ALLONE: A New Adaptive Failure Detector Model for Low-power Lossy Networks

Fatima Zohra Benhamida, Yacine Challal, Mouloud Koudil

▶ **To cite this version:**

Fatima Zohra Benhamida, Yacine Challal, Mouloud Koudil. ALLONE: A New Adaptive Failure Detector Model for Low-power Lossy Networks. IEEE Global Communications Conference, 2013, Atlanta, United States. pp.113-118. hal-00871191

HAL Id: hal-00871191
https://hal.science/hal-00871191

Submitted on 9 Oct 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ALLONE: A New Adaptive Failure Detector Model for Low-power Lossy Networks

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

Yacine Challal
Université de Technologie de Compiègne
HEUDIASYC UMR CNRS 7253
Compiègne, France
ychallal@hds.utc.fr

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

***Abstract*—We consider the problem of failure detection in networks with energy and communication constraints. Most of current implementations of unreliable failure detectors (FD) use mechanisms to notify process failures in fully connected networks with reliable communication links.This assumption is not applicable to lossy networks. Furthermore, such implementations do not consider resource limitations in terms of energy, storage and bandwidth. This paper presents a new failure detection model for Low-power Lossy Networks(LLN). Our approach defines an adaptive timer-based paradigm. Besides, we introduce two techniques based on the proposed general model. We evaluate all contributions using implementation on Omnet++/Mixim framework. Simulation results demonstrate that our FD enhances detection performances for LLN compared to traditional FD.**

Keywords—Failure detector; low-power network; lossy links; adaptive timer

I. INTRODUCTION

Unreliable failure detector (FD) is a fundamental service for distributed systems. Its importance has beenrevealed by Chandra and Toueg[4]who define FD as a per process oracle which periodically provides a list of processes suspected of having crashed. This paper focuses on FD for low-power, lossy networks (LLN)such as MANET, MESH and WSN. This kind of networks is characterized by the following properties: (1) a node (i.e. process) does not necessarily know all the nodes of the network, (2) the network is not fully connected, which means that a message sent by a node might be routed through a set of intermediate nodes until reaching the destination, (3) communication between nodes use a periodic pattern; such as query-driven and time-driven routing protocols, (4) links are prone to failures and may momentarily drop messages during transmission, and (5) nodes are equipped with limited energy batteries communication protocols must consider this limitation to keep the network alive as long as possible.

We recall that conventional FD initially designed for distributed systems do not respond to LLN constraints. As we aim to enhance the failure detection paradigm for LLN, we propose, in this paper, a new FD class, called *ALLONE* (**A**daptive Failure Detector for **L**ow-power **Lo**ssy **Ne**tworks). Indeed, our solution relies on adaptive timers to detect nodecrashes 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 list of failure suspicions is piggybacked into periodic data messages of LLN application.

Furthermore, we carry out simulations using Omnet++/Miximto evaluate the detection and accuracy performance of the proposed solution. We particularly evaluated the induced energy consumption overhead. The proposed solution demonstrates that, in addition to the benefits 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: section II presents an overview and motivation of the proposed solution. Then, we define different models assumed for the implementation in section III. In section IV, we introduce the general algorithm of our new FD model tailored to LLN. We illustrate the use of this FD model by proposing two techniques based on its general algorithm in section V. Section VI presents simulation results, performance evaluation analysis and comparison.This paper ends up with conclusions in section VII.

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: Process P_j must periodically send data messages 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 FailureDetection Timeout (T_j): if T_j runs out before receiving the expected message, $P_i$suspects that P_j has crashed. T_j value is initiated according to application parameters; such as periodic sending interval, 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 T_j, P_i may make some mistakes, when $P_j$is still correctly working but some messages are lost because of intermittent failures (lossy links, congestion…). Moreover, the pattern of packet loss isn't stable during all the network

This work was partially carried out and funded in the framework of the Labex MS2T. It was supported by the French Government, through the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02).

lifespan. Actually, changes may occur depending on obstacles, geographic situation, suppression, deployment or mobility. For this reason, we need to update T_j following the dynamic lossy pattern of P_j. Besides, if the network contains more than two processes, every node's FD must locally detect failures and then notify its suspicion to neighbors. The supported mechanism must consider resources limitations as energy depletion, bandwidth and storage constraints. Actually, failure notification may need additional energy consumption and induce extra traffic in the network.

To circumvent this obstacle, we explore in this paper, the use of adaptive timer based on local interaction between direct neighbors. Moreover, we make use of a stochastic approach to determine the timer **T**taking into consideration loss ratio, links state, transmission delay, etc. It is important to notice that **T** is defined between two consecutive neighbors instead of any couple of nodes. This allows more accurate calculation of its value that relies then on local predictable transmission delay without requiring knowledge about the entire system composition and network topology. Nevertheless, local suspicions will propagate throughout the network. 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.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.

III. SYSTEM MODEL

A. Network model

We consider a network system consisting of a finite set V of n>1 processes, namely, V= $\{P_1,\ldots,P_n\}$. 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 $(P_i,P_j) \in E$*iff*they are within their wireless transmission range. In this case, P_i and P_j are considered neighbors.

B. Failure model

We assume that processes may crash. Moreover, links between two neighbors are considered lossy: they may drop messages during transmission. This is commonly due to transmission link failures; which are intermittent by nature. Subsequently, if a process 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. In order to tolerate intermittent failures for LLN, we use burst packet loss model as suggested by Elliott[6] and Gilbert [7]. The number of successive packet losses can be represented by a geometric distribution of variable BL (Burst Length), which defines the time sojourn in Bad state (i.e. successive packet losses). Then, we define hereafter a burst loss limit (BLL) which must be tolerated by the FD.

$$BLL = E(BL) + V(BL)$$

- *E(BL)*=$1/\lambda$: Mean Burst Length
- $\sqrt{V(BL)} = \sqrt{1-\lambda}/\lambda$: 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.

C. Data gathering model

We consider two types of communication models based on periodic packet transmission: Query-driven and Time-driven. First, the Query-driven model, also known as a Query-Response routing mechanism is based on network interrogation; where a process initiates a query to all network nodes (resp. neighbors). The destinationprocess responses within an interval of time predefined in the protocol parameter. DSR[3], AODV [12]and Directed Diffusion[8]are Query-based protocols. On the other hand, Time-driven model uses a periodic scheme for data dissemination. The routing protocol defines a time slot or an interval for each transmission round. Many proactive protocols (DSDV[13], OSR[10], LEACH[9]) use Time-driven model for their route construction and maintenance.

IV. ALLONE:FAILURE DETECTION ALGORITHM

The algorithm proceeds by rounds following the periodic pattern of communication model. At each round, a node sendsa message to its neighbor(s) until it possibly crashes. The basic principle of our model is to piggyback failure suspicion on existing data/routing messages, and dispatch this information hop by hop. Hence, we add up $Suspected_i$listto each message.$Suspected_i$ is the list of process P_i containing its suspected neighbors.`ALLONE` module for process P_i is described in Fig. 1. It is composed of two main primitives (send&receive) and one task: for each message MSG, `ALLONE` checks if MSG belongs to primitive *Send*(i.e. data message generated by P_i to be sent into the network). If so, P_i includes $Suspected_i$ set in MSG (line 4).After piggybacking the information, P_i sends the new message MSG_s to corresponding nodes in $neighbors_i$ (line 5).Then, P_i initializes a new timer for every correct known neighbor; say T_j for node P_j awaiting for the next message (lines 6&7).If any T_j runs out, P_i suspects P_j of being faulty in Task T1.Suspicion information is then inserted in $Suspected_i$ (line 20).This suspicion is included in the next data message when there is another MSG to send. The timer schedule is dynamic and depends on stochastic changes in the network. We explain how to adapt timers in next section. On the other hand, if MSG is received by P_i from neighbor P_j (line 8), `ALLONE` executes the second primitive (i.e. receive)to update suspected set according to the one piggybacked on the received message. It also tries to discover previous mistaken suspicions in order to remove the corresponding nodes from suspected list. First, P_i delivers the data messageto its own

application layer. Then, it retrieves information, namely, *suspected*$_j$ set with the source node P_j (lines 11&12). Since P_i has successfully received message from P_j, it stops timer T_j (line 13). P_i checks then if P_j was recently suspected in order to delete it from *Suspected*$_i$ set (lines 14&15).After that, P_i updates its own list using received information about suspicions (and mistakes) in P_j's message. Thus, P_i includes every suspected process notified by P_j. 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 16-18).The mistake is detected if *Suspected*$_j$ doesn't contain P_m while the latter is still in *Suspected*$_i$ list.

```
1    For each message MSG do
2    if MSG is to send then
3    /*Add suspicion information then send MSG*/
4      MSGs ← Piggyback(MSG, <suspectedi, Pi>)
5      Send new message MSGs
6    for allPjin neighborsi\suspectedi do
7      ScheduleTimerTj for Pj

8    else     /*MSG is to be received */
9     Retrieve    the    following    informationfrom
      received message MSG:
10 -MSGr:Data  messageto  be  deliveredtoapplication
      layer
11 -suspectedj:  List  of  suspicionspiggybacked  in
      MSG
12 -Pj:Source of suspectedj list
13 Stop Timer Tj for Pj
14   if Pjin suspectedithen
15   Delete Pj from suspectedi
16   Updatesuspectedifrom receivedsuspectedj:
17      -Addsuspectedprocess by Pjto suspectedi
18   -Remove mistaken suspicion notified by Pj

19   Task T1:
20     if TimeOut(Tj)thenAdd Pj into suspectedi
```

Fig. 1. ALLONEmodel algorithm.

To better understand the general model, let's explain one scenario with all possible situations in Fig. 2. Initially, each process contains some information about other processes in the network (*Suspected*list). This scenario aims to get the same and most recent information in all *Suspected* lists:

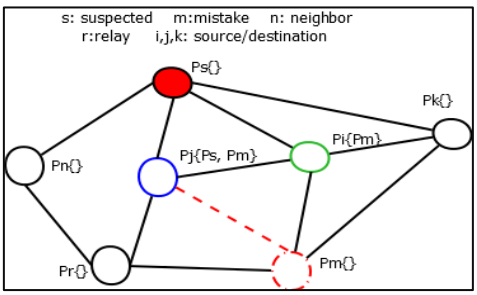

Fig. 2. Example of failure detection scenario

1. P_j detects P_s crash, and notifies to P_i; P_i updates Su*spected*$_i$={P_s, P_m}

2. P_i sends to P_k a message with its *Suspected*$_i$ ={P_m, P_s}; then P_k updates its *Suspected*$_k$={P_m, P_s}

3. P_j discovers mistaken suspicion P_m, deletes P_m from *Suspected*$_j$then sends a message to P_i. P_i updates *Suspected*$_i$={P_s} then P_k updates *Suspected*$_k$ from received *Suspected*$_i$

4. P_n recieves information of suspicions and mistakes from P_j through relay node P_r, then updates S*uspected*$_n$

V. TIMER ADAPTATION TECHNIQUES

In what follows, we explain how to schedule timer for failure detection in ALLONE (line 7 in Fig. 1). We recall that the proposed timer T should be dynamically updated while taking into consideration transmission failures, communication protocol behavior and network dynamics. To this end, we propose two timer adaptation techniques based on this mechanism. The first one aims to notify all crashes with lower false positive rate. Thus, we call it *Accuracy-aware Stochastic Adaptive Timer* technique, A-SAT. However, the second techniqueseeks to make in-time monitoring by notifying any process crash as soon as possible. Obviously, it 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, C-SAT.

```
1. T:=T0/*set adaptive timerto initial value*/
2. if (WDR>= TWR) then
3. Increase(T)
4. elseif (1-WDR<τr)
5. Decrease(T)
```

Fig. 3. Timer adaptation algorithm.

Fig. 3 explainsthe algorithm to set and update timer for both techniques. Later on, we add separately the difference in A-SAT and C-SAT mechanisms. In the beginning, since there is no former communication to make statistics from, the first T value is set according to hypothetical values of transmission interval and latency (line 1).Then, T is updated using the effective loss rate regarding received data messagesduring the application run. Namely, it is increased if the Wrong Detection Rate*WDR*has exceeded the Tolerated Wrong Detection rate*TWR*(lines2&3), which is a threshold that we define to control the accuracy of the FD. If this threshold is reached, ALLONE must enlarge the waiting period before notifying failures. However, when Tallows to satisfy the desired *TWR*, we try to enhance crash notification by decreasing backits value if it doesn't reach yet the predefined reliability threshold(τ_r) (lines4&5).τ_r is introduced by the application in order to define the desired reliability of the failure detector allowing faster detection.Obviously, the algorithm aims to offer the best combination of prompt detection with fewer mistakes as illustrated in Fig. 4. The performance depends on the choice of the predefined parameters (τ_r, *TWR*) in a hand, and both Decrease(T), Increase(T) functions on other hand.Our approach is inspired by control-theoretic adaptation similar to those widely used in Internet, such as AIMD or MIADbest known for TCP congestion avoidance[5].Hence, we explain hereafter the difference between A-SAT and C-SAT where time increase/decrease is differently implemented according to application requirements.

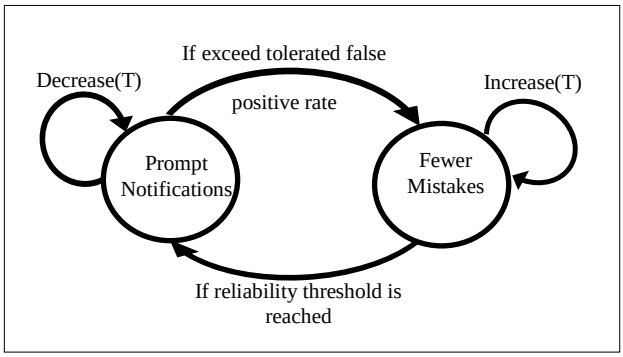

Fig. 4. Timer update process.

A. Accuracy aware-SAT

By A-SAT technique, we aim to notify crashes with good accuracy. This approach aims to reduce the mistaken suspicions rate. For this reason, we use *Multiplicative Increase Additive Decrease* algorithm (MIAD) [5]. 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:

$$T(t+1) = \begin{cases} \alpha_i T(t) & if\ Increase(T), \\ \beta_d + T(t) & if\ Decrease(T) \end{cases}$$

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 exceeds the tolerated threshold (WDR>TWR). According to algorithm inFig. 3, we increase the timer. Since A-SAT aims to make fewer mistakes, we multiply the timer by α (α>1) in order to enlarge its value. Moreover, the parameterαdepends 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, indecrease(T) function, the timer is reduced using parameter β. As for β value, it is deduced from the local notification delays during crash detections[1]. Notice that decrease(T)function is triggered once the FD is under the required reliability threshold. For simplicity reasons, we define the reliability as the rate of correct suspicions; namely 1-*WDR*.By aggressively increasing the timer value upon false suspicions control, A-SAT 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 in order to quickly overcome unreliability.

B. Completeness aware-SAT

We propose Completeness-aware Stochastic Adaptive technique(C-SAT), for prompt crash detection. Thus, completeness takes the advantageon accuracy. The purpose is to quickly notify any crash failure. For this reason, we use *Additive Increase Multiplicative Decrease* technique[5]. Inversely to MIAD approach, AIMDcombines linear growth of timer value with exponential reduce as given in the formula hereafter:

$$T_a(t+1) = \begin{cases} \beta_i + T_a(t) & if\ Increase(T), \\ \alpha_d T_a(t) & if\ Decrease(T) \end{cases}$$

As for A-SAT with AIMD, the timer increase, respectively decrease, function depends on reliability in crash notification, respectively the false positive rate. However, C-SAT aggressively diminishes the timer value to quickly detect any crash failure. For this reason, we multiply `T`by parameter αwhich is defined according to the effective false positive rate. The objective behind this choice is that C-SAT decreases the timer to promptly detect crashes regarding the actual false positive rate. Meanwhile, to avoid mistaken suspicions, C-SAT increases`T`when it reaches a required reliability. For this, we add up to `T` parameter β, 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).

VI. PERFORMANCE EVALUATION

In this section, we study and evaluate the behavior ofour dynamic FD,`ALLONE`and the proposed adaptation timer techniques. We start by comparing the general mechanism of our timer-based model with a timer-free FD. Then, we measure the importance of updating timer and the impact of each proposed dynamic technique; namely C-SAT and A-SAT. To this end, we have chosen HeartBeat[1] as a timer-free FD. The failure detector module of HBat a process *p*outputs a vector of counters, one for each neighbor *q*of*p*. Each process periodically sendsan 'I-am-alive message' (a "heartbeat") and every process receiving a heartbeat increasesthe correspondingcounter. Thus, if neighbor *q*does not crash, its counter increases with no bound. If *q*crashes, its counter eventually stops increasing. For failure detection, HBcounts the total number ofreceived heartbeatsfrom each process, and considers that neighbor *q* has crashed when the counter stops increasing.HB outputs these "raw" counters to applicationswithoutanyfurther processing or interpretation.

TABLE I. SIMULATION MODEL.

Scenario	Nb-Nodes	Nb_Faults (injected)	Failure Detectors	Simulation Period/ iterations
Scenario 1	[20-200]	10% of total nodes	• HB timer-free • Static-Timer • ALLONE-ASAT • ALLONE-CSAT	5 h /50
Scenario 2	100	[10%-50%] of total nodes.	• HB timer-free • Static-Timer • ALLONE-ASAT • ALLONE-CSAT	5 h /50

We carried out the simulations using MIXIM; an Omnet++ modeling framework[11]. The simulation model is summarized in TABLE I. In order to illustrate the outcome of using adaptive failure detection; we considered several simulation tests using two main scenarios. In scenario1, we vary the number of total nodesrandomly deployed (20 to 200 nodes),

[1] Notification delay is the period from crash occurrence until its detection.

while in scenario2; we change the number of crashes for the same network(10% to 50% of total nodes).As a periodic data communication model, we consider Directed Diffusion routing protocol DD[8]. In our simulation tests, we compare DD-HB(DD augmented with timer-free FD), FaT2D[2] (DD augmented with static timer FD) with two variants of ALLONE depending on the used timer adaptation technique: ALLONE-ASAT andALLONE-CSAT.

A. *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 allows classifying the proposed techniques according to their performance regarding the false positive rate (accuracy) and the crash detection rate (completeness).

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 LLN are resource-limited, we deem necessary to evaluate and compare both *Energy Consumption* and *Overhead*(amount of message in the network) for each implemented technique.

B. *Analysis*

We discuss hereafter some of the simulation results. Fig. 5shows completeness performance. Both proposed ALLONEtechniques give better results than static timer-based FaT2D. Moreover, C-SAT offers the best rates (70% to 95%). Obviously, this results from the timer adaptation in C-SAT which aims to prompt detection of all faults. 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 to application, the detection mechanism is very slow. The latter is clearly explained in Fig. 6 which shows the detectiontime. As mentioned before, HB waits during a long period to decide if any node has crashed. For this reason, it needs a longer detection period than timer-based FaT2D,C-SAT and A-SAT. Yet, if we consider bothdetection and recovery periods for these three newtechniques, C-SAT shows better performances. Notice that the addition of detection and recovery periods defines the total period of failure treatment (detection and notification then suppression and path recovery)[2].

On another hand, A-SAT gives the greatest accuracy rate among the three techniques (Fig. 7). Clearly, this results from the formula of the timer update in A-SAT which aims to minimize mistaken suspicions. Particularly, HB shows also good results since it takes longer time to notify any process as faulty. Thus, it avoids mistaken notifications.

[2] HB does not define any recovery mechanism.

The final measurement was performed for the resource management. As energy consumption, memory storage and bandwidth are critical constraints for LLN, it's important to evaluate the impact of implementing a failure detection mechanism on the amount of battery usage and overhead. Results in Fig. 8 show that, even though C-SAT and A-SAT implement an additional mechanism for timer update, the difference in battery consumption is very small comparing to static timer. This result is promising since the overhead due to implementing an additional technique for failure detection compensate the overhead that would have been inducedbecause of useless transmissions after node failures as shown in Fig. 9.

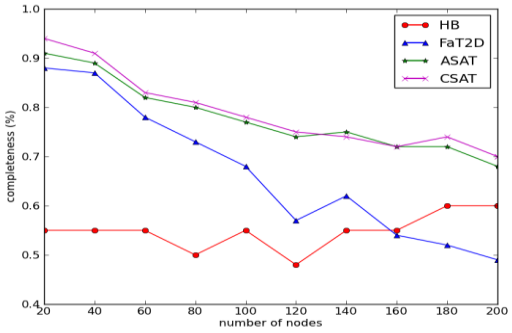

Fig. 5. Impact of failure detection timers on completeness.

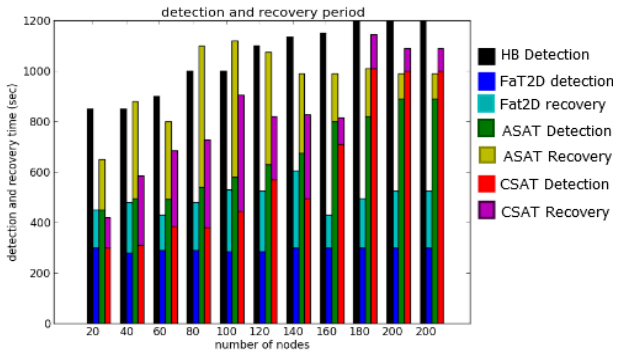

Fig. 6. Impact of failure detection timers on detection and recovery period.

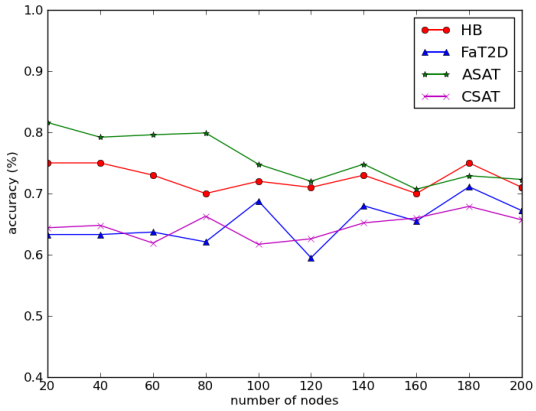

Fig. 7. Impact of failure detection timers on accuracy

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 only to direct neighbors. It will reach other nodes during data dissemination. Yet, both adaptation timer C-SAT and A-SAT do not induce extra energy consumption comparing to static-timer FaT2D.

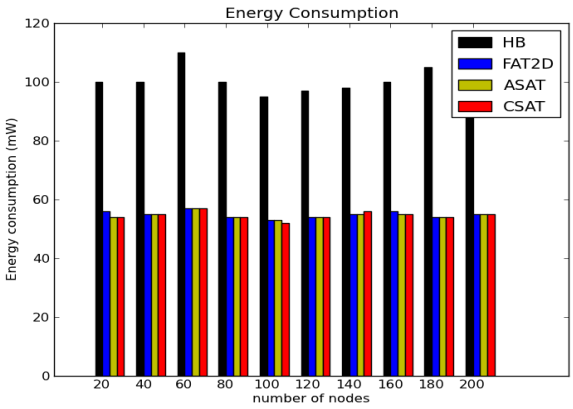

Fig. 8. Impact of failure detection timers on energy consumption.

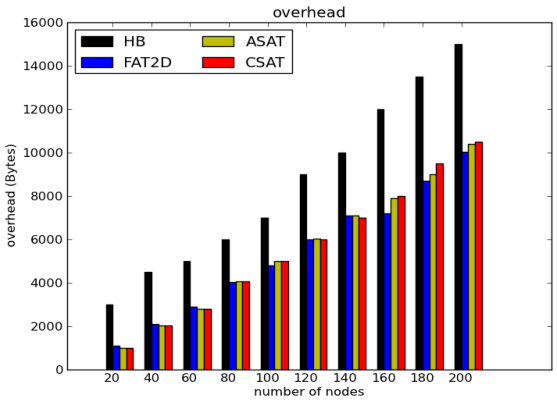

Fig. 9. Impact of failure detection timers on overhead.

VII. CONCLUSION

In this paper, we have proposed a new class for failure detection. ALLONE has the following characteristics: (1) it represents a general failure detection model defined for low-power lossy networks taking into consideration their resource constraints(energy, memory, bandwidth); (2) the message exchange pattern is based on local exchanged information among neighbors and not on global exchanges among nodes in the system; (3) it tolerates intermittent failures and adapt crash detection procedure using an interactive dynamic timer.

In addition, we illustrated the timer adaptation mechanism by introducing two techniques A-SAT and C-SAT. This allows to tune the operation of ALLONE depending on application requirements. If reducing false positive rate is at premium, then A-SAT would be preferable, especially for dense networks. Conversely, if detecting all failures is more important than avoiding mistakes, then C-SAT would be the best choice.Yet, both techniques help to enhance the detection time and data delivery. Furthermore, simulation resultsdemonstrate that piggybacking failure detection information into data messages and using adaptive timers allows saving energy and bandwidth.

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] J. Broch, D. B. Johnson, and D. A. Maltz, *The dynamic source routing protocol for mobile ad hoc networks*, 1998

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

[5] D.M. Chiu and J. Raj. *Analysis of increase and decrease algorithms for congestion avoidance in computer networks*. Computer Networks and ISDN systems. Volume 17, Issue 1, 10 June 1989, Pages 1-14

[6] E. O. Elliott. *Estimates of error rates for codes on burst-noise channels* The Bell System Technical Journal, volume 42, pages 1977–1997, Sept 1963.

[7] E. N. Gilbert. *Capacity of a burst-noise channel*. The Bell System Technical Journal, volume 39, pages 1253–1265, Sept 1960.

[8] 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.

[9] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "*Energy-efficient communication protocol for wireless microsensor networks*," Proceedings of the 33rd Annual Hawaii International Conference, volume 2, 4-7 Jan 2000.

[10] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti,A. Qayyum andL. Viennot, "*Optimized link state routing protocol for ad hoc networks*," IEEE INMIC 2001. In Proceedingsof Technology for the 21st Century. IEEE International , pages 62-68, 2001.

[11] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. Klein Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and 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 & workshops (Simutools '08). ICST, Brussels, Belgium, 2008.

[12] C.E. Perkins and E.M. Royer, *Ad-hoc on-demand distance vector routing*,in Proceedings ofMobile Computing Systems and Applications, WMCSA '99, pages 90-100, 25-26 Feb 1999.

[13] C.E.Perkins andP. Bhagwat: *HighlyDynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers*, In proceedings of the of the conference on Communications architectures, protocols and applications (SIGCOMM '94). pages 234-244, London England UK, October 1994.