

Evaluating Fault Tolerance Aspects in Routing Protocols for Wireless Sensor Networks

Daniel F. Macedo¹ Luiz H. A. Correia^{1,2} Aldri L. dos Santos^{1,3}
Antonio A. F. Loureiro¹ José Marcos S. Nogueira¹ * Guy Pujolle⁴

¹Computer Science Dept.
Federal Univ. of Minas Gerais
Belo Horizonte-MG, Brazil

²Computer Science Dept.
Federal Univ. of Lavras
Lavras-MG, Brazil

³Computer Science Dept.
Federal Univ. of Ceará
Fortaleza-CE, Brazil

⁴University Paris 6
Laboratoire d'Informatique de Paris
Paris, France

{damacedo,lcorreia,aldri,loureiro,jmarcos}@dcc.ufmg.br, pujolle@rp.lip6.fr

Abstract

Fault tolerance is an essential requirement in the design of protocols and applications for Wireless Sensor Networks (WSNs) since communication and hardware failures are frequent. In this paper we studied the resilience of routing protocols for continuous data dissemination WSNs in face of faults. The main causes of silent failure are presented, including some security attacks. Those failures are classified according to extension and persistence, and such classification is used to evaluate routing protocols for continuous data dissemination networks. Results show that failures under a large region of the network are the most damaging. The paper also shows how routing protocols may save energy by temporarily turning off disconnected nodes.

1. Introduction

Wireless Sensor Networks (WSNs) are a subclass of traditional ad hoc wireless networks, and consist of a large number of sensor nodes, composed of processor, memory, battery, sensor devices and transceiver. These nodes send monitoring data to an access point (AP) responsible for forwarding data to the users [7]. Unlike traditional ad hoc networks, in general it is not possible to replace or recharge node batteries due to the number of nodes deployed or inhospitable environmental conditions. Hence, energy conservation is a critical factor in WSNs.

Fault tolerance is one of the requirements of a dependable system [14]. According to the British Computing Society report [21], the development of dependable systems is currently one of the great challenges in computing. WSNs

are propitious to failure due to events such as node destruction, link quality degradation, among others. Since those networks may be employed in hostile environments, such as disaster sites, nodes can fail due to landslides, collapsing buildings, floods or other natural agents. Failures may also occur in the communication, caused by changes in weather or movement of objects near the nodes, which may block the signal, or due to malicious agents intending to disrupt network operation. Since nodes frequently interact with each other (since nodes cooperate to perform their tasks), application software is prone to failure caused by faults in other nodes. Thus, protocols and application software must be developed with fault tolerance mechanisms.

In routing protocols, failures arise as broken routes, and occur due to communication or hardware faults. Upon detecting a broken route, routing protocols must identify another operational route, thus allowing traffic flow to be restored. Routing failures are more severe in WSNs than in ad hoc networks, as protocols usually build only one route, since all communications in WSNs head towards the AP. Hence, a failed route may affect a substantial portion of the network. In ad hoc networks, on the other hand, nodes may communicate with any node, thus a failed route will affect only the communication using that route.

Data flow in WSNs usually follows a pattern, since data is preprocessed locally and then sent to the AP. This data flow can be categorized according to its frequency [20]. In *event-driven networks*, communication is sporadic, occurring only when an event of interest is detected. Such networks are used in wildlife monitoring, intrusion detection, among others. In *continuous dissemination networks*, nodes periodically send data to the AP. In those networks it is possible to build a “map” of the current state of the environment, which can be later used to study time and space variations in the observed phenomena. Such networks are

*In sabbatical period at universities of Evry and UPMC/Paris6/LIP6, France.

employed in environmental studies, intelligent traffic, industrial plants, among others.

Due to the intrinsic differences in traffic, and since WSNs should spent as low energy as possible, routing protocols are usually designed to operate on a single network class. Continuous dissemination networks tend to employ proactive protocols, since nodes are periodically sending data to the AP. In event-driven networks, in contrast, routes are build only when an important event is detected, since the energy burden of the periodic reconstruction of routes is too high for this scenario. The same fact occurs with fault-tolerance mechanisms. In continuous data dissemination networks, proactive protocols are justifiable since there is always data being sent, while fault tolerance protocols designed for event-driven networks tend to be reactive, operating only when a failure occurs.

In this paper we study the performance of routing protocols for continuous dissemination networks in faulty scenarios, where *silent* communication and hardware faults occur (that is, no packets are sent during failure situations). The main causes of failure are presented, and are then categorized according to extension and persistence. This characterization was used to extract common aspects of failures, simplifying the evaluation of the protocols. Next, a performance evaluation through simulation was performed for three routing protocols.

This text is organized as follows. Section 2 presents the related work. Section 3 presents an overview of the protocols evaluated and describes its fault-tolerance algorithms. Section 4 describes the main causes of silent failures in WSNs. Section 5 categorizes the causes of failures presented in the previous section. This categorization is then used to evaluate three routing protocols in Section 6. Finally, Section 7 draws the conclusions and future work.

2. Related Work

Avizienis et al. present a taxonomy of failures, which also encompasses security issues [1]. Hollick et al. present the challenges in the development of fault tolerant systems for WSNs, ad hoc networks and cellular networks, and list the requirements which should be met by fault-tolerant protocols in such networks [6]. Koushanfar et al. present an overview of fault tolerance in WSNs, mainly focused on hardware components, such as sensors and actuators. The authors also summarize the existing techniques for detection and correction of byzantine failures in sensor readings [12].

Fault tolerance in protocols for WSNs has been widely studied. The first protocols developed [9, 25] were concerned with failures caused by energy depletion, and proposed only mechanisms to increase the life time of a node by distributing the energy spent among as many nodes as

possible. Other protocols were designed to be resilient against communication faults caused by failed nodes [8, 10]. Those protocols mitigate failures by sending multiple copies of data among different routes, thus increasing the probability of correct reception at the AP, albeit at a higher energy cost. A study of the probability of the reception of a packet at the AP given multiple routes with different degrees of similarity was performed by Ganesan et al. [5]. This study showed that partially disjoint routes are as effective as totally disjoint routes, although they spend much less energy to be established.

Since the cost of maintaining multiple routes is significant, some protocols define only one high-quality route. De Couto et al. presented a modification to the ad hoc routing protocol DSR, which calculates the reliability of a route [4]. When setting up a new route, the modified DSR calculates the quality of the route, using the received signal strength of each hop along the route. Nodes always choose the route with the best quality, thus increasing the probability of a successful delivery. Alec Woo et al. [26] proposed a similar mechanism, adapted to perform efficiently in WSNs.

Given the occurrence of a failure, it is necessary to identify an alternative route. Vieira et al. proposed two protocols to mitigate failures due to energy depletion [23]. In the first algorithm, called *Smart-Sink*, the AP notifies nodes to modify its routes whenever a failure occurs. In the second, nodes build a list of “second-best routes”. Upon the detection of a failure, one route in this list is selected to become the default route. The authors, however, do not specify how nodes identify node failures. Khanna et al present a modification in SPIN which adds backup routes to the protocol [11]. Although the protocol was devised to provide fault tolerance, Khanna et al. only present analytic results, and do not define the failure model considered. In this paper, on the other hand, we present a clear definition of the failure model, and present an extensive evaluation through simulation of the selected three protocols.

Another alternative to mitigate failures consists of fault forecasting and prevention. This focus, however, is not very popular in WSNs, since nodes are low-cost and have severe resource constraints, hindering the deployment of diagnosis hardware and software. Some studies, however, identified the correlation of faulty sensors with the failure of a sensor node [22].

3. Evaluated Protocols

We evaluate the performance of three routing protocols for continuous dissemination networks (TinyOS Beaconing, EAD and PROC) in face of failures. Those protocols were selected because they are suitable to continuous data dissemination networks, and provide different levels of fault tolerance. Due to the limitations of simulators, we were not

able to evaluate protocols which operate with link quality estimates [4, 26]. TinyOS Beaconing is a simple routing protocol, which is the standard routing for the Mica2 architecture [13]. EAD is a protocol devised to decrease the energy consumption [2]. This protocols provides an insight of how energy-aware techniques behave in the presence of failures. The third protocol, called PROC, possesses internal mechanisms to mitigate failures [15].

3.1. Protocol Overview

TinyOS Beaconing is a protocol used in the Mica Motes platform [13]. This protocol periodically creates a minimum distance tree rooted at the AP. A beacon message is propagated by the AP towards nodes in order to create a routing tree. Nodes snoop traffic to estimate the link quality of their neighbor nodes. In order to reduce retransmissions, only nodes with good link quality are used to route messages. TinyOS Beaconing was not designed with fault tolerance mechanisms, although the periodic recreation of routes provides some degree of fault tolerance, as explained below.

Boukerche et al. proposed a routing algorithm, called EAD (*Energy-Aware Distributed routing*), which creates a routing tree that maximizes the number of leaf nodes [2]. This tree ensures that all nodes are able to send messages to the AP. Leaf nodes, which do not need to send messages, turn their radios off in order to extend network lifetime. The protocol also uses backoff timers based on current node energy for decreasing collision probability. As in TinyOS Beaconing, EAD uses the periodic reconstruction of routes to provide fault-tolerance. In EAD, however, traffic is concentrated in a few nodes, hence failures in those nodes will be more severe than in “ordinary” nodes.

The PROC (*Proactive ROuting with Coordination*) protocol was developed with the goal of reducing energy consumption and increasing network lifetime [15]. PROC creates a routing tree, called *backbone*. The structure of the backbone is influenced by the application, which defines which nodes are more suitable to route data. In PROC, the *backbone* is periodically rebuilt, in a process initiated by the AP. The protocol provides fault tolerance using link layer acknowledgments. Whenever the number of data packets not acknowledged reaches a certain threshold, PROC determines that the route is failed, and selects a new route. As in EAD, the failure of nodes which concentrate traffic, the *backbone* nodes, will be severe. The proactive probe of nodes using link layer acknowledgments, though, mitigates this issue.

3.2. Fault Tolerance Mechanisms

To better understand the behavior of the protocols evaluated, it is important to understand how each protocols deals with failure situations. This section describes the fault tolerance mechanisms implemented in each protocol.

The periodic recreation of routes is a fault tolerance scheme employed in all three protocols. In each execution of the route creation algorithm in EAD, PROC and TinyOS Beaconing, routes are completely reconstructed, using only active nodes. This process occurs as follows: The AP sends a message indicating that routes must be recreated. Each node broadcasts this message to its neighbors, allowing nodes to identify which neighbors are active at that time. By forming routes only among the currently active nodes, that is, the nodes that correctly forwarded the last route recreation message, the protocols avoid routes passing through failed nodes.

The interval between each route recreation must be adjusted according to the expected degree of fault tolerance and the amount of energy spent at each route recreation. As this process requires sending messages to every node in the network, route recreation is an energy-intensive process. Thus, in order to save energy, routes should be sporadically recreated. However, if routes are frequently recreated, the degree of fault tolerance increases since failures are detected earlier.

PROC allows earlier detection of failures than EAD and TinyOS Beaconing, since it employs proactive monitoring of sensor nodes with the use of *heart-beat* messages, similar to ping-pong messages. For every data packet sent (ping message), an acknowledgment (ACK) must be returned by the receiver (pong message). A counter registers how many packets in a row were not acknowledged. Whenever this counter reaches a certain threshold, PROC assumes that the node has failed, and rebuilds its routes. The use of ACKs avoids sending messages specifically to identify failures, saving energy. Usually, MAC protocols for WSNs will not use ACKs in order to decrease energy consumption. However, the use of ACKs marginally increase energy consumption [18], justifying its use to provide fault-tolerance.

The minimum number of lost packets (*threshold*) that determine a failure can be calculated using the probability of a packet being lost ($PER - Packet Error Rate$)¹. The threshold must be such that the number of false positives (transmission errors accounted as node failures) is near to zero:

$$PER(s)^{threshold} = P, P \approx 0 \quad (1)$$

High threshold values will increase the certainty of a node failure, but more data will be lost before the failure is detected. Lower threshold values, however, will allow

¹ $PER(s) = 1 - (1 - BER)^s$, where s is the size of the packet sent, and BER is the mean bit error rate.

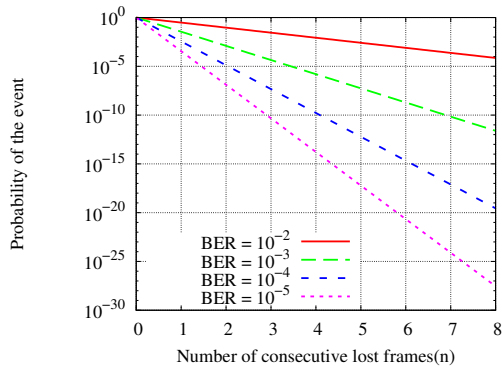


Figure 1. Probability of n consecutive transmission errors for different BERs.

earlier detection of faults, but false positives will be more frequent. Figure 1 shows the probability of n consecutive, frame losses, calculated using the equation 1, for different bit error rates (BER), in 36 byte frames. Figure 1 shows that the probability of false positives drops sharply as n increases. As an example, the CC1000 radio, employed in the Mica2 nodes [13], has typical bit error rates of 10^{-3} [3]. In the performance evaluation presented in Section 6, the threshold value in PROC is set to two. For the standard BER in Mica2 nodes, false positives will account for 0.001% of the detected node failures.

4. Failure in WSNs

This section identifies the main causes of communication failures in WSNs. In this study we do not consider communication failures due to invalid results (erratic failures) produced by sensor nodes. Valid routing messages can be ensured with error correction codes and formal validation of the routing protocols, thus we assume that protocols are correct and messages are correctly received. Hence, we focus our study on silent failures, caused by packet not being received at all, or due to node failures. The following failures were identified:

Atmospheric phenomena – Changes in weather modify the signal propagation, which may in turn cause communication errors as signal strength decreases [22]. Several environmental conditions such as humidity, temperature, among others, modify signal propagation. As weather is constantly changing, communication quality varies along with time.

Mobile sources of interference – Other devices operating at similar frequencies or even vehicles, animals and humans may interfere with communicating nodes [19]. As WSNs usually employ frequencies in the ISM (*Industrial, Scientific and Medical*) range, which do not require li-

censing for operation, WSNs are exposed to interferences of other devices operating in those frequencies. In order to decrease per-unit cost, sensor nodes usually employ single channel radios, with a fixed modulation scheme. Those constraints hinder the use of dynamic selection of frequencies and modulation, detection of low-interference channels and frequency hopping schemes [24, 17].

Natural disasters – Sensor nodes may be deployed outdoors or in disaster locations, thus being exposed to landslides, floods and earthquakes. Those events may cause massive destruction of sensor nodes by permanently damaging hardware components. Unlike failures caused by atmospheric phenomena, failed nodes due to natural disasters will permanently be non-operational.

Accidental breakage – Sensor nodes can be accidentally destroyed, for example due to animals trampling over nodes, or falling trees. Since nodes will be several meters away from each other, only one node tends to fail at a given time.

Processor crashes – The application software may contain programming errors, which might lead the processor to crash situations. Embedded systems use cooperative multi-task schedulers or run-to-completion schedulers, thus faulty software may block the processor. To avoid such situations, microcontrollers employ watchdogs², which restart the processor if a software malfunction occurs. Thus, nodes will be unavailable for a finite amount of time, then will return to normal operation.

Malicious failures – WSNs are prone to malicious failures due to security attacks, aimed at disrupting network operation, caused by an outsider or by a corrupted node. This article does not evaluate security protocols. However, some security attacks can be avoided or partially recovered with the use of fault tolerance techniques that avoid routes passing through areas under attack [27]. In this work we use fault tolerance techniques to avoid the following denial of service attacks: *interference attacks*, *collision attacks*, and *sinkhole attacks*. These attacks behave like silent faults, which are the aim of this work.

Energy depletion – energy depletion may generate communication failures. Usually, batteries will not be replaced, since WSNs are employed in harsh environments where the presence of an operator is prohibitive, or the number of nodes deployed makes battery replacement a daunting task. Since sensor nodes have limited resources, current nodes do not allow a reliable measurement of the amount of remaining energy stored in the battery, hence nodes are unable to notify their neighbors when their energy reserves are nearly over.

Our study does not encompass energy-related failures, since those are difficult to model. Energy consumption

²A watchdog is a timer that must be periodically restarted by the application, or the processor reboots.

tends to be homogeneous in the nodes' vicinity, as protocols tend to balance energy consumption among nodes in order to increase network lifetime [25]. This leads to nodes leaving the network nearly at the same time, thus nodes will tend to fail simultaneously. Energy consumption is hastened as energy gets scarce, since traffic is concentrated in fewer nodes. Due to difficulty of modeling such situations, energy-related failures are not considered in this work.

5. Failure Grouping

This section divides the failures described in Section 4 according to common characteristics. This characterization, summarized in Table 1, aids the performance evaluation of routing protocols presented in Section 6. Failures are characterized according to persistence and extension:

Persistence – Indicates if a node will resume correct operation after its failure (*transient failures*), or if the node will fail indefinitely (*permanent failures*) [1]. From a routing perspective, transient failures occur when nodes are out of service for a few minutes, while in permanent failures nodes are out of service for hours. Hence, failures due to atmospheric phenomena, for example, are classified as permanent.

Extension – Relates to the number of failed nodes. Failures can be *isolated*, when only one node fails, or *grouped*, when various nodes in a region fail. The latter is more severe, since it significantly decreases the number of nodes able to route data in a vicinity. Figure 2 gives an example of both types of failure (arrows denote routes).

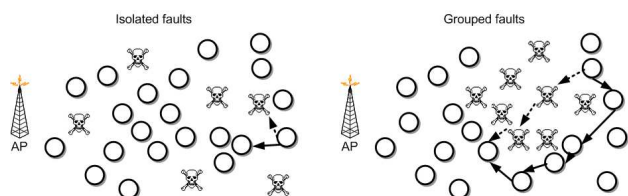


Figure 2. Example of node failures, classified according to their extension.

Malicious failures due to collision and sinkhole attacks can significantly vary their persistence according to the attacker's intent; those can be brief, in order to avoid detection, or can be long, in order to increase the disruption produced. Thus, those failures are classified as both permanent and transient. Interference attacks, on the other hand, will always be permanent, since their disruption potential increases, even though the attack is detected.

Cause of failure	Persistence	Extension
Atmospheric phenomena	permanent	grouped
Mobile sources of interference	transient	isolated
Natural disasters	permanent	grouped
Accidental breakage	permanent	isolated
Processor crashes	transient	isolated
Interference attacks	permanent	grouped
Collision attacks	both	isolated
Sinkhole attacks	both	isolated

Table 1. Failure characterization, divided by their causing agents.

6. Evaluation

The three protocols were implemented in the simulation environment NS-2 [16]. We simulated an homogeneous network, composed of sensor nodes configured similarly to the Mica2 platform, running the TinyOS operating system [13]. The application simulated has traffic characteristics similar to the sensor network deployed in Great Duck Island for ecosystem and bird studies [22]. In this network each sensor sends a data message of 36 bytes of size every 70 seconds. Those messages are sent to the AP, which forwards sensed information for further analysis. In this article we simulate only the interaction of the sensor nodes with the AP.

The medium access protocol employed is a modified version of the IEEE 802.11 protocol, which emulates the behavior of B-MAC [18]. Bandwidth is limited to 12kbps, and radio parameters were adjusted to resemble those of CC1000 [3], the radio used in the Mica2 architecture. The route recreation interval used for EAD and TOSB (a simplified version of TinyOS Beaconing without link quality estimators) was 120s, while for PROC this interval was set to 180s. Those values, which yield the best performance for each protocol, were empirically determined in [15].

The simulated network consist of 150 nodes deployed in a square area, measuring 70m on each side. The AP is located at the corner of the area, in order to maximize path length. The network operates without failures for 1500s, allowing the protocols to reach their stationary state. After that, a failure occurs, and the simulation continues for 1500s, allowing routing protocols enough time to recover from failures. In the scenarios where isolated failures occur, failed nodes are randomly selected. In the grouped and permanent scenario, a central point is defined, and all nodes within a given radius of this point fail.

The evaluated metrics are average end-to-end delivery rate, average latency, average hop distance towards the AP, average energy consumption and throughput. The energy

consumption metric only considers nodes which are operational by the end of the simulation. All results are the mean values of 33 simulations, plotted with 95% confidence values. Next, we show the results achieved for transient and isolated failures, permanent and isolated failures, and permanent and grouped failures. Due to space limitations, our analysis is mainly concerned with energy consumption and average delivery rates, since those metrics are the most important in fault-tolerance techniques for WSNs. We comment briefly on the other metrics.

6.1. Transient and Isolated Failures

Transient failures were evaluated under three aspects: route recreation interval, failure time and number of failed nodes. The routing recreation interval will affect the degree of fault tolerance, since EAD and TOSB rely on route reconstructions to recover from failures. In this scenario 20 nodes fail for 120s. The first set of simulations evaluate the role of route recreation interval in performance, thus we varied route recreation times from 60 to 300s. Figure 3 shows that nodes tend to consume more energy when route updates are frequent, as expected. The average delivery rate, shown in Figure 4, decreases for larger route recreation intervals. This reduction is less pronounced in PROC, since this protocol proactively probes routes, allowing faster identification of failures. For all protocols, average delivery rates increase for 300s recreation intervals, since network load decreased, and less packets were dropped due to full packet queues. Latency also decreased for all protocols as route recreation intervals increased, since there was a lower load imposed on the network.

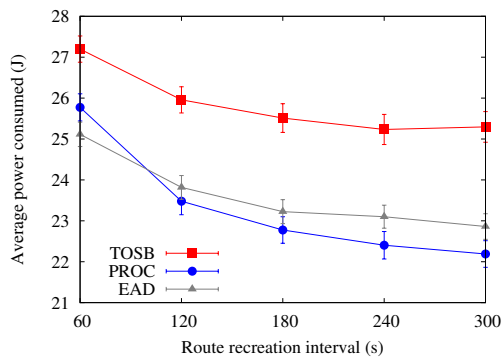


Figure 3. Average power consumed varying the route recreation interval.

Next, we evaluated how failure time affects the performance of the protocols. All protocols evaluated presented a drop in throughput during the failure interval, which is usually recovered after routes are recreated (Figure 5).

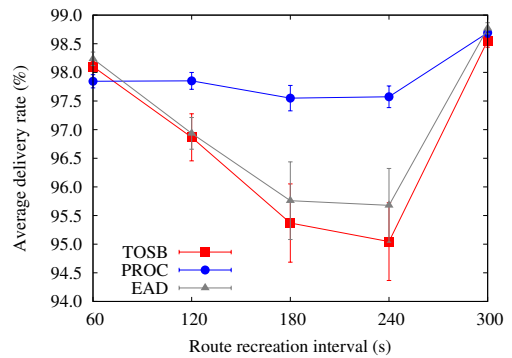


Figure 4. Average delivery rate varying the route recreation interval

Again, PROC presented higher delivery rates (around 0.5% higher), as shown in Figure 6. Since EAD and TOSB showed delivery rates slightly lower than PROC's, we conclude that periodic routing recreation is enough to guarantee good results in this scenario. The amount of energy consumed decreased with longer failures, since nodes had to route less data. PROC was the most energy-efficient protocol, consuming around 22J of energy, while EAD and TOSB consumed 4% and 14% more energy than PROC, respectively.

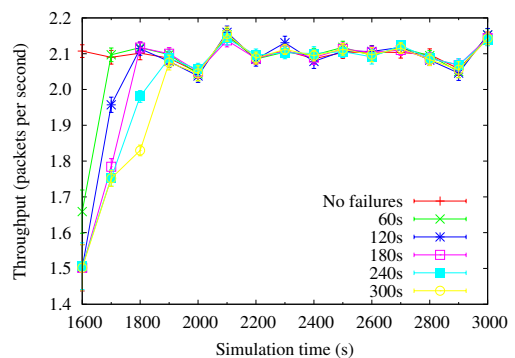


Figure 5. PROC's throughput varying the time of failure.

Finally, we varied the number of failed nodes from 25 to 100 nodes, with increments of 25 nodes. All protocols behave similarly in this scenario. Throughput decreased when more nodes failed, but again it was completely restored after 200s. The drop in throughput is proportional to the number of failed nodes, as exemplified in Figure 7. The proactive mechanism in PROC allowed this protocol to recover from failures faster than the other protocols evaluated, which provided a 0.5% increase in average delivery rates. This be-

