during all the lifespan of the network. Actually, changes may occur according to obstacles, geographic situation, sensors suppression, deployment or mobility. For this reason, we need to update FDT_{ij} value following the dynamic lossy pattern of sensor nodes. Besides, if the network contains more than two nodes, every node's FD must detect failures locally then notify its suspicion to neighbors.

In this paper, we explore the use of adaptive timer FDT to circumvent this obstacle: we define a new failure detector as a general model for WSN. We call this model AFDEL; **A**daptive **F**ailure **D**Etector for **L**ow power lossy WSN. AFDEL brings two new properties of accuracy and completeness: one-hop accuracy and one-hop completeness. Moreover, we introduce an adaptive timer strategy. This strategy makes use of a stochastic approach to determine dynamic timer FDT taking into consideration loss ratio, links state, transmission delay, etc. We illustrate the use of this strategy by proposing three detection techniques based on AFDEL general model. Each technique

defines different application requirements in terms of completeness and accuracy. It is important to notice that FDT is defined between two consecutive neighbors 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. We take into consideration all notifications in route maintenance for the overall nodes of the WSN. We present thereby, a new failure management class for WSN, namely Eventually Strong class (see section 2) for Adaptive Local detection; $\Diamond S^{al}$. This class adapts the properties of eventually strong class $\diamond S$ to a dynamic system using local interaction for failure notification. 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). This list is piggybacked on periodic data messages using neighborhood interaction to avoid extra overhead that would have been induced if dedicated signaling packets were used.

4.2. System model

4.2.1. Network model

We consider a wireless sensor network system 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 radio 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. Nodes P_i and P_j are connected by a link $(P_i, P_j) \in E$ if and only if they are within their wireless transmission range. In this case, P_i and P_j are considered neighbors.

Assumption 1. *We suppose that the remaining nodes in the same neighborhood where a fault might occur, can still communicate with each other. Obviously, this assumption is weaker than f-covering property used in many related works (e.g. [9] and [10]). Our assumption is equivalent to 2-node connected graph restricted to one neighborhood instead of the complete network*[1].

4.2.2. Failure model

We assume that sensors may crash. Moreover, links between two 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 node fails to send some packets, this loss should be considered as intermittent failure. However, if a process crashes, it will never belong to the network anymore; and has to be notified to all its neighbors. Hence, even if a failure detector 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.

[1] This is always satisfied except in the case where the faulty node isolates the network in two separate sub-graphs (i.e. the faulty node was a Single Point of Failure).

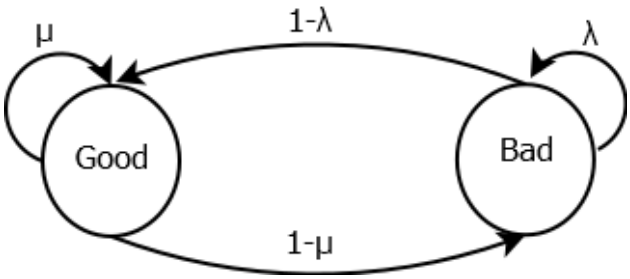

Figure 1: Gilbert-Elliot packet loss model.

4.2.3. Packet loss model

It is widely accepted that wireless communication transmission is an error prone channel. A model of error characteristic in wireless networks was suggested by Gilbert and Elliott [21]. It is a 2-state Markov model (Good and Bad)[2] as shown in figure 1. At each packet interval, the channel (i.e. radio link) changes to a new state with transition probabilities. The number of successive packet losses can be represented with a geometric distribution of variable BL (Burst Length), which defines the time sojourn in Bad state (i.e. successive packet losses). Hence, the mean state sojourn time of transmission errors (duration of being in Bad state) is given by $E(BL)$:

$$E(BL) = 1/\lambda \tag{1}$$

Where:

λ is the probability for two consecutive packet loss.

[2]Good state refers to successful packet delivery and Bad state refers to packet loss.

Then, given $E(BL)$, it is possible to define a burst loss limit (BLL) which must be tolerated by the FD. BLL is a function of $E(BL)$, and eventually other network and/or traffic parameters: In other words each FD timeout (FDT) must be greater than the defined tolerated burst loss limit (BLL). Thus, the timeout value covers a period of successive packet losses.To this end, we define the value of BLL hereafter:

$$BLL = E(BL) + V(BL) \tag{2}$$

Where:

- BLL: Burst Loss Limit;
- $E(BL)$: Mean Burst Length;
- $\sqrt{V(BL)} = \frac{\sqrt{1-\lambda}}{\lambda}$: Standard deviation (λ is the probability for two consecutive packets loss).

In BLL, we empirically add up standard deviation value to the mean burst loss $E(BL)$ in order to tolerate reasonable long burst losses. This allows to better separate intermittent failures due to lossy channels from other failures causes.

4.2.4. Data gathering model

We consider two types of data gathering models, namely time-driven and query-driven. First, we define the query-driven model, also known as a Query-Response routing mechanism. This kind of routing protocols is mainly

based on network interrogation; where a collector node (also known as a sink) periodically broadcasts a query for data (sensed or stored) to all network 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 response to all nodes from which it has received the corresponding query. Eventually, a single path is elected to send periodically data from source to sink. Directed Diffusion [22] is a query-driven protocol. On the other hand, time driven model uses a periodic scheme for data dissemination. The routing protocol defines a slot for each neighbor to send data within a forwarding interval. LEACH [18] is a time driven protocol that uses TDMA schedule for its routing scheme. Notice that Query-Response and Time-driven models cover a very large set of routing protocols in WSN[3].

4.3. Interaction between system model and FD model

In figure 2, we illustrate the interactions between system models with our FD model: AFDEL 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) periodic messages to (resp. from) AFDEL module in order to update (resp. deliver) the list of suspicions. Thus, every message piggybacks information about nodes' crashes. Moreover, list updates (insert suspected process or delete

[3]The third model is Event-driven.

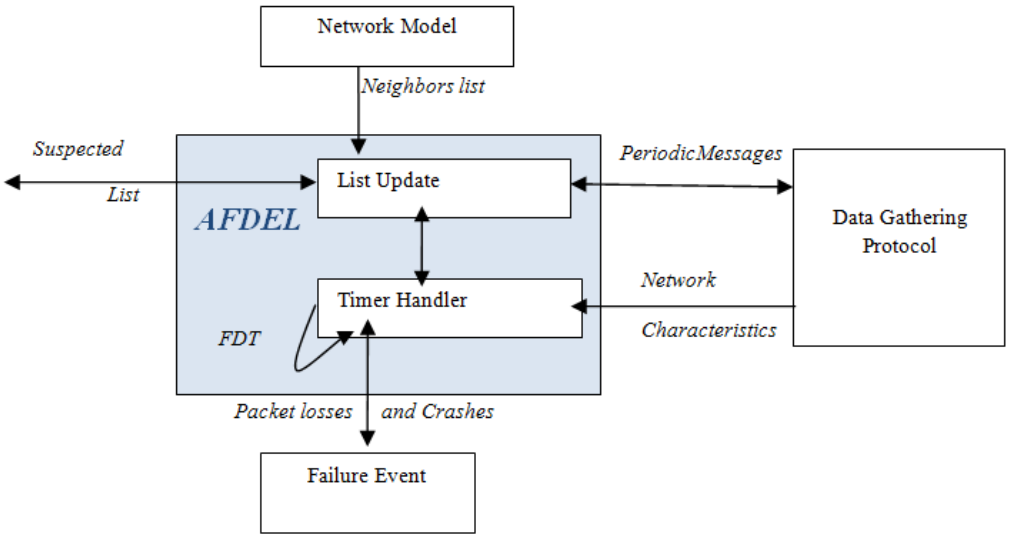

Figure 2: Interactions between AFDEL and system modules.

mistaken suspicion) will lead to a new FDT refresh in order to give more accurate notifications in next rounds (prompt detection vs. less mistakes). FDT is then updated according to the new proposed techniques following the proposed general model. This update is triggered by the Timer Handler in our AFDEL module.

We recall, this model can be adapted to several applications according to the chosen detection technique. The adaptation of FDT can be done in several ways and depends on the actual dynamics in the network. In section 6, we will define three FDT adaptation techniques and present performance evaluation and comparison through simulations.

5. AFDEL: Algorithm and properties

5.1. AFDEL algorithm

We recall that we consider a periodic data gathering model (time driven and query driven). The algorithm proceeds by rounds. At each round, a node sends a message to its neighbor(s) until it possibly crashes. The basic principle of our model is to piggyback failure suspicion on the data messages, and dispatch this information hop by hop. Hence, in addition to application data, we add up $Suspected_i$ list to each message. AFDEL module for process P_i is described in figure 3. It is composed of two main primitives (send & receive) and two procedures (UpdateList & Timer.Fire): The primitive AFDEL.Send(msg) includes $Suspected_i$ set in the message msg originated from the upper layer (line 2). After piggybacking the information, P_i sends the new message to the downward layer (line 3) in order to forward it to the destination node. Then, P_i initializes a new timer for every correct known neighbor; say FDT_{ij} for node P_j awaiting for the next message (lines 4-7). If any FDT_{ij} runs out , Timer.Fire(P_j) is triggered (line 23); that is, P_i suspects P_j of being faulty. The new suspicion information is then inserted in $Suspected_i$ (line 24). This suspicion will be included in the next data message when AFDEL.Send is executed again. The second primitive AFDEL.Receive(msg) aims to update suspected set according to the one piggybacked on the received message delivered from the downward layer. It tries also to discover previous mistaken suspicions in order to delete the corresponding nodes from suspected list. First, P_i retrieves information, namely, suspected set with its source node P_j (line 10), then it stops FDT_{ij}, since

P_i has successfully received message from P_j (line 11). P_i checks then if P_j (the source) was recently suspected in order to delete it from $Suspected_i$ set (line 12). After that, NetLayer.Receive sends the new message msg to the upward layer (line 15). Finally, P_i calls the UpdateLists procedure in order to treat the received information about suspicions (and mistakes) in P_j's message. The two loops of procedure UpdateLists handle information about suspected (respectively erroneously suspected) processes. Thus, for each suspected node P_s included in $Suspected_j$ set, P_i includes P_s in its $Suspected_i$ (lines 16-18). Moreover, if there is a node P_m notified as a previous mistaken suspicion by P_j, P_i removes P_m from its $Suspected_i$ (lines 19-21). The mistake is detected if $Suspected_j$ does not contain P_m while the latter is still in $Suspected_i$ list.

5.2. Properties of the new failure detection class $\Diamond S^{al}$

Any failure detector FD is characterized by two properties: completeness and accuracy (see section 2). 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 for AFDEL class new completeness and accuracy properties that better suit the conventional system model:

5.2.1. Definition 1: (1-hop-Completeness)

There is a time after which every process that crashes is suspected by all its correct neighbors.

Proposition 1. $\forall\, P_i crashes, \forall\, P_j \in Neighbors_i, \exists\, \tau \in T : \forall t > \tau, P_i \in Suspected_j$

1: **procedure** AFDEL.SEND(msg)
2: $Piggyback(msg, Suspected_i, P_i)$
3: $DLayer.Send(msg)$
4: **for all** $P_j \in Neighbors_i \setminus Suspected_i$ **do**
5: $FDT_{ij} \leftarrow setFDT(P_j)$
6: $Timer.schedule(P_j, FDT_{ij})$
7: **end for**
8: **end procedure**
9: **procedure** AFDEL.RECEIVE(msg)
10: $Retrieve(msg, Suspected_j, P_j)$
11: Timer.stop(P_j)
12: **if** $P_j \in Suspected_i$ **then**
13: $Suspected_i \leftarrow Suspected_i \setminus P_j$
14: **end if**
15: $NetLayer.receive(msg)$
16: **for all** $P_s \in Suspected_j$ **do**
17: $Suspected_i \leftarrow Suspected_i \cup \{P_s\}$
18: **end for**
19: **for all** $P_m \in Suspected_i \setminus Suspected_j$ **do**
20: $Suspected_i \leftarrow Suspected_i \setminus \{P_m\}$
21: **end for**
22: **end procedure**
23: **Task:** Timer.Fire(P_j)
24: $Suspected_i \leftarrow Suspected_i \cup \{P_j\}$

Figure 3: AFDEL Algorithm.

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

5.2.2. Definition 2 (1-hop-Accuracy)

There is a time after which some correct processes are never suspected by any correct neighbor.

Proposition 2. $\exists\ P_i correct, \forall\ P_j\ \in\ Neighbors_i, P_j\ correct, \exists\ \tau\ \in\ T :$ $\forall t\ >\ \tau, P_i\ \notin\ Suspected_j$

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

With these new properties, we define a new FD class; namely, $\Diamond S^{al}$ eventually strong class limited to 1-hop interaction. The merit of this new class 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 neighborhood of the faulty node remains connected (assumption 1). Therefore, due to energy, storage and bandwidth limitations, every node monitors only its direct neighbors, even though the information is initiated by a further node. 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 of three new techniques based on the proposed FD model.

Proposition: AFDEL belongs to $\Diamond S^{al}$, i.e. AFDEL guarantees the following

properties:

- 1-hop-completeness
- 1-hop-accuracy

5.2.3. Proof

Consider that the most recent status about a process P_k is stored in $Suspected_i$ set of correct process P_i. From the algorithm in figure 3 each correct process P_i will execute line 2 to piggyback additional information about its suspected list using the periodic data gathering scheme. P_i sends then its message containing P_k in $Suspected_i$ set. Upon reception of such a message by neighbor P_j, P_j executes AFDEL.Receive primitive to retrieve information from received packet then update its own list. P_j executes then lines 16-18 to add P_k in $Suspected_j$. In the next round, P_j, the same as P_i, must broadcast this new status regarding P_k in its set. Using the same mechanism, P_j must retrieve a previously suspected P_k once it receives most recent information from P_i notifying the mistake (lines 19-21). Given assumption 1, all 1-hop neighbors eventually add (resp. retrieve) P_k in (resp. from) their Suspected set. This assures 1-hop-completeness and 1-hop-accuracy properties all over the network.

5.3. Timer adaptation strategy

We recall that AFDEL uses adaptive timer strategy to detect failures. Actually, AFDEL dynamically updates FDT respecting the application requirements in terms of completeness and accuracy, considering in-time changes

in the system; such us transmission failures, communication protocol behavior and network dynamics. Notice that there is no general formula since it depends on the choice of the routing protocol and its parameters. In this abstract definition, we don't precise any detection mechanism. The motivation behind introducing a generic strategy instead of defining a special formula for timer adaptation, is to offer the possibility to each application to customize the dynamic pattern according to expected accuracy and completeness performance. However, to better understand the utility of the new FD, we illustrate more the use of its strategy by proposing three detection techniques based on the generic model. In what follows, we propose then three FDT adaptation techniques that can be used by AFDEL. Later, we provide performance analysis and comparisons through simulations and illustrate the impact of network dynamics model and the used FDT adaptation technique on the overall performances.

6. Timer Adaptation techniques

We define three FDT adaptation techniques, namely ASAT, CSAT and HAT. Let's first define by SAT all detection techniques based on a Stochastic Adaptive Timer to notify failures. SAT mechanism uses in-time information about changes in the same routing path. This will allow adapting timer value according to network dynamics during the application run. We define now, two main techniques based on SAT mechanism; ASAT and CSAT. ASAT aims to notify all crashes with less false positive rate. Clearly, ASAT tries to enhance the accuracy property. Thus, we call it *Accuracy-aware Stochastic Adaptive*

Timer technique. However, CSAT seeks to make in-time monitoring by notifying the sooner any process crash. Obviously, CSAT may increase the false positive rate. Yet, prompt detection is the goal behind this mechanism. We call it then *Completeness-aware Stochastic Adaptive Timer* technique. Conversely to both SAT techniques, the third one is based on the changes of the nodes participating in the routing protocol and their position in the network area. Therefore, we define another adaptive timer based on the hops number that separate the monitored node from the sink (final destination node), namely HAT; *Hop-based Adaptive Timer*. We consider this technique since sink neighbors are more prone to failures because of congestion and high probability of message loss. Clearly, there is more congestion risk around sink neighbors than for distant nodes. This is due to the amount of sent data from all over the network toward the sink. HAT tries then to minimize false positive by separating intermittent failures from process crash according to node distance from the sink. We explain further the motivation behind each proposal. Before that, we give in Table 4 the common parameters used to implement these three techniques.

6.1. Stochastic Adaptation Techniques

We first recall that SAT techniques are based on stochastic modeling of network dynamics. At each round, we use statistics from previous rounds to update the timer according to dynamic changes in the routing protocol. We first explain in figure 4 the algorithm to set and update timer for SAT techniques. Later on, we add separately the difference in both ASAT and CSAT mechanisms. In the beginning, since there is no former communication

Table 4: Variables definition for Timer Adaptation

Variable	**Definition**
FDT	Failure Detection Timer: to be defined and updated by the algorithm.
I_f	Forwarding Interval: defines interval of sending periodic data packets
TBL	Tolerated Burst Loss: defines delay to tolerate for intermittent failures following the burst loss model BLL (see section 4)
SHC	Sink Hops Count: defines number of hops separating the node from the sink.
WDR	Wrong Detection Rate:defines the false positive rate during detection.
TWD	Tolerated Wrong Detection: defines the maximum degree of accepted wrong detection
T_r	Reliability Threshold: introduced by the user in order to define the reliability of the failure detector allowing faster detection.

to make statistics from, the first FDT value is initially set according to hypothetical values of transmission interval and latency (line 2). Then, FDT is updated using the effective loss rate regarding received data messages during the application run. Specifically, it is increased if the wrong suspicion has exceeded the *Tolerated Wrong Detection* rate TWD (lines 7 & 8), which is a threshold that we define to control the accuracy of the FD. If this threshold is reached, AFDEL must enlarge the waiting period before notifying failures.

```
1: Init
2: FDT ← FDT_0
3: WDR ← 0
4: define(TWD)
5: define(T_r)
6: procedure SetFDT
7:     if WDR ≥ TWD) then
8:         Increase(FDT)
9:     else if (1 − WDR) ≤ T_r then
10:        Decrease(FDT)
11:    end if
12: end procedure
```

Figure 4: Stochastic Timer adaptation algorithm.

However, when FDT satisfies the desired TWD, AFDEL tries to enhance crash notification performance. Actually, AFDEL decreases back timer value if it doesn't reach yet the predefined reliability threshold (T_r) (line 9). T_r is introduced by the application user in order to define the desired reliability of the failure detector allowing faster detection. FDT performance depends on the choice of the predefined parameters (T_r, TWD) on the one hand and both Decrease(FDT), Increase(FDT) functions on the other hand. The markov chain in figure 5 illustrates the stochastic timer process. Our

approach for SAT techniques is inspired by control-theoretic adaptation similar to those widely used in Internet, such as AIMD or MIAD best known for TCP congestion avoidance [23]. Hence, we propose two techniques based on SAT mechanism where time increase/decrease is differently implemented according to application requirements.

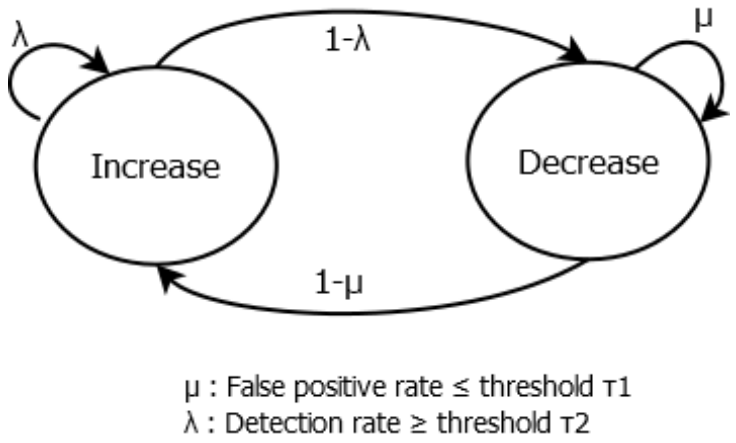

Figure 5: Stochastic timer update process.

6.1.1. ASAT

By ASAT technique, we aim to notify crashes with good accuracy (**A**ccuracy-aware **S**tochastic **A**daptive **T**echnique). This approach seeks to reduce the mistaken suspicions rate. For this reason, we use Multiplicative Increase Additive Decrease algorithm(MIAD) [23]. MIAD combines exponential growth of timer value with linear reduce to satisfy a minimum reliability in crash detection. Hereafter, the general formula for MIAD technique at time period

$t+1$:

$$FDT\,(t+1) = \begin{cases} \alpha \times FDT\,(t) & if\ Increase \\ FDT\,(t) - \beta & if\ Decrease \end{cases} \tag{3}$$

Let us first explain the objective of using MIAD for timer update. Suppose that the failure detector has made many mistaken suspicions, so that the false suspicion rate WDR exceeds the tolerated threshold ($WDR > TWR$). According to algorithm in figure 4, we increase the timer. Since ASAT aims to make fewer mistakes, we multiply the timer by $\alpha(\alpha > 1)$. Moreover, the parameter α depends on local stochastic information observed during the application run, namely WDR and TWR. Hence, the exponential increase of the timer depends on the effective false positive rate that induces its growth. However, in Decrease(FDT) function, the timer is reduced using parameter β ($\beta > 1$). As for β value, it is deduced from the local notification delays during crash detections. Notice that Decrease(FDT) is triggered once the FD is under the required reliability threshold (line 9 in figure 4). For the sake of simplicity, we define the reliability as the rate of correct suspicions; namely $1 - WDR$. By aggressively increasing the timer value upon false suspicions control, ASAT expands its waiting time to tolerate more intermittent failures and then, avoid false suspicion. Whenever the crash detection rate subsides, the timer is decremented according to the mean value of notification delay to quickly overcome unreliability.

6.1.2. CSAT

We propose **C**ompleteness-aware **S**tochastic **A**daptive **T**echnique (CSAT), for prompt crash detection. Thus, completeness takes the advantage on ac-

curacy. The purpose is to quickly notify any crash failure. For this reason, we use Additive Increase Multiplicative Decrease technique [23]. Inversely to MIAD approach, AIMD combines linear growth of timer value with exponential reduce as given in the formula hereafter at time period $t+1$:

$$FDT(t+1) = \begin{cases} \beta + FDT\,(t) & if\ Increase \\ \alpha \times FDT\,(t) & if\ Decrease \end{cases} \tag{4}$$

As for ASAT with AIMD, the timer increase, respectively decrease, depends on reliability in crash notification, respectively the false positive rate (see figure 5). However, CSAT aggressively diminishes the timer value to quickly detect any crash failure. For this reason, we multiply FDT by parameter α ($\alpha < 1$) which is defined according to the effective false positive rate. The objective behind this choice is that CSAT decreases the timer to promptly detect crashes regarding the actual false positive rate. Meanwhile, to avoid mistaken suspicions, CSAT increases FDT when it reaches a required reliability. For this, we add up to FDT parameter β ($\beta > 1$), which depends on the stochastic information during the application run (e.g. we consider the mean period value of delayed delivery that caused some of the previous false notifications).

6.2. HAT technique

In the Hops-driven Adaptive Timer based technique, the timer update depends on the nodes in the routing path and their distance from the sink. In addition to packet loss, mistaken suspicion can be made because of transmission delay due to congestion. To avoid notifying this as failure, we must give

more time to nodes according to the congestion risk in their region. Notice that there are more data packets as we get closer to the sink, because all sources send their messages toward the final destination (i.e. the sink). For this reason, HAT defines a timer for each node following equation 5. Notice that this timer represents a period to tolerate burst loss (intermittent packet loss). Then, we add up a second period inversely proportional to the distance separating the monitored node from the sink. We found convenient to set the maximum value (for 1-hop sink neighbors) to one forwarding interval I_f.

$$FDT = TBL + I_f/SHC \tag{5}$$

Where:

TBL: Tolerated Burst Loss.
I_f: Forwarding Interval.
SHC: Sink Hops Count.

7. Analysis of the AFDEL techniques

In this section, we study and evaluate the behavior of our dynamic AFDEL and the three proposed adaptation timer techniques. We start by comparing the general mechanism of our timer-based model with a timer-free FD. Then, we value the importance of updating timer and the impact of each proposed dynamic technique; namely CSAT, ASAT and HAT. To this end, we have chosen HeartBeat [2] as a timer-free FD. The failure detector module of HB at a process p outputs a vector of counters, one for each neighbor q of p. Each process periodically sends an 'I-am-alive' message (a "heartbeat")

and every process receiving a heartbeat increases the corresponding counter. Thus, if neighbor q does not crash, its counter increases with no bound. If q crashes, its counter eventually stops increasing. For failure detection, HB counts the total number of heartbeats received from each process, and considers that neighbor q has crashed when the counter stops increasing. HB outputs these "raw" counters to applications without any further processing or interpretation.

7.1. Simulation model

We carried out the simulations using MIXIM; an Omnet++ modeling framework [24]. The simulation model is resumed in Table 5. We consider three network topologies: random, grid ans star. By varying the area of nodes deployment, the network density may vary (i.e. number of neighbors for each node). In order to illustrate the outcome of using adaptive failure detection, we consider several simulation tests using two main scenarios. In scenario 1, we vary the number of total nodes (20 to 200 nodes), while in scenario 2, we change the number of crashes for the same network (10% to 50% of total nodes). We repeat every configuration for 50 iterations, and then we calculate the mean value for each performance metric. As a periodic data communication model, we consider Directed Diffusion routing protocol, DD. In our simulation tests, we compare DD-HB (DD augmented with timer-free HB), FaT2D [1] (DD augmented with static timer FD) with three variants of the new proposed model AFDEL, depending on the used timer adaptation technique: AFDEL-ASAT, AFDEL-CSAT and AFDEL-HAT. All AFDEL timer-based techniques start with the same timer FDT_0 as used in static-

timer FaT2D. However, FaT2D keeps the same value during simulation run, while each AFDEL technique adapts its own FDT according to equations and algorithm given in section 6. the motivation behind this comparison is to evaluate the following potential failure detection mechanisms:

1. Timer-free mechanism: this category regroups solutions that manage failures without using timeout [25].

2. Static timer-based mechanism: in this category, proposed solutions use timer to detect failures. However, this timer is fixed to the same value during the network lifespan [26] [27] [28]. For reason of simplicity, we consider in our simulation FaT2D, a static-timer based failure detection technique proposed for Directed Diffusion.

3. Adaptive timer-based mechanism: our proposed techniques are based on dynamic values of timeout. We will evaluate the use of this adaptive mechanism comparing to both timer-free and static timer-based mechanisms.

7.2. Performance metrics

During our experiments, we were interested in six main metrics classified in three main categories:

1. Failure detector properties: in this category, we test Accuracy and Completeness properties. This will allow classifying the proposed techniques according to their performance regarding the false positive rate (accuracy) and the detection rate (completeness).

Table 5: Simulation scenarios.

Scenario	**Nb nodes**	**Nb faults**	**Data Gathering Protocol**	**FD**	**Topology**	**Simulation period/ Iterations**
Scenario 1	[20-200]	10% of total nodes	Directed Diffusion	- HB - FaT2D - CSAT - ASAT - HAT	- Random - Grid - Star	5h / 50
Scenario 2	100	[10%-50%] of total nodes	Directed Diffusion	- HB - FaT2D - CSAT - ASAT - HAT	- Random - Grid - Star	5h / 50

2. Detection and recovery performances: every detection mechanism aims to notify crashes in the network in order to recover the routing path and stop the failure effect. For this reason, Detection and Recovery Periods are performed with Packet Loss Rate. These measurements test how fast is the detection technique and its impact on data message loss.

3. Resource management: since WSN are resource-limited, we deem necessary to evaluate and compare both Energy Consumption and Overhead (amount of messages in the network) for each implemented technique.

7.3. Simulation results

We discuss hereafter some of the simulation results. Graphs in figure 6 and figure 7 show completeness performance using both scenarios (nodes variations and crash rate variation). Obviously, all three proposed techniques in

AFDEL give results greater than both timer-free HB and static timer-based FaT2D. Moreover, CSAT and HAT offers the best rates (70% to 95% in scenario 1). Since timer adaptation strategy is application aware, and since CSAT aims to make prompt detections, the dynamic timer pattern allows then to enhance completeness property and gives greater results compared to other techniques. Surprisingly, HAT gives great results, especially for small networks and minor crash rate. This explains that setting timer value according to number of hops allow covering a great completeness rate. We have noticed during simulation, that for the same configuration, the mean timer value in HAT is less than values for other remaining techniques (including CSAT). Obviously, considering node position from the sink allows to enhance global detection rate. However, the timer-free HB detects only about 50% of the total faults in the network. Since HB can't make any decision until the gathered information from all the nodes is sent, the detection mechanism is very slow. Notice that completeness performance for all techniques is slightly decreasing while the network size is getting larger. This is mainly due to the increasing probability of dynamic changes for the routing paths. As the network size is bigger, there is more chance to participate other nodes for data transmission. Subsequently, this may cause to elect a new routing path before even any FD mechanism can detect some eventual faults in the previous path. Furthermore, we have noticed during simulation that completeness in timer-free method HB has the lowest value for scenario with small number of crashes in the network. This is due to the central decision of failure detection and notification. The base station promptly notifies failures when there are many packets lost due to many crashes in the network (figure 7).

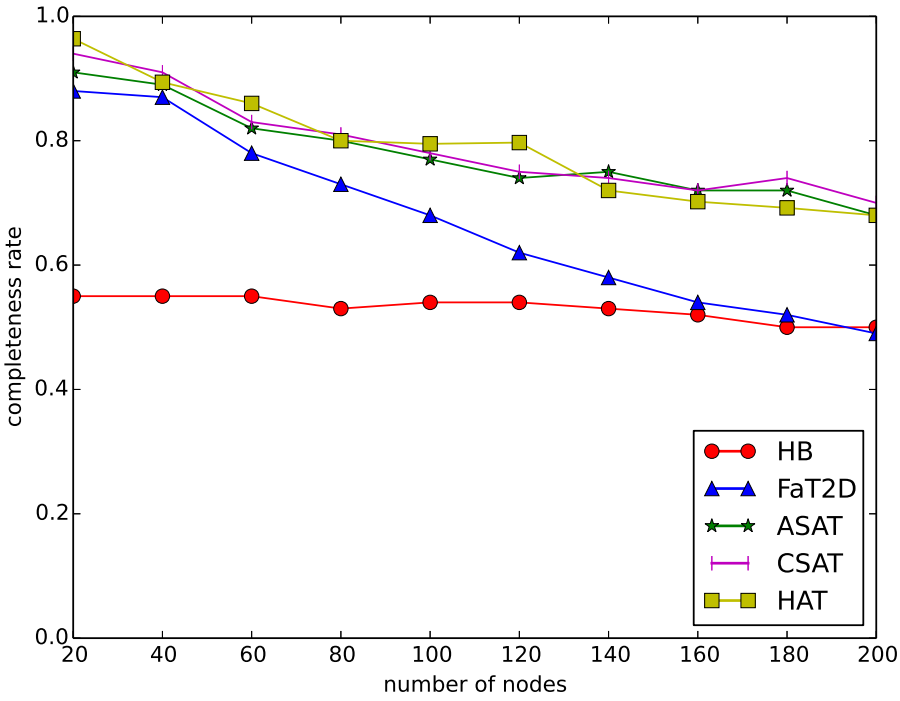

Figure 6: Completeness evaluation VS number of nodes (scenario1).

On another hand, ASAT gives the greatest accuracy rate. Clearly, this results from the formula of the timer update in ASAT which aims to minimize mistaken suspicions. We recall that ASAT timer adaptation strategy is based on additive decrease multiplicative increase AIMD, which promotes to enlarge timer values. Consequently, this will avoid false suspicions, and then, offer great accuracy rate. Yet, HAT offers remarkable performance as shown in both figure 8 and figure 9. We recall that HAT gives the lowest mean value of adaptive timer; which offers great completeness rate. Besides, HAT enlarges the timer mostly in regions prone to intermittent failures. Actually, inter-

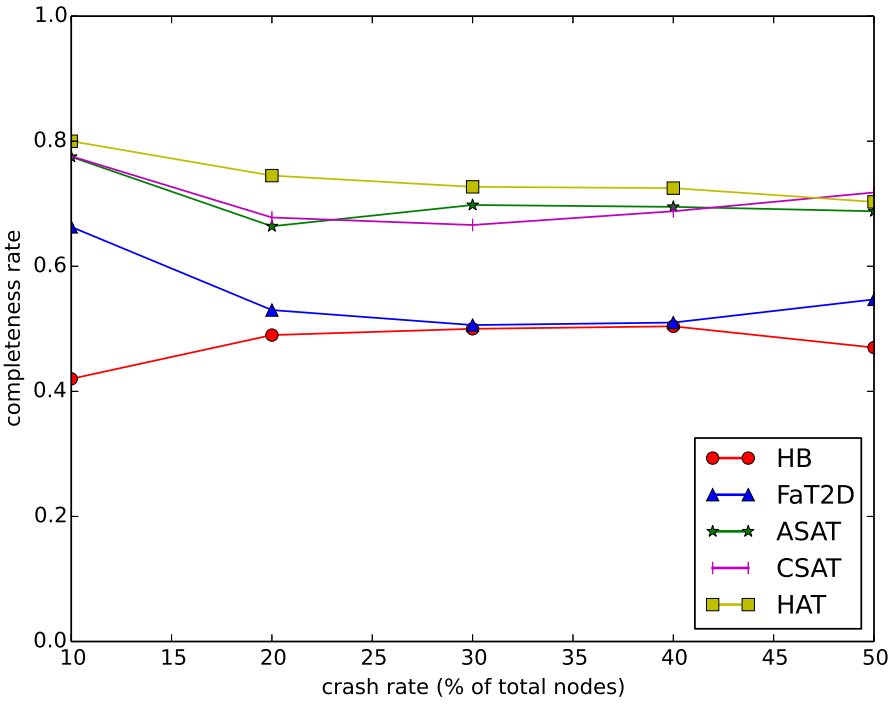

Figure 7: Completeness evaluation VS number of crashes (scenario2).

mittent failures are more frequent in sink neighborhood (because of packets collisions and MAC overload). Since HAT updates timer according to distance from the sink, it allows then to prevent from mistaken suspicions in this region. Subsequently, HAT offers great accuracy rate. However, CSAT gives lowest accuracy rate. As CSAT decreases its timer in order to give prompt detections, it leads to increase false suspicions rate. Subsequently, CSAT has the weakest accuracy rate (61% to 70%). However, timer-free HB shows great results. Obviously, this is due to mechanism used by HB. As mentioned before, HB sends all counters to application layer which decides

for any failure occurrence. Since the process needs large delays to make any notification, it avoids therefore false suspicion to occur.

It is important to notice that accuracy and completeness do not vary very much according to number of crashes for the same technique (figures 7 & 9). This is mainly due to the local detection mechanism. Actually, each AFDEL technique uses local interactions to detect/notify failures. Moreover, timer adaptation strategy depends exclusively on exchanged information in the same neighborhood. For this reason, completeness/accuracy performance is directly related to the dynamic timer pattern and does not depend on the number of crash in the network. However, in figure 9, we notice that the accuracy rate varies from 60% to 80% with variable fluctuations. Some of the observed fluctuations in this interval are due to the impact of intermittent failures on network activity. Actually, the influence of injected intermittent failures depend on routing path length (number of nodes participating in routing data).

Graphs in figure 10 represent detection and recovery delays overlapped in the same histogram. Detection delay is defined by the crash detection period (i.e. first detection by any supervising neighbor). While the recovery period is the time after which all nodes (i.e. neighbors) have noticed the crash (i.e. the crash notification and node suppression from neighbors' list). Their addition defines the total period of failure treatment (detection and notification then suppression and path recovery). Results show that implementing a dynamic mechanism for timer update has slightly increased the detection and recovery period comparing to static timer based FaT2D. Naturally, this is due to additional processing to adapt timer. Furthermore,

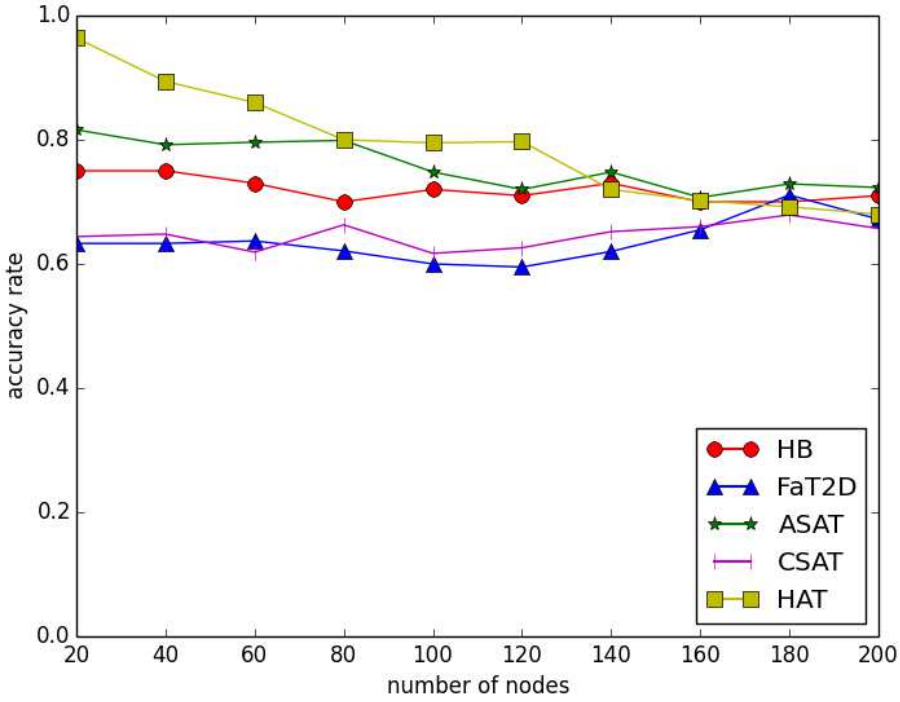

Figure 8: Accuracy evaluation VS number of nodes (scenario1).

notice that ASAT has the longest detection period. Obviously, this is due to large values of dynamic timer in order to avoid mistaken suspicion. Yet the compromise between failure detector properties (accuracy & completeness) and failure detection delay is better insured by the new adaptive techniques, especially CSAT and HAT. However, we notice in figure 10 some fluctuations in detection/recovery period. This variation is due to the nature of the implemented routing algorithm Directed Diffusion. We recall that DD periodically forwards exploratory data in order to reelect routing paths. In case where exploration is triggered just after a failure occurrence, the detection/recovery

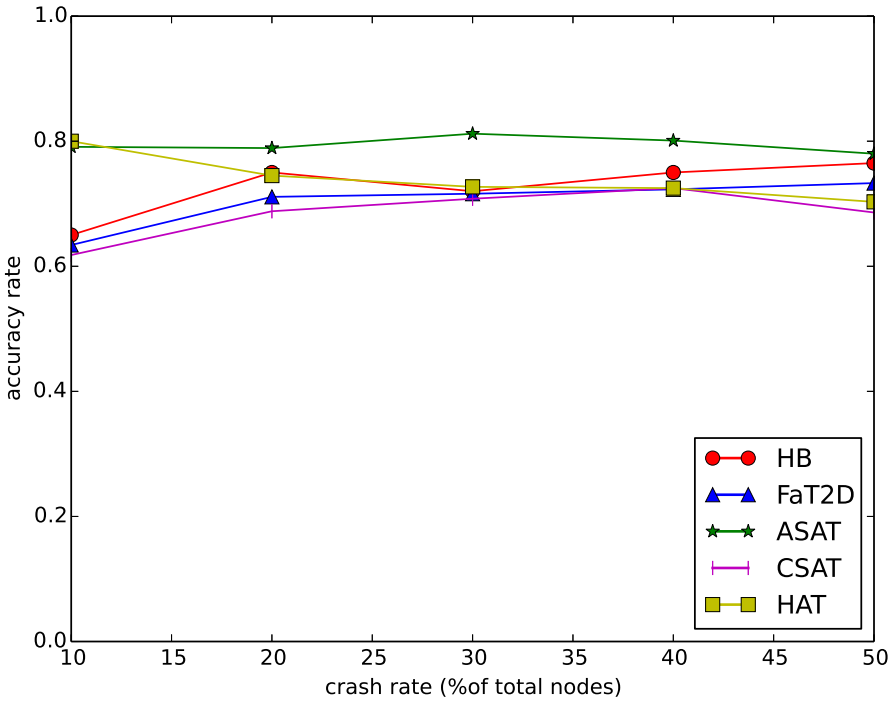

Figure 9: Accuracy evaluation VS number of crashes (scenario2).

may be covered by DD route reelection. In this case, detection/recovery period is faster than the implemented failure management technique (DD launch recovery mechanism before the end of failure detection timer).

This conclusion is satisfied in figure 11 for packet loss rate. For the same reasons discussed above, as some techniques may take a longer detection delay, this leads to a bigger loss rate. CSAT gives the best results for this metrics. We recall that HB shows the longest detection period as it waits for application layer to make decision about any crash notification. Besides, HB does not define any recovery mechanism.

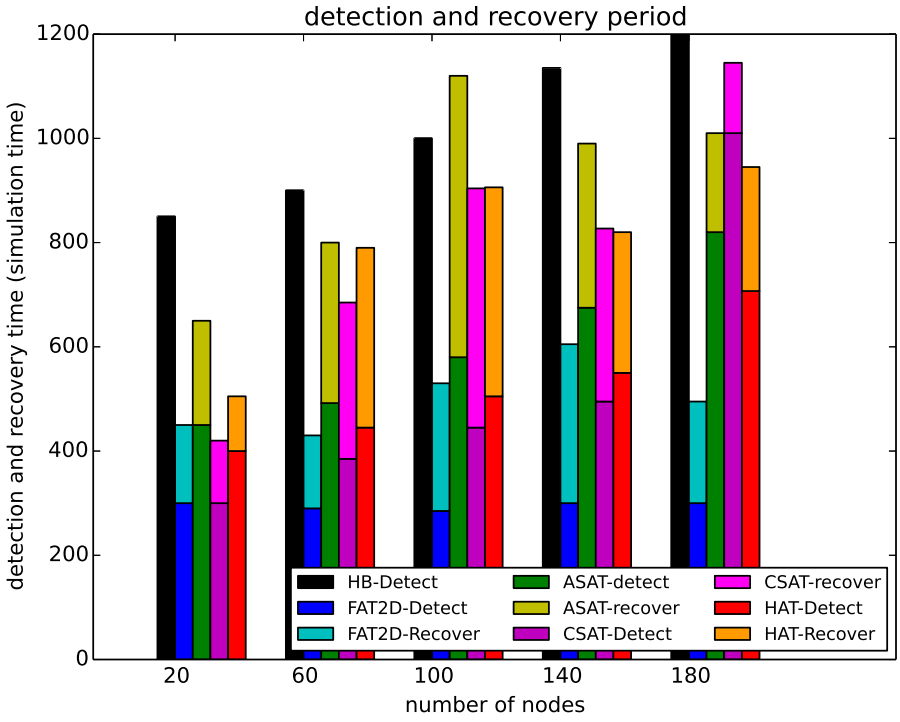

Figure 10: Detection and Recovery time.

The final measurement was performed for the resource management. As energy consumption, memory storage and bandwidth are critical constraints for WSN, it's important to evaluate the impact of implementing a failure detection mechanism on the amount of battery usage and exchanged data overhead. Results in figure 12 show that, even though all three techniques implement an additional mechanism for timer update, the difference in battery consumption is very small comparing to timer-free HB. This result is promising since the exchanged data overhead due to implementing an additional technique for failure detection compensate the overhead that would

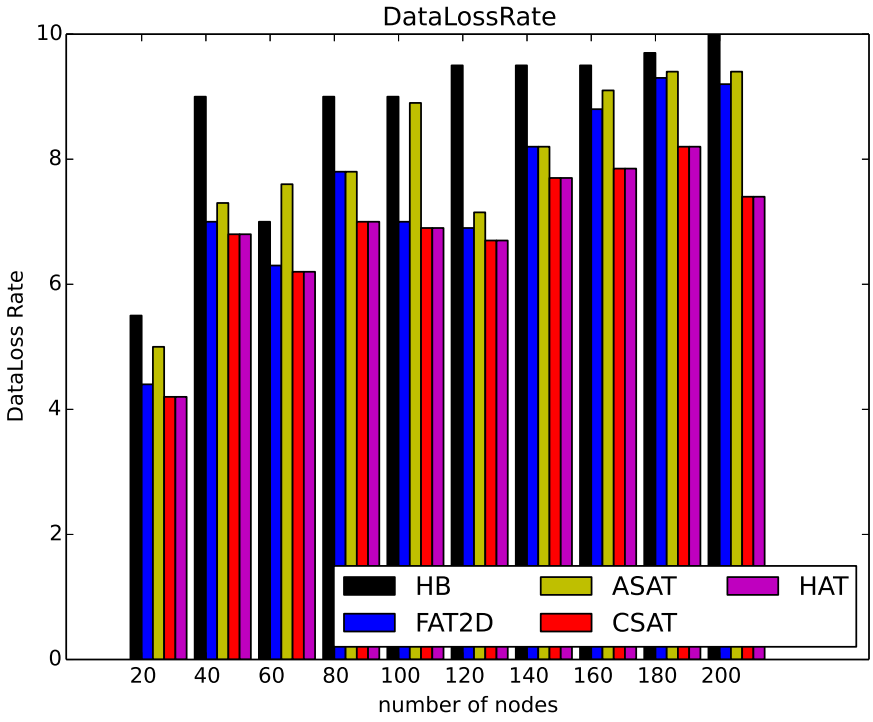

Figure 11: Data Loss Rate.

have been induced because of useless transmissions after node failures as shown in figure 13. Notice that HAT performance is slightly greater than other techniques. Moreover, using local interaction and piggybacking notifications on data messages has considerably reduced the overhead in the network. These features have clearly enhanced failure detection performance comparing to timer-free HB. Actually, HB forwards additional packets to all nodes in the network in order to send information about failure detection (i.e. each node's counter). While all implemented timer-based techniques use data message to send this information. Furthermore, the message is sent

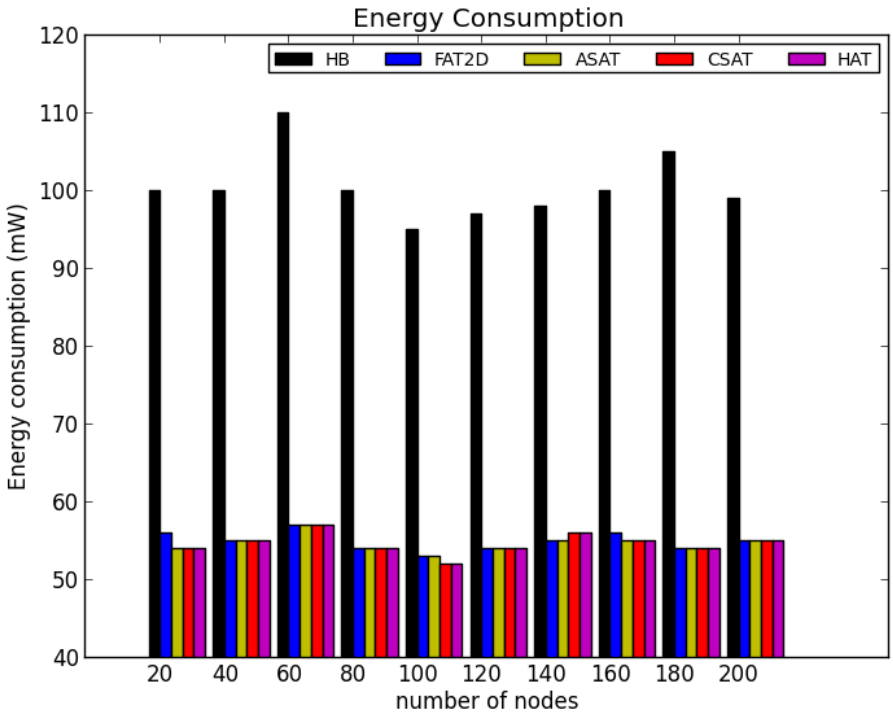

Figure 12: Performance of energy consumption.

only to direct neighbors. It will reach other nodes during data dissemination. Yet, all adaptation timer techniques CSAT, ASAT and HAT do not induce extra energy consumption comparing to static-timer FaT2D. Subsequently, even though AFDEL uses additional processing for timer update, it does not induce extra energy consumption or bandwidth congestion considering the enhancement of detection performance.

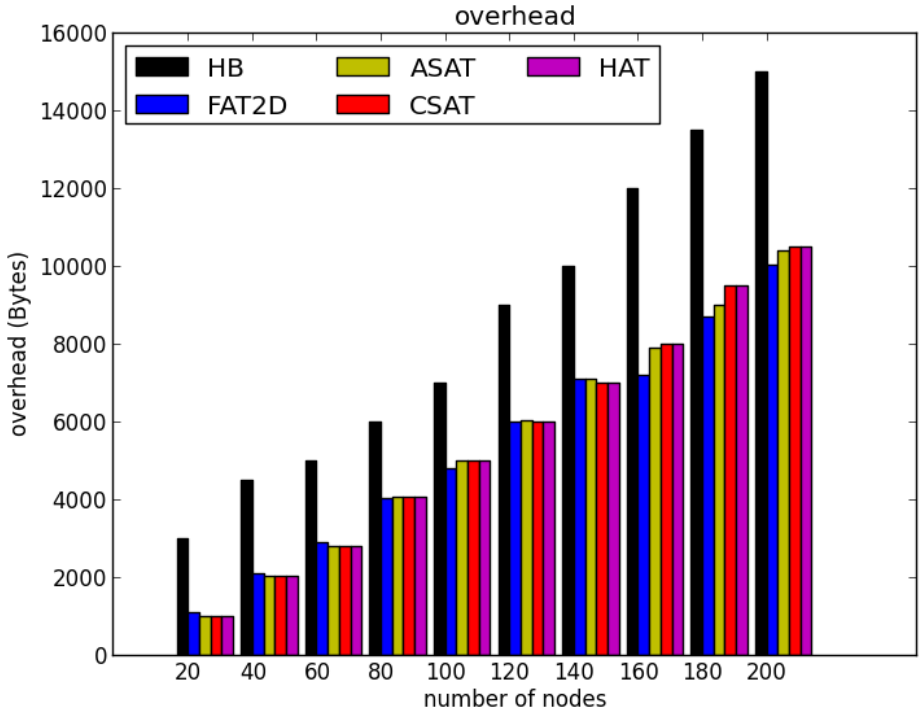

Figure 13: Overhead evaluation.

7.4. Interval of confidence

We have constructed an interval of confidence for all simulation results. To this end, we have used the formula with 95% of probability that true mean values lies in the following range :

$$\{\bar{x} - 1.96\frac{\sigma(X)}{\sqrt{n}}; \bar{x} + 1.96\frac{\sigma(X)}{\sqrt{n}}\} \tag{6}$$

Where:

$\bar{x}$: the mean value of the sequence of simulation results during

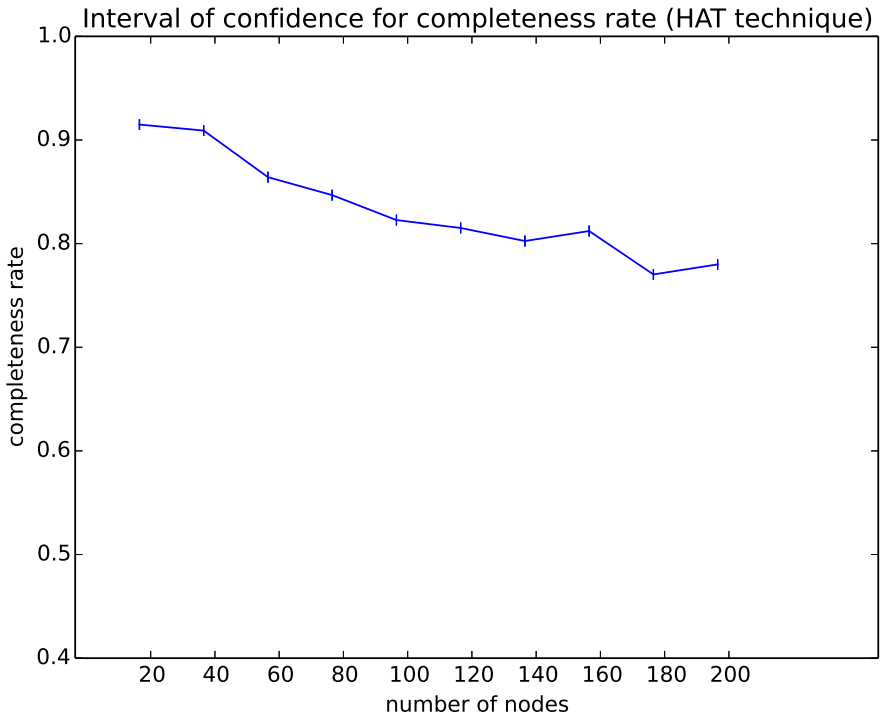

Figure 14: Confidence Interval for completeness rate (HAT technique).

several iterations.

$\sigma(X)$: the standard deviation of the sequence of simulation results during several iterations.

n: lenght of the sequence of simulation results (i.e number of iterations).

As illustration, we show in graph of figure 14 the mean value of completeness rate with their interval of confidence for HAT technique. Notice that the maximum value of error range is [-0.6, +o.6]. These values allow to rely on simulation scenario results and make reliable synthesis.

8. Conclusion

Table 6: Synthesis of performance evaluation.

	FD properties		**Detection & Recovery**		**Resources management**	
	Completeness	Accuracy	Recovery time	Data loss rate	Energy	Overhead
HB(timer-free)	+	+++	+	+	++	+
Fat2D(static-timer)	++	++	+++++	+++++	++++	+++(1)
AFDEL-ASAT	+++	+++++(1)	++	++	++++	+++(2)
AFDEL-CSAT	++++	+	++++	++++	++++	+++(2)
AFDEL-HAT	+++++	++++(3)	+++	+++	++++	++(1)

(1) Dense networks.

(2) Frequent crashes.

(3) Small networks.

In this paper, we present different failure detection solutions designed for distributed systems and study their application in WSN with respect to a set of proposed classification criteria. Then, we define a general failure detection model that can be used in WSN while considering packet loss, environment constraints and resource limits. Our model, called AFDEL, extends eventually strong $\Diamond S$ class to $\Diamond S^{al}$ using adaptive timer and local interactions. Besides, we illustrate how AFDEL model is used, by introducing three different timer adaptation techniques (ASAT, CSAT, HAT), each with distinctive preferences and goals in what relates to accuracy, completeness and recovery delay. We carried out extensive simulations using Mixim/Omnet++ to evaluate the performance of our techniques and compare them against some solutions from the literature. Finally, our evaluation allows to make some recommendations on the AFDEL-based technique to use depending on application requirements (Table 6). If reducing false positive rate is at premium,

then ASAT would be preferable, especially for dense networks. Conversely, if detecting all failures is more important than avoiding mistakes, then CSAT would represent the best choice. Furthermore, HAT helps to enhance the detection time and data delivery, particularly in small-size networks and mainly in case where intermittent failures are more frequent while approaching the sink region. Note that HAT relies only on the distance separating the monitoring node from the sink. However, SAT techniques use in-time network statistics to update timer. This may induce extra processing time and resource overhead. For this reason HAT shows better results compared to SAT with respect to resource overhead.

As an extension to this work, we suggest to investigate the use of this model considering other WSN constraints such as mobility, security and heterogeneity.

References

[1] F. Benhamida, Y. Challal, Fat2d: Fault tolerant directed diffusion for wireless sensor networks, in: Proceedings of the 5th International Conference on Availability, Reliability, and Security. ARES'10, 2010, pp. 112–118.

[2] M. Aguilera, W. Chen, S. Toueg, Heartbeat: A timeout-free failure detector for quiescent reliable communication, in: 11th International Workshop on Distributed Algorithms (WDAG 97), Saarbrucken, Germany, LNCS, vol:1320, Springer-Verlag Berlin, 1997, pp. 126–140.

[3] T. D. Chandra, S. Toueg, Unreliable failure detectors for reliable distributed systems, Journal of the ACM 43 (2) (1996) 225–267.

[4] L. Guo, L. Zhang, Y. Peng, J. Wu, X. Zhang, W. Hou, J. Zhao, Multipath routing in spatial wireless ad hoc networks, Computers Electrical Engineering 38 (3) (2012) 473 – 491, the Design and Analysis of Wireless Systems and Emerging Computing Architectures and Systems.

[5] A. Mouradian, I. Aug-Blum, F. Valois, Rtxp: A localized real-time mac-routing protocol for wireless sensor networks, Computer Networks 67 (0) (2014) 43 – 59.

[6] C. Abreu, M. Ricardo, P. Mendes, Energy-aware routing for biomedical wireless sensor networks, Journal of Network and Computer Applications 40 (0) (2014) 270 – 278.

[7] F. Bonnet, M. Raynal, Anonymous asynchronous systems: the case of failure detectors, in: Proceedings of the 24th international conference on Distributed computing, DISC'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 206–220.

[8] F. Bonnet, M. Raynal, Looking for the weakest failure detector for k-set agreement in message-passing systems: Is Π_k the end of the road?, in: Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 149–164.

[9] A. Mostefaoui, E. Mourgaya, M. Raynal, Asynchronous implementation

of failure detectors, in: Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks, 2003, pp. 351–360.

[10] P. Sens, L. Arantes, M. Bouillaguet, V. Simon, F. Greve, An unreliable failure detector for unknown and mobile networks, in: Proceedings of the 12th International Conference on Principles of Distributed Systems, OPODIS '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 555–559.

[11] F. Greve, P. Sens, L. Arantes, V. Martin, Asynchronous Implementation of Failure Detectors with partial connectivity and unknown participants, Research Report RR-6088, INRIA (2011).

[12] F. Greve, P. Sens, L. Arantes, V. Simon, A failure detector for wireless networks with unknown membership, in: Proceedings of the 17th international conference on Parallel processing - Volume Part II, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 27–38.

[13] N. Hayashibara, M. Takizawa, Design of a notification system for the accrual failure detector, Advanced Information Networking and Applications, International Conference on 1 (2006) 87–94.

[14] S. Rost, H. Balakrishnan, Memento: A health monitoring system for wireless sensor networks, in: 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks. SECON '06., Vol. 2, 2006, pp. 575–584.

[15] J. W. Branch, C. Giannella, B. Szymanski, R. Wolff, H. Kargupta, In-network outlier detection in wireless sensor networks, Knowledge and Information Systems 34 (1) (2013) 23–54.

[16] J. Chen, S. Kher, A. Somani, Distributed fault detection of wireless sensor networks, in: Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks, DIWANS '06, ACM, New York, NY, USA, 2006, pp. 65–72.

[17] R. Ceretta Nunes, I. Jansch-Porto, Qos of timeout-based self-tuned failure detectors: the effects of the communication delay predictor and the safety margin, in: Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks, 2004, pp. 753–761.

[18] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Vol. 8, 2000, pp. 8020–8029.

[19] F. Greve, P. Sens, L. Arantes, V. Simon, Eventually Strong Failure Detector with Unknown Membership, Computer Journal 55 (12) (2012) 1507–1524.

[20] M. Larrea, A. Fernandez, S. Arevalo, Optimal implementation of the weakest failure detector for solving consensus, in: Proceedings of The 19th IEEE Symposium on Reliable Distributed Systems, SRDS '00, Washington, DC, USA, 2000, pp. 52–59.

[21] H.-S. Wang, N. Moayeri, Finite-state markov channel-a useful model for radio communication channels, IEEE Transactions on Vehicular Technology 44 (1) (1995) 163–171.

[22] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in: Proceedings of the 6th annual international conference on Mobile computing and networking, MobiCom '00, ACM, New York, USA, 2000, pp. 56–67.

[23] R. Jain, K. K. Ramakrishnan, D.-M. Chiu, Congestion avoidance in computer networks with a connectionless network layer, in: C. Partridge (Ed.), Innovations in Internetworking, Artech House, Inc., Norwood, MA, USA, 1988, pp. 140–156.

[24] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. K. Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, S. Valentin, Simulating wireless and mobile networks in omnet++ the mixim vision, in: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems, Simutools '08, ICST, Brussels, Belgium, 2008, pp. 71:1–71:8.

[25] M. de Lima, F. Greve, L. Arantes, P. Sens, The time-free approach to byzantine failure detection in dynamic networks, in: IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), 2011, pp. 3–8.

[26] I. Banerjee, P. Chanak, H. Rahaman, T. Samanta, Effective fault detection and routing scheme for wireless sensor networks, Computers Electrical Engineering 40 (2) (2014) 291 – 306.

[27] A. Mahapatro, P. M. Khilar, Detection and diagnosis of node failure

in wireless sensor networks: A multiobjective optimization approach, Swarm and Evolutionary Computation 13 (0) (2013) 74 – 84.

[28] A. Jhumka, M. Bradbury, S. Saginbekov, Efficient fault-tolerant collision-free data aggregation scheduling for wireless sensor networks, Journal of Parallel and Distributed Computing 74 (1) (2014) 1789 – 1801.