Revisiting Gossip-style Failure Detection in Wireless Sensor Network

Christophe Pitrey, Françoise Sailhan

Revisiting Gossip-style Failure Detection in Wireless Sensor Network

Christophe Pitrey, Francoise Sailhan

Cedric Laboratory, CNAM, 292 rue Saint Martin, 75003, Paris, France

{firstName.name}@cnam.fr

This work has been supported by the French national agency (ANR) under contract ANRBLAN- SIMI10-LS-100618-6-01, as part of the Murphy project.

Abstract. Wireless sensor networks are often operating over inaccessible and inhospitable environment so as to monitor phenomena. Providing dependable monitoring is henceforth especially challenging. As a first step upon that goal, we propose a gossip-style failure detector that provides hints about failures. More precisely, we introduce few gossiping policies involving a random selection of the gossiper as well as a link-aware and degree-aware selection. We also provide a preliminary evaluation of our failure detector, which involves an analysis based on the percolation theory and empirical benchmarks.

Keywords: gossip, failure detection, WSN

1 Introduction

Wireless Sensor Network (WSN) constitutes a flexible solution for monitoring a physical/environmental phenomenon and transmitting the related data to a remote end-user. Nevertheless, several projects [1, 2] reported the difficulty to provide dependable monitoring, especially under harsh operating conditions. This is due to a combination of factors including the versatility of the wireless links and the very resource-contained nature of sensor. This implies providing a careful memory management, a lightweight implementation while taking into account the failure-prone nature of wireless communication. But, all these requirements are not always met. We herein concentrate on failure detection as a building block for providing dependable monitoring. Informally, a failure detector can be viewed as a distributed oracle that provides hints about the processes (and subsequently the sensors) that fail. A failure detector is commonly characterized [3] by two properties: the *completeness* which is defined as the ability of suspecting every failing process and the *accuracy* which characterizes the ability of not suspecting correct processes. In practice, a failure detector operates following two patterns of interaction: query-reply or heartbeat. In a nutshell, with a query-reply interaction, a process p monitors another process q by querying it. If no answer is provided, p assumes that q failed. With *heartbeat-style* communication, p continually informs q about its aliveness, by sending a so-called

heartbeat message. In absence of heartbeat, q suspects p. In an asynchronous WSN, it is acceptable to report that a process fails whereas it is too slow or unreachable for a reasonably sufficient period of time. For practical reason, we hereafter assume a reasonable partial synchrony. The issue with heartbeats is the high transmission overhead involved by the continuous all-to-all communication. To tackle this issue, gossip-style failure detection has been introduced [5]; the idea lies in continuously heart-beating a very few (logical) nodes. These nodes are typically picked up randomly among the overall set of nodes. As a result gossip is characterized by its probabilistic nature.
We herein revisit gossip-style failure detection, focusing our attention on the propagation of the gossip. This is motivated by the fact that so far, most of the failure detectors have been customized to operate over large-scale distributed system, which implies that some underlying and low-level protocols are responsible for establishing a reliable propagation so that any process p can reach another process q: they henceforth require the interactions to be made among the complete set of processes, which typically involves flooding/broadcasting facilities. Mimicking the behaviour of some failure detectors that have designed for large-scale distributed system would be inefficient in WSN.
This motivates our work on a gossip-style failure detector that does not assume any such underlying communication. We propose a gossip-style failure detector wherein essentially only a subset of the neighbouring sensors gossip (i.e., forward the heartbeat). Thus, this failure detector does not unnecessarily flood the WSN by requiring that each node forwards the heartbeat again and again so as to ensure that each node receives the heartbeat. We further propose several gossiping policies. The simplest chooses uniformly and randomly of the gossipers (i.e., of the intermediate that forward the gossip). A priori, if the number of gossipers is well adjusted, then this leads to high redundancy, reliability and timely-spread to the expense of a fairly high bandwidth usage. Keeping in mind the faulty nature of the wireless links, we propose two additional policies; both require nodes to be more informed. Intuitively, this lies in selecting primarily the nodes characterised by the highest degree and resp. The highest link quality: the gossipers are randomly selected in proportion to their degrees (for the former) and link quality (for the latter). In the following, we propose a preliminary, theoretical and empirical evaluation of the proposed policies.
The rest of the paper is organized as follows. We survey the related work (Section 2) and introduce our failure detector (Section 3) and evaluate its performance (Section 4). Finally, we conclude with a summary of the contribution (Section 5).

2 Related Work

The pioneering work of Backer and al. [19] on gossip protocols paved the way for their usage in various applications including failure detection [5] or the database replication [18]. R. Van Renesse and al. were the first to propose a gossip-style failure detector [5]. Briefly sketched, each node gossips to a single node that has

been selected in a random manner. This basic gossip-strategy has been further customised so as to consider the physical topology that underlays a large-scale wired network (e.g., the Internet). More precisely, the idea lies in reducing cross-subnet and cross-domain gossiping: within a subnet, the gossip runs the basic protocol discussed above. However, (on average) only one member per subnet gossips to another subnet within the same domain. Similarly, (on average) only one member per domain gossips to another domain. For this purpose, the probability of gossiping over a subnet and over a domain is tuned. This permits to reduce the flow of gossips that goes through the Internet while concentrating this flow on the lower level of the Internet.

Similarly, when considering a WSN or a MANET (Mobile Ad hoc NETwork), the underlying topology should be taken into account. As pointed out in [20, 14] in the context of MANETs, the latency related to the failure detection is proportional to the network diameter and depends of the network topology (and more especially of the node density). Meanwhile, the mobility also ameliorates the performance of the failure detection [14, 16]. Leveraging this work on failure detection in MANETs, we herein also propose a gossip-style failure detector concerned by the underlying topology. More precisely, we take into account the noisy nature of a wireless communication and the impact that has the node density. This lead us to further introduce few self-adaptive gossiping strategies. Meanwhile, missing from previous studies and only briefly/partially sketch here is also a topology-aware and analytical evaluation of the failure detection relying on a systematic model as offered by the percolation theory.

3 Gossip-Style Failure Detection

We propose a gossip-style failure detector (see Algorithm 1), which, roughly speaking, attempts to figure out both which sensor is alive and which process is running on this alive sensor. For this purpose, each sensor (said) i maintains three data structures (called $Alive_i$, P_i and $Suspect_I$). The two formers are used to state the aliveness of (i) the sensors as well as (ii) the related processes that are running on those sensors ; the later is used internally by i to define which sensor is suspected. These three structures are updated as follows.
Each sensor (said) i periodically sends a gossip. Let T_{gossip} be this period. This gossip primarily attempts to establish the aliveness of all the sensors (as established in $Alive_i$). It includes their related heartbeats. Any of those heartbeats is materialised by a counter. Naturally, it contains the heartbeat of the sensor i itself; this counter is incremented by i at each emission. The gossip also includes the list of processes that are actually running on i. This list is provided by a watchdog. It corresponds to a vector of bits, denoted[1] $P_i(1),\ldots,P_i(a)$, stating the processes that are actually running on i (i.e., if $P_i\ (j) = 1$, then $P_i\ (j)$ is alive). For the sake of clarity, we hereafter purposely omit to mention this list of processes. Upon reception of a gossip, a sensor merges it with its own gossip list

[1] In practice, we implemented a watchdog that provides a detailed state of each process.

(see algorithm 1). This merge consists in keeping the highest heartbeat counter for each sensor. As a result, each sensor maintains a gossip list constituted of the highest received heartbeat counters. In order to detect a failure, each sensor also maintains the last time each counter has been increased, and, if a counter has not been increased for a duration exceeding T_{fail}, then this sensor is suspected.

Algorithm 1 Failure Detection

1: $Alive_i = [0, \cdots, 0]$ {sensors aliveness}
2: $Suspect_i = [0, \cdots, 0]$ {suspected sensors}
3: **Task emitGossip()**
4: **for** *each* T_{gossip} **do**
5: $Alive_i(i) = Alive_i(i) + 1$ {increment its own heartbeat}
6: *select* $N_{ki} \subseteq N_i$ {select some 1-hop neighbours}
7: *send* $N_{ki} : gossip(i, alive_i)$ {send to these neighbours}
8: **end for**
9: **EndTask**
10: **Task receivedGossip (Nk,** $alive_{N_k}$**)** {merge the local list and the receive one}
11: **for** each $C_x \in alive_i$ and $alive_{N_k}$ **do**
12: **if** $alive_i(j) < alive_{N_k}(j)$ **then**
13: $alive_i(j) = alive_{N_k}(j)$ {retain the smallest counter}
14: $suspect_i(j) = 0$ {update the suspect list}
15: $arm.Timer(j)$ {arm the time out to detect failure}
16: **end if**
17: **end for**
18: **EndTask**
19: **Task suspectFailure**
20: $UponTime.out(j)$
21: $suspect_i(j) = 1$
22: **EndTask**

One fundamental aspect of the gossip is the selection of the sensor(s) towards which a gossip is sent. Recall that a WSN is constituted of sensors that are spatially distributed, this selection should be performed among the 1-hop neighbours that in turn gossip, again and again. Thus, each sensor i selects a subset of sensors among its 1-hop neighbours. In the following, we present several selection/propagation strategies, starting with the simplest form of gossiping with involves a blind flooding. Let N_i be the overall set of 1-hop neighbours and S_i be the selected subset.

Blind Gossip – With a simple flooding, each sensor i periodically sends a gossip to all its 1-hop neighbours. Thus, S_i=N_i. In practice, this is easily done by broadcasting the gossip. As aforementioned, when a node receives such a gossip, it merges the received gossip list with its own gossip list. If each sensor applies such a strategy, all the sensors receive the gossip.

Uniform gossip – Each sensor i selects in a uniform and purely random manner a subset of sensors out of its 1-hops neighbours. This implies that $S_i \subseteq N_i$.

Weighted gossip – Each sensor selects randomly a subset of 1-hops neighbors in proportion to their weights. We herein to take into account two criteria for the weighting. The former involves the link quality and the later the connectedness of the sensor. More precisely, the link quality corresponds to the signal strength, which is measured and averaged over m measurements. The connectedness is expressed as the node degree. These two criteria necessitate the sensor to get more informed about its neighbours. Such information can be provided through a dedicated message broadcasted over 1 hop or piggybacked into the gossip message. In both case, the information provided by a node (said) i includes *the state of the links* that i shares with each of its 1-hops neighbours N_{ki}. Two parameters characterise such a link: the signal strength that i observes (i.e., the inbound signal strength) and the directionality of the link (i.e., bidirectional *versus* unidirectional) of the link. As a result, any node is able to determine:

- the signal strength of any link with a neighbours N_{ki},
- the degree of each of neighbour N_{ki}, expressed as the number of outbound links of N_{ki}.

Overall, the sensor i can henceforth provide a weighted gossiping. In the following, let analyze the above described gossip.

Analytical study – Generally speaking, a gossip process can be perceived as an epidemic dissemination wherein an individual (herein a sensor) can only infect neighboring individuals. An epidemic is characterized by the fact that each individual is not always susceptible of getting infected. Let $p \in [0, 1]$ denote such a probability. The problem of characterizing such an epidemic dissemination can be addressed relying on the percolation theory [9], as originally established by Haas [7]. More formally, let consider a network graph denoted $G(V, E)$, wherein each edge has a bounded degree following a given distribution denoted λ. The gossiping can be perceived as an epidemic starting from any arbitrary sensor v. One central issue lies in defining what is the percolation probability $\theta(p)$, i.e., roughly speaking, what is the probability that there is a giant component (intuitively, there is a giant dissemination taking place over G). Supposing an infinite graph G, and letting $C(v)$ denote a connected component containing v and $|C(v)|$ the number of sensors of $C(v)$, the percolation probability can be defined as:

$$\theta(p) = p_p\left(|C(v) = \infty|\right)$$

It is known that $\theta(0) = 0$, $\theta(1) = 1$ and θ is increasing (an increasing p makes it more likely to hold more edge in C). One interesting tuning lies in determining the critical probability p_c before a giant component composed of infected sensors forms.

By definition,

$$\theta(p) = \begin{cases} = 0, \ for\ p < p_c \\ > 0, \ for\ p > p_c \end{cases} \tag{1}$$

As illustred in Figure 1, this means that there are two phases: a subcritical phase where $p < p_c$ and a super critical phase where $p > p_c$ and wherein each individual has a non-zero probability of belonging to the giant component. Determining p_c

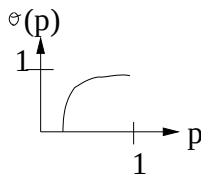

Fig. 1. Critical probability of percolation

is challenging and has lead to a great amount of literature [10], which is out of the scope of this paper. Broadly speaking, p_c depends of the network topology that is studied. Let herein consider one of the simplest topology referring to a 2-dimensional lattice (as the one used during our experiments). In such a topology, the probability is not known exactly but has been bounded [11].

Lemma 1. Supposing that the sensor does not gossip to an ascendant gossiper, the probability of percolation is subject to [12]:

$$\theta(p) \leq \begin{cases} \frac{4}{3}(3.p)^h \\ \frac{4(1-p)\sum_{h=8m}^{\infty} h(3(1-p))^{h-1}}{3^2} \end{cases} \quad (2)$$

With h that defines the number of hops from i and m referring to the length of the dual lattice $[-m, +m]$. This implies that the percolation takes place with $\frac{1}{3} \leq p_c \leq \frac{2}{3}$. This permits to answer to one sub-critical question which is related to how many 1-hops neighbours should be gossiped so that the network percolates. Nevertheless, additional analysis is still needed to detail the latency involved to obtain such a percolation.

4 Empirical Evaluation

In order to evaluate our gossip-style failure detector, we used the worldsens platform [7] that permits to both emulate the real hardware of a sensor and simulate a wireless network. We simulated a wireless network composed of 50 sensors organised into a lattice (5 x 10). In a nutshell, the hardware sensor that is emulated includes a Ti MSP430 16-bits micro-controller (48kB flash, 10240B RAM) and the CC2420 radio transceiver which operates at 2.4Ghz using the IEEE 802.15.4 protocol. We use the FreeRTOS[2] operating system upon which is implemented our gossip-style failure detector. Our objective is twofold: measuring the time devoted to detect a failure (Figure 2) and the resulting traffic (Figures 3 and 4). We restrict our evaluation to the uniform gossip ; one node is selected in a uniform and random manner (i.e., p = $\frac{1}{4}$). Measures are performed once the network is already configured, supposing that T_{gossip}= 2.5s and assuming 40-bytes gossips. We plot, three curves representing the minimum, maximum and average.

In Figure 2, the time devoted to detect a failure (expressed in terms of rounds,

[2] $www.freertos.org$

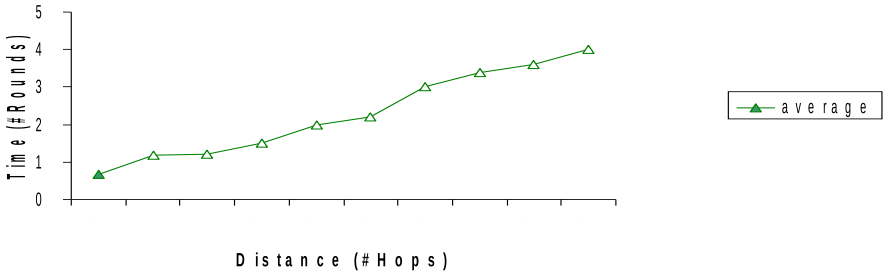

Fig. 2. Latency of a Failure Detection

i.e., as a multiple of T_{gossip}) depends of the distance separating the failing sensor and the detector. Naturally, the higher is the distance (in terms of hops) the greater is the average detection time. Although, the absence of synchronisation among the sensor may lead to a fortuitous fast (minimum plot) versus slow (maximal plot) dissemination of the gossip, a deeper analysis of the result showed us that the time is mainly centred on the average. The traffic related to the hello (Figure 3) and gossip messages (Figure 4) is expressed as a function of the sensor degree, over a period of 60sec. Obviously, the traffic increases with regards to the number of sensors in the vicinity. The traffic generated by the hello message is higher comparing to the gossiping. Both curves show a slope that is related to packet losts coming from the links failures. Note that this is especially visible with the gossip. This motivates us for privileging link-aware and degree-aware gossiping.

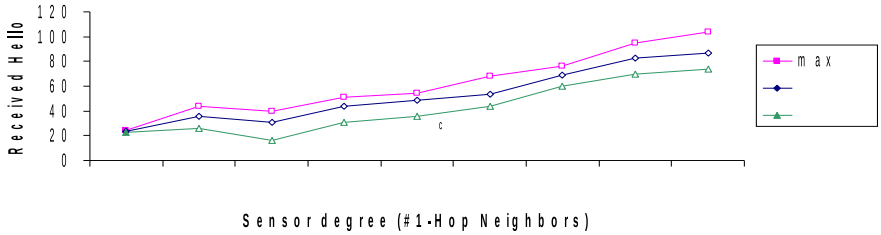

Fig. 3. Traffic related to the Failure Detection

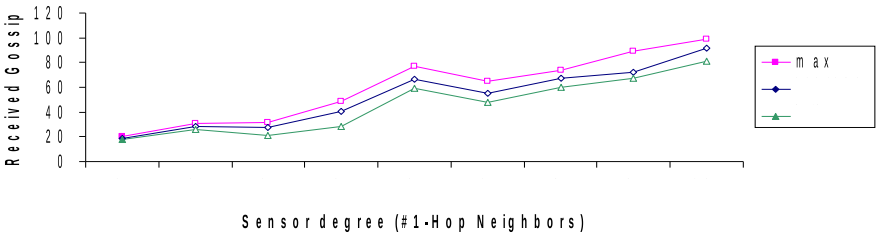

Fig. 4. Traffic related to the failure detection, i.e., heartbeating(a) and gossiping (b)

5 Conclusion

A failure detector constitutes a crucial component to ensure the dependability of a distributed system and more specifically of a wireless sensor network. We herein propose a gossip-style failure detector that provides hints about the failing processes running on the sensors. Meanwhile, we investigate several gossip policies involving a random and uniform selection of the gossiper as well as weighted gossiping that takes into account the link strength as well as the sensor degree. Relying on the percolation theory, we sketch an analysis of the proposed gossip assuming that the sensors are organized into a grid structure. In a parallel way, we empirically evaluate the traffic generated and the time devoted so as to detect failures. These experiments confirmed us the importance of considering a link-aware and degree-aware gossiping. In the near future, we attempt to empirically determine the threshold that ensures that the gossips are spread over the network.

References

1. Langendoen K., Baggio K., Visser A.: Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In proc. of IEEE international Parallel Distributed Processing Symposium (IPDPS) (2006)
2. Barrenetxea G., Ingelrest F., Schaefer G. and Vetterli M.: The hitchhiker's guide to successful WSN deployments. In proc. of the ACM conference on mebedded networked sensor systems (SENSYS) (2008)
3. Chandra T. D. and Toueg S.: Unreliable failure detectors for reliable distributed systems. Journal of the ACM (JACM) vol. 43, no. 2, pp. (1996)
4. Greve F., De Lima M. S., Arantes L. and Sens P.: A time-free byzantine failure detector for dynamic networks. In proc. of the European Dependable Computing Conference (EDCC) (2012)
5. Van Renesse R., Minsky Y., Hayden M.: A gossip-style failure detection service. Technical report, Cornell University, Ithaca, NY, USA, (1998)
6. Demers A., Gealy M., Greene D. and al.: Epidemic algorithms for replicated Database maintenance. In proc. of the ACM Symposium on Principles of distributed computing (PODC) (1987)

7. Haas Z., Halpern J.Y. and Li L.R.: Gossip-based ad hoc routing IEEE/ACM transactions on networking, vol. 14, no. 3, (2006)
8. Chelius G., Fraboulet A. and Fleury E.: Worldsens: development and prototyping tools for application specific wireless sensors networks. In International Conference on Information Processing in Sensor Networks (IPSN), (2007)
9. Gilbert E.N.: Random plane networks Journal of the society for industrial and applied mathematics, vol. 9, no. 4, 1961
10. Kesten H.: Percolation theory for Mathematicians. Birkhuser Pub., (1982)
11. Van Den Berg J. and Ermakov A.: A new lower bound for the Critical Probability of site percolation on the square lattice. In random structure and algorithms, vol. 8, no. 3 (1996)
12. Grosslauser M., Thiran P.: Networks out of control: models and methods for random networks. In: Lecture notes, EPFL, (2012)
13. Ruijing H., Sopena J., Arantes L., Sens P. , Demeure I.: Fair Comparison of Gossip Algorithms over Large-Scale Random Topologies. In: IEEE 31st Symposium on Reliable Distributed Systems (SRDS) (2012).
14. Friedman R. and Tcharny G.: Evaluating failure detection in mobile ad-hoc networks. In: International Journal Pervasive Computating and Communication (2009)
15. Tai A., Tso K. and Sanders W.: luster-Based Failure Detection Service for Large-Scale ad hoc Wireless Network Applications In International Conference on Dependable Systems and Networks (DSN) (2004).
16. Sens P., Arantes L., Bouillaguet M., Simon V. and Greve F.: An unreliable failure detector for unknown and mobile networks. In 12th International Conference on Principles of Distributed Systems (OPODIS) (2008)
17. Greve F., de Lima M. S., Arantes L., and Sens P.: A time-free byzantine failure detector for dynamic networks. In proc. of the Ninth European Dependable Computing Conference (EDCC), (2012)
18. Shostak S., Backer B.: Gossip and telephones. In: Discrete Mathematics, vol. 2, no. 3, (1972)
19. Taylor K., Golding R.: Group membership in the epidemic style. In: technical report. placePlaceTypeUniversity of PlaceNameSanta Cruz, (1992)
20. R. Freidman, D. Gavidia, L. Rodrigues and al.: Gossiping on MANETs: the beauty and the beast. In: ACM SIGOPS, Operating Systems Review (ACM OSR) vol. 41, no. 5, (2007)