A Decentralized Damage Detection System for Wireless Sensor and Actuator Networks

Igor L. Santos, Luci Pirmez, Luiz R. Carmo, Paulo F. Pires, Flávia C. Delicato, Samee U. Khan, *Senior Member, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—The unprecedented capabilities of monitoring and responding to stimuli in the physical world of wireless sensor and actuator networks (WSAN) enable these networks to provide the underpinning for several Smart City applications, such as structural health monitoring (SHM). In such applications, civil structures, endowed with wireless smart devices, are able to self-monitor and autonomously respond to situations using computational intelligence. This work presents a decentralized algorithm for detecting damage in structures by using a WSAN. As key characteristics, beyond presenting a fully decentralized (in-network) and collaborative approach for detecting damage in structures, our algorithm makes use of cooperative information fusion for calculating a damage coefficient. We conducted experiments for evaluating the algorithm in terms of its accuracy and efficient use of the constrained WSAN resources. We found that our collaborative and information fusion-based approach ensures the accuracy of our algorithm and that it can answer promptly to stimuli (1.091 s), triggering actuators. Moreover, for 100 nodes or less in the WSAN, the communication overhead of our algorithm is tolerable and the WSAN running our algorithm, operating system and protocols can last as long as 468 days.

Index Terms—Decentralized algorithm, information fusion, structural health monitoring, wireless sensor and actuator networks, smart cities

1 Introduction

Advanced sensing systems play a major role as enabling technologies to build smart cities [15]. In smart cities, infrastructures, such as bridges and buildings, are equipped with smart sensing and actuator devices interconnected via wireless links composing a wireless sensor and actuator network (WSAN) [1]. The WSAN nodes are able to measure a variety of environmental parameters, process the sensing data locally, work in a collaborative way, make decisions on the occurrence of relevant events, and react to such events performing local control actions or sending warnings to remote operators. Applications running on top of WSAN are able to provide a wide variety of services to the citizens.

An important application domain in smart cities is the smart building [16]. A smart building can be defined as a structure in which technologies and processes are used to increase security and comfort for occupants, to minimize power consumption and to increase operational efficiency for its owners. Among the features to increase the security of building occupants, a major requirement is to monitor the building structural integrity. To assess the structural health of buildings and make proper decisions to keep such structures in good service, structural health monitoring (SHM) [4] techniques are employed.

The SHM is an emerging technology, dealing with the development and implementation of continuous and reliable monitoring systems for civil infrastructure using a dense WSAN. The sensing devices commonly used for SHM applications are strain gauges, anemometers, thermistors, and accelerometers. These devices collect data from the monitored environment, such as vibration measurements using accelerometers, and deliver them as digital data. Therefore, by processing this data, the SHM techniques allow the detection, localization and extent determination of damage in structures.

Most of the SHM algorithms found in the literature employ centralized architectures with sensing nodes transmitting messages to a centralized entity wherein the damage detection processes effectively happen [4]. One of the major drawbacks of a centralized architecture is the additional delay on top of control response time, because of the aggregated communication delay between sensor and actuator nodes. A feasible approach to overcome this architectural restriction is to perform damage detection processes inside the WSAN nodes. It is noteworthy to mention that by taking the damage detection processes off the centralizing entity and incorporating them into the WSAN, new challenges arise.

A key challenge is how to maximize WSAN lifetime while maintaining the functionality of damage detection within the network. Once in the decentralized approach the WSAN nodes have the additional burden of performing damage detection processing beyond just monitoring physical variables and transmitting messages, the development of solutions to save the network energy becomes even more

- *I.L. Santos, L. Pirmez, P.F. Pires, and F.C. Delicato are with the Universidade Federal do Rio de Janeiro—21941-901, Rio de Janeiro, Brasil. E-mail: {igorlsantos, luci.pirmez, paulo.f.pires, fdeli-cato}@gmail.com.*
- *L.R. Carmo is with the Instituto Nacional de Metrologia, Qualidade e Tecnologia (Inmetro)–Av. N.S. das Graças, Duque de Caxias, Brasil. E-mail: lfrust@inmetro.gov.br.*
- *S.U. Khan is with the Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58108-6050. E-mail: samee.khan@ndsu.edu.*
- *A.Y. Zomaya is with the Advanced Networks Research Group, School of Information Technologies, University of Sydney, Sydney, NSW 2006, Australia. E-mail: zomaya@it.usyd.edu.au.*

Manuscript received 31 Jan. 2015; revised 3 Aug. 2015; accepted 7 Sept. 2015. Date of publication 22 Sept. 2015; date of current version 13 Apr. 2016. Recommended for acceptance by S. Hu, G. Betis, R. Ranjan, and L. Wang. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TC.2015.2479608

pressing. Such reduction in energy consumption is important since WSAN nodes have very limited energy resources, often supplied by non-rechargeable batteries. One possible approach used to reduce the WSAN energy consumption is developing energy efficient techniques, protocols and strategies [1]. Among these techniques, information fusion algorithms are a promising option [2], which we adopted in this work, as well as in our previous work [5]. Information fusion algorithms exploit the processing capacity of the sensor nodes and the inherent redundancy of the sensor-generated data to reduce the need of data transmissions, thereby trading the communication energy costs by processing energy costs. Since radio transmissions are major sources of energy consumption in WSANs, while processing cycles are minor sources [4], the approach of data reduction uses the available energy efficiently to extend the WSAN lifetime.

In the algorithm proposed in this paper, we make use of an information fusion process that acts in the three data abstraction levels (measurement, feature and decision) often considered in the information fusion literature [2]. This is usually the case of a fully in-network SHM process (from data collection to decision making about structural damage detection), thus our proposal is classified as a multilevel fusion process, according to [2]. It is noticeable, specifically in the SHM literature, the widespread use of damage coefficients as results of multilevel fusion processes [3], [4], [5]. In general, the numerical value of damage coefficients is understood as a representation of the damage in the monitored structure. The immediate benefit of using information fusion techniques for calculating a damage coefficient is that only such coefficient, with a reduced amount of data, is transmitted for further analysis. Consequently, less energy is spent in the WSAN due to data transmissions.

Besides maximizing WSAN lifetime, a second key challenge is how to develop algorithms capable of supporting accurate damage detection, in terms of the rate of success in the detection of damage in different structures. Existing damage detection algorithms based only in the variation of modal frequencies, such as our own previous work [5], are generally less accurate when applied to certain structures whose modal frequencies do not shift significantly in the presence of damage. However, for these same structures, other structural features can also be assessed, such as the vibration energy related to the modal shapes of the structure, which can be measured by the amplitude of the corresponding modal frequencies. The act of assessing information from two different sources simultaneously to infer decisions is understood as a type of information fusion called cooperative fusion [2], and it was not explored in our previous work [5]. A damage coefficient calculated from both frequency and amplitude shifts can be interpreted as a local decision about the existence of damage on the structure. Moreover, this decision inferred locally by each individual sensor node can also be exchanged among other WSAN nodes. Through specific collaboration procedures, the nodes can communicate their local decisions among themselves, evaluating their neighbor decisions and reaching a consensus, which may be more accurate than the single local decision of a node. Moreover, such collaboration may comprise consensual multilevel decisions (e.g., floor level, part of building level, whole building level), that take into account the cooperation among nodes in the same level to have a broader view of the monitored structure. Finally, without reducing the data exchanged, the collaboration would require an unfeasible amount of energy spent on transmissions. Therefore, the use of a damage coefficient that uses a small amount of data fosters the collaboration among the WSAN nodes. Our previous work [5] performs a collaboration procedure in a single level, while in our current work we use a multilevel consensus collaboration.

In this context, our work proposes a decentralized and information fusion based damage detection algorithm for civil structures using WSANs. In the algorithm, the nodes have only a partial view about the integrity of the structure and collaborate among themselves to reach a consensual multilevel decision to have a broader view of the result of damage detection. Thus, our proposal can be also classified as a localized algorithm [7]. Another key characteristic of our algorithm is that it is based on the concept of cooperative information fusion. Moreover, a new damage coefficient, called cooperative damage coefficient (CDC), is proposed as a representation of local decisions made by each WSAN node. The CDC describes the damage using only a few bits representing which modes of vibration had variations, in terms of both frequency and amplitude shifts simultaneously.

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 presents the proposed algorithm. Section 4 describes the implementation of our algorithm in WSAN nodes. Sections 5 and 6 present our experimental evaluation methodology and the results of performed experiments. Finally, Section 7 presents conclusions and future research directions.

2 Related Work

In this section, we present existing approaches for damage detection algorithms that are fully decentralized, highlighting their advantages and drawbacks in relation to our work. Gao et al. [3] proposed a strategy for damage detection and localization on a truss structure. They adopt a WSAN hierarchically divided into clusters, reducing the amount of transmissions in the network, since it is not necessary that all sensors forward their data to the sink. At the same time, since the amount of data sent to the sink is smaller and already condensed, there is a reduction in the time of damage identification. A differential of our work regards our proposed damage representation, the CDC, which is able of representing damage using a smaller amount of data than in [3], and uses information from both amplitude and frequency shifts simultaneously. Thus, our proposal achieves faster execution of control actions and energy saving, as well as a higher accuracy in the result of damage detection.

Santos et al. [5] proposed a decentralized algorithm called Sensor-SHM, for damage detection, localization and extent determination in civil structures using a WSN. Sensor-SHM uses a cluster-based topology. Our proposal retains the original idea of [5] regarding to the strategy of comparing the respective currently measured modal properties with the modal properties measured at the beginning of the algorithm operation (healthy states). However, unlike [5], the decision making process of damage detection in our

work is performed collaboratively between the neighboring sensor nodes without the help of the sink node, neither the cluster-head, and we use a flat topology. Our proposal also includes the concept of cooperative information fusion, using information of both natural frequency shifts and frequency amplitude variation simultaneously for detecting damage, while in [5] only the natural frequency shifts is used to calculate the damage coefficient. In addition, our damage representation, CDC, differs from the damage coefficient of [5], which uses a larger amount of data for representing damage. Our CDC represents an advance in relation to [5]. The CDC is used for indicating which modal frequencies have changed using only an amount of data in the scale of bits. Therefore, the CDC summarizes only the information about all frequencies/amplitudes that shifted due to damage. The damage index used by Sensor-SHM requires more bytes for being represented than our proposed CDC. Therefore, our work has a higher degree of data reduction than [5].

3 Description of the Proposed Algorithm

3.1 Assumptions

The procedures of our algorithm are performed by the set of WSAN nodes (excluding sink nodes) deployed over a monitoring area of a structure. We consider that each node may be equipped with both sensing and actuating devices. We consider two roles for the nodes: (i) sensor (SENs) and (ii) actuator (ATNs). Such roles are set during the algorithm setup phase, when the respective logical capabilities of nodes are defined as active or non-active. A same node may have only the SEN role set, or only the ATN role or even both SEN and ATN roles simultaneously. A SEN is considered the basic "sensing, processing and decision unit" in the WSAN. Such nodes are equipped with at least one physical sensing device, and have their logical sensing capability active. Similarly, each ATN is considered the basic "actuation unit" in the WSAN. Such nodes are equipped with at least one physical actuation device. Both SEN and ATN nodes have unique non-zero network addresses.

We identify the single sink node (SKN) in the network by the address 0, being also a role set during the setup phase. The SKN serves as a gateway between the WSAN and external networks. It has no sensing units and unlimited power supply. Therefore, the SKN is an intermediate between the human operators and the WSAN nodes, disseminating commands in the WSAN and collecting reports from actions performed on the structure. Finally, before the algorithm starts, the network must be already deployed on the structure to be monitored.

3.2 Algorithm Overview

Our algorithm encompasses a sequence of procedures performed in two main phases. Initially, a setup phase is performed, starting right after the network is physically deployed. The procedures of this phase are performed for all the network nodes and consist on setting the algorithm initial parameters and performing required initializations. Thereafter, the monitoring cycle phase (Section 3.4) starts and consists in a main loop. In our algorithm, the structural monitoring is performed periodically, and each period is based on a new data collection.

The monitoring cycle starts when the SKN schedules the beginning of the next data collection, by disseminating a message for the whole network, assuming that the network is properly synchronized. Such approach does not generate a significant energy/communication overhead on the WSAN since the data collection is not frequent. In SHM applications, the frequency of data collection is typically set at once an hour or once a day, because damage progression in civil structures is typically slower than such interval [4]. Then, data collection starts at the time scheduled by the SKN. All the SENs collect acceleration data from the structure simultaneously. Next, each SEN performs a fast Fourier transform (FFT) [4] over the collected data and, from the resulting power spectrum, extracts the current values of natural frequencies and amplitudes. The parameters required to perform data collection and feature extraction were set during the setup phase. Next, the SENs perform a local evaluation of the CDC. First, the current values of frequency and amplitude are compared with the respective reference (healthy) values (obtained during the setup) and shifts (deviations from the healthy values) are calculated. The CDC is calculated from such shifts that exceed a given threshold (set during the setup phase). According to the rules used for local evaluation of the CDC, each SEN may locally decide that there is damage on the structure and, therefore, neighbor SENs must collaborate (exchanging their respective CDCs) among themselves for reaching a consensus on this decision-making.

After a consensual decision about the structure integrity is reached, SENs send messages for the respective ATNs, informing which control actions should be performed. Finally, on receiving a message from the SEN, the ATNs perform the respective actions and send their reports to the SKN, concluding the monitoring cycle.

3.3 Setup Phase

The setup phase encompasses six procedures that are performed for all the nodes in the network. This phase starts with the *boot()* procedure that represents the hardware initializations of each node, required for making the node operational and ready to run the algorithm. During the boot procedure the *NodeID* parameter is set. The *NodeID* stores a unique identification for each node in the network. Next, the *set_role()* procedure starts, setting the logical sensing and actuating capabilities of the node. These capabilities identify, during the execution of the algorithm, the respective SEN, ATN or SKN role for a node. All the roles can be set by human operators and disseminated in the network through the SKN. After, the *init()* procedure is performed, which consists of setting the initial values of the following parameters (that can be specifically defined for monitoring different structures): (i) number of modes of vibration of interest (*NModes*), (ii) frequency and amplitude variation limits (*LFreqs* and *LAmps*, respectively), (iii) number of collected samples (*NSampl*), (iv) sampling rate (*SamplRate*), and (v) network addresses of the neighbor nodes (*NeNodeID*). The neighborhood of a node is defined from the application point of view. An expert on the application can divide the monitoring area into small sensing zones and assign each zone to a node. Node neighborhood is set based on the sensing zones dedicated to each node.

In our algorithm, we consider the assessment of a finite number of vibration modes of the structure. Each vibration mode has two features: a corresponding modal frequency (value) and an associated modal energy (amplitude value). The values of such features can be extracted from the FFT calculated within each SEN, based on the data acquired in its position. The parameter *NModes* is determined as the exact value of the amount of relevant vibration modes to detect damage on the structure. For defining this exact value, it is necessary to perform a study on the structural properties of the structure by an expert.

LFreqs and *LAmps* parameters are data structures of size *NModes*. Each position of the respective structure must be set with a value regarding to a vibration mode, and the positions of both data structures correspond to the same vibration mode. Each value set to *LFreqs* and *LAmps* is a limit representing the maximum absolute amount of frequency/amplitude deviation from the respective reference value, for a given vibration mode. If a given frequency/amplitude absolute variation is within the respective limit, then we consider that no shift has occurred, so preventing small random disturbances, which do not imply the occurrence of abnormal conditions, from being considered by our algorithm as such. These limits are determined for each SEN based on knowledge and analysis of the structure by an expert. There is no general rule to calculate such values. We consider that the values for these limits can be set/updated at any time by the application expert. The expert may order the dissemination of messages carrying values for *LFreqs* and *LAmps* to the whole network without disturbing the algorithm operation. Each node, on the event of reception of such messages, immediately updates its *LFreqs* and/or *LAmps* data structure with the new value. However, we claim that it may not be enough realistic to assume this threshold as a static value for the entire lifetime of the monitoring system. This is because in most monitoring scenarios, a clear cut deterministic value at which it is possible to assume that there is structural damage or not does not exist. As a drawback, this static limit will lead to wrong results about damage detection, because shifts will be assumed when they do not exist, or vice-versa. In practice, in most monitoring scenarios, the values of frequencies/amplitudes have a time-based dependence on current and past values. Therefore, it is possible to follow approaches based on likelihood level for recalculating and updating limits dynamically. It is possible to consider that these limits could change after the network deployment, being calculated, for instance, automatically within each node. In future works, we can explore approaches existing in the literature for calculating limits dynamically [19], including the use of evolutionary algorithms, statistical methods and fuzzy inference mechanisms. One of such approaches could be implemented with our algorithm, so that each node could locally, based on historical data, adjust its threshold. A function called *update_limits()* could be called right before every assessment of frequency/amplitude shifts. However, this change would cause an impact on node memory, because each node would need to store the values of frequencies/amplitudes obtained in the most recent monitoring cycles.

NSampl is set according to the following criteria. It must: (i) be enough to ensure a good resolution in the power spectrum that will be returned, which implies in better precision in the modal frequencies determination; (ii) be a power of 2, since this is a requirement for the entry of data in the FFT algorithm; (iii) not exceed the sensors storage capacity (Flash memory). *SamplRate* must be: (a) greater than the value of the first modal frequencies of interest so that these are shown in the power spectrum, (b) high enough to ensure accuracy, (c) twice the highest modal frequency of interest, to meet the *Nyquist* criterion. *NeNodeID* is a data structure, defined for each node, whose elements store the unique network addresses (*NodeID*) of all the neighbor nodes. The first position of the *NeNodeID* data structure stores the amount of neighbors of each node.

Each period of the monitoring cycle phase must start at the same time on all SENs, requiring synchronization [18] among them, so that there is meaning in the comparison of their collected data and decisions taken. In our algorithm the synchronization process is carried out in the *first_time_synchronization()* procedure. Such synchronization process has to be maintained and adjusted during the monitoring cycle phase. Our algorithm, by definition, is agnostic to any particular synchronization protocol. The synchronization can be performed by any WSAN synchronization protocol in the literature that meets the requirement of keeping, for long periods of time, the desired degree of exactness (in terms of temporal deviation among the clocks of the nodes) in the synchronization, defined by an expert in the SHM application scenario. The protocol in [8] is an example of existing synchronization protocols tailored to WSAN that meet this requirement.

Also as part of the setup phase, SENs must collect the reference values from its fixed position in the structure and store them in *RefFreqs* and *RefAmps* data structures, during the *ref_values_acquisition()* procedure. A procedure consisting of part of the monitoring cycle (from the beginning of the monitoring cycle to the feature extraction, as mentioned in Section 3.2) is performed once during the setup phase by the SENs, to acquire the reference values. Thus, data structures such as the acceleration samples (*AcSampl*), the power spectrum (*PwrSpec*), Frequencies (*Freqs*) and Amplitudes (*Amps*) are used for the first time in the algorithm to support the reference values acquisition, but they are the same used during the monitoring cycle phase.

The *AcSampl* data structure stores the raw vibration data obtained from the accelerometers of the SENs, and the *PwrSpec* data structure stores the output of the FFT performed over the data in *AcSampl*. The *Freqs* and *Amps* data structures store, respectively, the values of frequencies and amplitudes extracted from the *PwrSpec*. The *Freqs*, *Amps*, *RefFreqs* and *RefAmps* data structures have the same sizes, equal to *NModes*. The same position in each of such data structures stores a value respective to the same vibration mode. The *AcSampl* and *PwrSpec* have the same sizes, equal to *NSampl*.

The data structures cooperative damage coefficient of the node and neighbor CDCs (*NeCDCs*), as well as the current monitoring cycle (t), are initialized during this procedure. The *CDC* is the artifact used for indicating which modal frequencies and amplitudes have changed from the viewpoint of a SEN. The *CDC* is subdivided into two parts: shifts in frequencies ($\Delta\omega$) and shifts in amplitudes (Δa). Both parts

are calculated through significant variations between *RefFreqs* and *Freqs*, and *RefAmps* and *Amps*, respectively. A variation is considered significant if it surpasses the limits stored in *LFreqs* and *LAmps*, respectively. If a variation in a modal frequency/amplitude is not significant, the binary zero value is attributed to the respective position of the $\Delta\omega/\Delta$a parts of the *CDC*, and the binary one value is attributed otherwise. Equations (1) and (2) summarize the calculation of $\Delta\omega$ and Δa

$$\Delta\omega_i = \begin{cases} 1 \text{ if } |\text{Freqs}_i - \text{RefFreqs}_i| > \text{LFreqs}_i \\ 0 \text{ if } |\text{Freqs}_i - \text{RefFreqs}_i| \leq \text{LFreqs}_i, \end{cases} \quad (1)$$

$$\Delta a_i = \begin{cases} 1 \text{ if } |\text{Amps}_i - \text{RefAmps}_i| > \text{LAmps}_i \\ 0 \text{ if } |\text{Amps}_i - \text{RefAmps}_i| \leq \text{LAmps}_i. \end{cases} \quad (2)$$

Since $\Delta\omega$ and Δa have, each, size *NModes*, the *CDC* data structure has a total size of 2xNModes bits. The bits in the first (most significant) half of *CDC* represent the variations on the modal frequencies of the structure ($\Delta\omega$). The bits in the second half of *CDC* represent the variations on the amplitudes of each vibration mode of the structure (Δa). In both halves the most significant bit refers to the first vibration mode of the structure, the second most significant bit refers to the second vibration mode, and so on. The minimum and maximum decimal values achieved by this coefficient depend on *NModes*. For instance, for $NModes = 5$ the *CDC* varies in range 0-1023, being the maximum value achieved when all amplitudes and frequencies shifted).

The data structure *NeCDCs* is used during the collaboration procedure described in Section 3.5. Each of its positions stores the *CDC* values of a neighbor SEN. Therefore, *NeCDCs* has a variable size for each SEN, which is equal to its number of neighbors. Consequently, *NeCDCs* and *NeNodeID* have the same size for the same node, and this size is stored in the first position of *NeNodeID*. The period of the monitoring cycle to be performed is initialized with value zero ($t = 0$).

The *start_monitoring_cycle()* procedure ends with all the nodes entering in sleep mode, and waiting for the beginning of the first period of the monitoring cycle.

3.4 Monitoring Cycle Phase

The pseudo code of the Monitoring cycle phase is shown in Table 1. The SKN is responsible for verifying if a monitoring is requested by an external source using the procedure *mon_requested()*. This procedure returns True if a monitoring was requested or False otherwise. If True was returned, then it also stores in a local variable called *time* the respective moment when such monitoring must start. Also if a monitoring was requested, the SKN is responsible for reviewing time synchronization for the whole network through the *review_time_synchronization()* procedure. The SKN and SENs have their internal clocks synchronized after this procedure, and the SKN can then perform the *transmit_schedule_mon_msg()* procedure to disseminate a message to the whole network, which schedules the starting time of the next period of the monitoring cycle phase.

Upon receiving the message disseminated by the SKN, the monitoring cycle phase starts for the SENs. During this

TABLE 1
Pseudo Code of the Monitoring Cycle Phase

```
       while True:
         if!SENRole && !ATNRole && SKNRole then:
           def time
1:         if mon_requested(time) then:
2:           review_time_synchronization();
3:           transmit_schedule_mon_msg(time);
           end-if
         else-if SENRole && !SKNRole then:
           def time
4:         if schedule_mon_msg_rcvd(time) then:
5:           schedule_mon(time);
           end-if
6:         if start_mon() then:
7:           t+=1
8:           data_collection(AcSampl,NSampl,_
             SamplRate);
9:           perform_FFT(NSampl,AcSampl,PwrSpec);
10:          feature_extraction(PwrSpec,Freqs,Amps);
11:          calc_CDC(CDC,Freqs,Amps,RefFreqs,_
             RefAmps,LFreqs,LAmps);
12:          if variations_due_damage(CDC)>1 then:
13:            start_collaboration();
             end-if
           end-if
         else-if ATNRole && !SKNRole then:
           def ActRules
14:        if actuation_msg_rcv(ActRules) then:
15:          trigger_act(ActRules);
           end-if
         end-if
       end-while
```

phase, the SENs check a variable, through the procedure *schedule_mon_msg_rcvd()*, to find out if a schedule monitoring message was received. If so, the SEN performs the *schedule_mon()* procedure, passing as a parameter the time informed in the message received from the SKN. Such procedure defines the moment when the SEN monitoring must start, setting its internal timer to fire in such a moment. The SENs check through the *start_mon()* procedure when the timer fired. If the timer was fired, then the *start_mon()* procedure returns True, and the SEN increments the counter, for identifying the next period of the monitoring cycle to be performed.

At the beginning of every period of the monitoring cycle, each SEN collects the acceleration data in the time domain at its relative position through the *data_collection()* procedure, passing the *SamplRate* and *NSampl* as parameters. As a result, the *data_collection()* procedure fills the *AcSampl* data structure with the collected data. Next, the SEN performs a FFT on the respective collected data stored in *AcSampl* (of size *NSampl*). The result of the FFT is stored in the *PwrSpec* data structure. Then, the *feature_extraction()* procedure is performed to extract the modal frequencies and amplitudes features in the power spectrum stored in the *PwrSpec* data structure. After extracting and storing the respective values from the power spectrum, each SEN is able to calculate its *CDC* through procedure *calc_CDC()*, using information from *Freqs*, *Amps*, *RefFreqs*, *RefAmps*, *LFreqs* and *LAmps*, as described in Section III.C. Also, a check is performed using the *variations_due_damage()* procedure, passing the recently calculated *CDC* as a parameter. This procedure represents the local decision of the SEN that consists of checking all the

positions (bits) of the *CDC*, and verifying if at least two are different from the binary zero. If such damage is found locally, the SEN must reinforce the existence of the damage starting a collaboration with neighboring SENs, performing the *start_collaboration()* procedure. This procedure is performed only by the SENs that locally decided for damage, and is detailed in Section 3.5. All SENs that did not detect damage are set to sleep until the beginning of the next period of the monitoring cycle phase.

During the monitoring cycle phase, the ATNs are waiting for the reception of a message commanding their actuation. The *actuation_msg_rcvd()* procedure returns True if an actuation message was received, and it sets the local variable *ActRules* with the actuation rules stored in such message. It is important to mention that every SEN was set, via application deployment, with the rules to be used for triggering the actuators. An example of such a rule is "IF consensual decision is True THEN turn one led on", what requires, in case the actuators are represented by leds, the information of which led must be turned on. The procedure *trigger_act()* is performed if such message is received, with the *ActRules* passed as parameters. The procedure *trigger_act()* triggers the physical actuation devices of the node.

3.5 Collaboration Procedure

When performing a decision process within a SEN, a main problem that can arise is that such SEN can be biased due to, for instance, sensor malfunctioning, malicious behavior due to attacks, low battery level, environmental influences or electro-magnetic disturbances [13]. In all such cases, SENs may cause the wrong behavior of the actuators, triggering them when not necessary. It is therefore necessary to mitigate such risk, what can be accomplished by pursuing a consensual decision for a given SEN neighborhood. By reaching a consensus among the SENs in a given neighborhood it is possible to reduce or even eliminate the influence of faulty SENs. To achieve such a consensus, we adopt the Byzantine Algorithm described in [13] as our collaboration procedure. The neighbor SENs participating in the collaboration process must reach a consensus among themselves to whether apply a decision or re-evaluate it, based on the exchanged messages that contain the decisions made by each of the neighbor SEN.

Another important feature to ensure the scalability of our algorithm is that the decisions can be made at various levels. Each level represents a new consensus, for example, a building level would represent the consensus of all floors, and a floor level is a consensus of all nodes in a given floor. To achieve a decision process for supporting decisions at different levels, we have used a consensus algorithm for multilevel decisions (Table 2).

The key idea of our algorithm is as follows. The collaboration procedure comprises a loop that is repeated for each given level (L) of the total number of levels (*NLev*). Within this loop, through procedure *transmit_msg()*, each neighbor SEN informs its decision (stored in variable Consensus) to the other neighbor SENs that pertain to current level L. The SENs that detected damage (*Consensus* = "True") will transmit messages to neighbor SENs, otherwise (*Consensus* = "False") no message will be transmitted. When a SEN does not receive a message from a neighbor, it assumes that such decision is "False". After, the SEN waits on its *NeCDCs* is completely set with new data from the current level, or a timeout expires. These checks are performed by procedure *NeCDCs_fully_refreshed()*. During this wait, the *msg_rcvd()* procedure checks if a message from another SEN that detected damage arrives. If such message arrives, the respective decision value is stored in the *NeCDCs*, by the *fill()* procedure. After this wait finishes, procedure *level_consensus()* takes the majority of the decisions (among the local and the received decisions), and uses that as the final decision for its level. The local variable Consensus has, as initial value, the current *CDC* value of the node. For the second level on, the *Consensus* variable will store Boolean values representing the consensual decisions taken at prior levels. Finally, the current level counter L is increased. After the consensus was reached at all levels, if the result of the whole multilevel consensus is "True", then the node decides if it must send messages to the SKN and ATN. As proposed in [13], only the SEN with the smallest *NodeID* in the neighborhood of the highest level sends messages to the SKN and ATN, to reduce the redundancy of messages received by these nodes. So, the procedure *Min()* finds the smallest *NodeID* among the nodes in the highest level. This SEN triggers the ATN, through the *transmit_actuation_msg()*, and sends a message to the SKN reporting the existence of the damage, through the *transmit_sink_report_msg()*.

According to the properties of the original Byzantine Generals Problem [13] our approach is known to work successfully (ensuring a consensus around the correct decision at each level) only when the number of participating SENs 'N' is at least '3M + 1', where 'M' is the number of participating SENs that send an incorrect decision. A decision is considered incorrect when it assumes damage and the damage does not exist, or when it does not assume damage when it exists. Moreover, *NLev* is a configurable parameter in our algorithm, to be set by an application expert, according to the type and size of the structure, application requirements and the goal of monitoring (eliminating incorrect information or evaluating the damage extension). However, the energy cost to perform our multilevel consensus algorithm increases when the number of nodes in a given level increases.

TABLE 2
Pseudo Code of the Collaboration Procedure

```
      while (L<NLev):
1:      transmit_msg(Consensus,NeNodeIDs,L);
2:      while !NeCDCs_fully_refreshed(L):
3:        def rcvd
          if msg_rcvd(rcvd) then:
4:          fill(NeCDCs,L,rcvd);
5:        end-if
        end-while
        Consensus=lev_consensus(Consensus,NeCDCs,L)
6:      L+=1;
7:    end-while
      if Consensus then:
8:      if Min(NeNodeIDs,NodeID,L-1)==NodeID then:
9:        transmit_actuation_msg();
10:       transmit_sink_report_msg();
11:     end-if
      end-if
```

4 Use Case and Implementation

To evaluate our proposal, we developed a use case in the domain of SHM for smart buildings. The goal of the use case is to implement a system based on our proposed algorithm for monitoring the health of smart city buildings. Among the requirements of a smart building, it must ensure the safety of the people living in it, especially considering seismic actions. Frequently, buildings are closed for months to undergo a detailed inspection, in case of relevant seismic events, for ensuring the safety of the people. Such approach is inefficient.

For illustrating this scenario, consider a smart building composed of several floors, with a WSAN deployed on each of the floors. Within each floor, sensors are deployed in key locations where the shock response of the structure is the highest. These locations are defined by an expert in the structure and can be, for instance, joints of floor plates and/or joints of plate beams with columns. In the same edge of each floor, consider the deployment of respective sink nodes of each WSAN, each of which attached to a local standard PC. These sink nodes serve as gateways and connect all the floors through the building local area network (LAN). The LAN connects to the Internet by using existing wireline or wireless broadband internet connections (ADSL, VDSL, Satellite, WiMax) in the building.

In this building, our collaboration algorithm is configured with two consensus levels: floor level and building level. Each level represents a new consensus, i.e. a building level represents the consensus of all floors about the structural integrity of the building, and a floor level is a consensus of all nodes in a given floor about the integrity of the floor. The sink nodes of each level can be assigned as the node with the smallest NodeID of a given floor, and so they will partake in the consensus of the whole building level. It is worth mentioning that each WSAN must be deployed so that its extension is not excessive, to avoid generating long delays in the final consensus on damage. If a large-scale deployment is foreseen within one level, then it must be divided into smaller consensus levels, to avoid obstacles in the monitoring area. Therefore, routing impairment problems such as shadowing (by big concrete slabs and pillars), that may render neighbor nodes blind to each other, will be avoided. Regarding actuators, each floor of the smart building is outfitted with stoplight-style signs that automatically announce structural soundness.

The WSAN deployed in each floor consists of MICAz motes [10], whose radio supports 2.40-2.48 GHz band and 250 kbps data rate. Each MICAz mote includes the board containing the processor, radio, memory and batteries. The common energy source of MICAz motes consists of two AA batteries, which provide up to 16 kJ of energy, as estimated in [4]. The motes are programmed in NesC language, under the TinyOS development environment [9], version 2.1. For dealing with communication at the physical layer, we adopted the standard 802.15.4 protocol implementation provided by TinyOS. For dealing with the transmission of point-to-point messages at link-level, we adopted the Active Message protocol [9] implementation, also provided by TinyOS. The maximum payload size allowed for the messages exchanged, limited by TinyOS 2.1, is 28 bytes. The implementation of our algorithm itself consists of a single program running inside each mote. Our algorithm runs over the BMAC protocol [23] in the MAC layer. Using pure physical layer information, the BMAC is capable of detecting neighboring wake-up transmissions on the channel. Consequently, BMAC allows nodes to get into deep-sleep state until wake-up transmissions are detected. Our algorithm and the BMAC work independently. When a node in our algorithm becomes idle, it does not make transmissions anymore. Therefore, it is a matter of time until the BMAC protocol returns nearby nodes to deep-sleep. Besides, since the monitoring cycle starts when the SKN disseminates a message for the whole network, nodes in deep sleep will be woken up by the SKN transmission during one of the periodic receive checks of BMAC. Therefore, the use of BMAC contributes for saving nodes' energy during the idle time of our algorithm.

In our implementation, the SamplRate and NSampl were set to 1 kHz and 512 samples, respectively. We also set the feature extraction method to extract the first five modal frequencies of the frequency spectrum (NModes = 5). Other works, such as [3] also used similar amounts of modal frequencies in their experiments. This amount is sufficient to perform the analysis of the health of a civil structure in the range of 500 Hz in the spectrum (half the SamplRate, according to the Nyquist criterion [5]).

The resource constraints of the SENs imply in the implementation of a less precise feature extraction method than conventional methods, such as the PCF from [4], since the chosen feature extraction method must run completely within the SEN. Thereafter, to identify the frequency peaks, we used the method proposed by [5], which takes as input the desired frequency ranges and calculates the frequency value of the highest peak in that range, returning such frequency and its respective amplitude values. However, the lack of precision of this method can be balanced by increasing the number of SENs in the monitoring area for data redundancy.

We programmed the ATN to toggle a LED for simulating an actuation function when receiving an actuation message. In a real environment, this actuation would consist in setting values of stoplight-style signs, or communicating to the building operation center/supervisors.

Finally, our prototype uses four types of messages to perform communication among motes: schedule monitoring message, CDC message, actuation message, and sink report message. The schedule monitoring message has two bytes in its payload to transport the value of time representing the moment when the monitoring should start (this amount of bytes is the default amount used to represent an interval in TinyOS 2.1). The CDC message has two bytes in its payload, since the CDC data structure has size 2xNModes bits (10 bits), but in TinyOS 2.1 the message structure must be represented by a round number of bytes (two bytes can represent 10 bits). The actuation message has a null payload (no configuration is required to perform the action programmed in our prototype). The sink report message has a 4 bytes payload, two for storing the CDC of the current node, and two for the NodeID of the current node. It is important to mention that each message type requires an extra 8-byte header (in TinyOS 2.1).

TABLE 3
Metrics and Acronyms

Metric	Question
True positives/True negatives	Q1, Q2
WSAN lifetime (WL)	Q3, Q5
Amount of bytes rx (ABR) / tx (ABT)	Q4, Q5
Packet loss rate (PL)	
Percentage of free bytes in RAM (BR) / flash (BF)	Q5
System response time	Q6

5 Evaluation Methodology

5.1 Evaluation Metrics

Considering the objectives of this work and following the Goal Question Metric (GQM) methodology [12], three goals were defined. Goal G1 is to analyze our algorithm for the purpose of evaluating its accuracy, in terms of successes in the detection of damage in civil structures. Goal G2 is to analyze our algorithm for the purpose of evaluating its overhead in terms of the consumption of computing, communication and energy resources of the WSAN. Goal G3 is to analyze our algorithm for the purpose of evaluating its response time in terms of the duration of a period of the monitoring cycle phase. These goals were refined in six questions. Questions Q1 and Q2 relate to goal G1, questions Q3, Q4 and Q5 relate to goal G2, and question Q6 relates to goal G3. Q1: How does the use of cooperative information fusion contribute to the accuracy of our algorithm? Q2: How does the use of the collaboration procedure contribute to the accuracy of our algorithm? Q3: How long can the WSAN last when running our algorithm, operating system and protocols? Q4: Does the communication overhead of our algorithm impact the correct operation of our proposed collaboration procedure? Q5: How does the multilevel decision procedure (collaboration procedure in Section 3.5) impact the overhead of our algorithm? Q6: How fast can our algorithm respond (triggering actuators), since the beginning of a period of the monitoring cycle phase?

Finally, metrics are defined (Table 3) to support the answers to the questions. Regarding Q1 and Q2, we used the metrics: true positives (TP), true negatives (TN). The TP metric counts the number of situations in which our algorithm detects damage that actually occurs during a period of the monitoring cycle phase. The TN counts the number of situations in which our algorithm does not detect damage during a period of the monitoring cycle phase, and it in fact did not occur. The quantity TP+TN is referred in this work as the amount of trues, which denotes the number of situations in which our algorithm makes correct damage detection or not. For Q3 and Q5 we defined the WSAN lifetime (WL) metric as the time elapsed from the beginning of the algorithm execution until the moment in which the WSAN is not able to achieve its main goal. In our work, the main goal is to perform the damage detection and actuation. Because of the collaboration procedure, at least one ATN and two or more SENs are required for achieving the main goal. The WL is calculated based on the energy consumption value measured for a mote during a period of the monitoring cycle, and the initial amount of energy of this mote.

Regarding Q4 and Q5, we defined three metrics: the amount of bytes received by each SEN (ABR), the amount of bytes transmitted by each SEN (ABT) and the packet loss rate (PL). The ABT is calculated as the sum of the bytes transmitted by a single SEN, in average, during one period of the monitoring cycle phase. The ABR is calculated similarly, but for the sum of bytes received by a single SEN. The PL is calculated as the ratio between the number of packets lost by a receiver SEN (packets which did not reach the correct destination of this SEN) and the total amount of packets transmitted to this same SEN, in average, during one period of the monitoring cycle phase. Regarding Q5, we defined two metrics: percentage of free bytes in RAM (BR) and percentage of free bytes in flash memory (BF). The BR metric results from the subtraction of the RAM occupation of the implemented program from the total amount of RAM in the MICAz platform. The BF is defined similarly, but for the Flash memory. Finally, for Q6 we defined the system response time (SRT) metric as the time elapsed between the beginning of a period of the monitoring cycle phase, during which damage detection is performed, and the reception of a message by an ATN, pointing that a control action should be performed.

5.2 Experimental Environment

We used two experimental environments: one simulated and one with real motes. The environment with real motes has the goal of validating the results obtained in simulations. We considered both environments as indoor environments. In each experiment, we performed 30 repetitions of the period of the monitoring cycle phase of our algorithm, what provided a reasonable confidence interval of 95 percent for the results.

A desktop computer equipped with an Intel Core 2 Duo 2.80 GHz processor and 4 GB of RAM was used to run the simulations. The simulations were performed with version 2.6 of the Avrora simulator [11], which is an open source simulator for WSANs. AvroraZ extension [11] was used to analyze the energy consumption and communication for the MICAz platform. The energy model used by Avrora is called "accurate prediction of power consumption" (AEON) [11] and is the energy model that represents more precisely the processing cycles of MICAz motes.

The environment with real motes is a simplification of the environment described in Section IV that considers a deployment on a single floor (one level) of a building. We assembled this controlled environment within the wireless networks laboratory (LabNet), at Universidade Federal do Rio de Janeiro (UFRJ), where the nodes were kept stationary and disposed on the ground. In this environment, every message transmitted on the network was read by the SKN, where a software program written in Java, connected to the computer serial interface, recorded messages for further analysis. The hardware of the SKN consisted on a MICAz mote connected via USB cable to a desktop computer equipped with an Intel Core 2 Duo 2.80 GHz processor and 4 GB of RAM.

In the environment of the experiments: (i) the acceleration data collected by each SEN were simulated (as in [5]); (ii) we used a methodology to simulate damage that gradually inserts damage in the structure, so that the collected acceleration data is properly changed for reflecting the

damage; (iii) the values of modal amplitudes and frequencies used as input for this simulation were extracted from a plate structure in [6]; and (iv) we assumed that only natural excitation was sufficient to excite the modes of vibration of the structure.

In this work, we use a flat and static WSAN topology. The NeNodeIDs data structures are filled with fixed values before the beginning of our algorithm operation, so that the SENs, ATNs and SKN are always the same nodes (no rotation of roles). Our algorithm also assumes that the WSAN nodes have their clocks synchronized. However, this synchronization requirement is only essential when considering the collection of real data from the structure, using real accelerometers. Once collected acceleration data were simulated, as described in Section 5.3, implementing data synchronization is not necessary. The delay with which each sensor node begins its period of the monitoring cycle phase is minimal. It is only a result of the time between consecutive transmissions of two messages requesting data collection. In the prototype implemented in TinyOS, this time is approximately 10 ms.

Studies found in the WSAN literature [1] make use of the multi-hop transmission scheme in their experiments due to the low radio range of the motes and the great distances between the motes and the sink node. However, in the performed experiments, the distances between the motes are small, as also in related works [4]. Therefore, all motes are within the same coverage area of each other, eliminating the need of using a multi-hop transmission scheme. In the case of adopting structures with dimensions larger than those used in the present work, the multi-hop transmission scheme should be reconsidered for possibly improving the network scalability.

6 Experimental Results

This section describes the results of experiments E1 and E2, which relate to the goal G1 (Section 4.1) and E3, E4, E5 and E6, which relate to G2 and G3 (Section 4.2).

6.1 Set of Experiments A: Accuracy

Experiment E1 was conducted to answer Q1. This experiment consisted in evaluating our algorithm with and without the use of cooperative information fusion. Therefore, three versions of our decision mechanism were considered. Version VFA uses both modal frequencies and amplitudes shifts for composing the CDC: it is our implementation presented in Section 4. Version VF uses only modal frequencies shifts and version VA uses only modal amplitudes shifts for composing the CDC. In addition, for performing this experiment, we considered a deployment scenario similar to the one used in Santos et al. [5], and the SENs were considered installed over a plate structure as the one studied in Reddy and Swarnamani [6]. We replaced the cluster-head nodes in the original WSAN topology from Santos et al. [5] by SENs in our work, since we consider a flat network topology. In this scenario, our multilevel decision procedure was configured so that all the SENs in the WSAN pertain to the same and only existing decision level (NLev = 1).

In our scenario, we had one case of healthy structure, and four cases of damage progression. Each damage case represents a progression of the damage in the scenario, thus the damage becomes more evident in the later periods of the monitoring cycle phase. During the experiments, 150 repetitions were performed, from which 30 repetitions were performed for the undamaged case and 120 for the four damage cases (30 for each damage case), and no damage and undamaged cases were mixed during the same 30 repetitions. Because we had one case of healthy structure, and four cases of damage progression, our mix of cases to calculate the TP and TN metrics was 20 percent undamaged and 80 percent damaged. Using the available mix of cases of damage, the ideal result for a detection mechanism would be TP = 80 percent (four of five damage cases) and TN = 20 percent (one of five damage cases).

TABLE 4
Results of the Experiments E1 and E2

Version	TP (%)	TN (%)	Amount of Trues (%)
VFA	77.33	16.50	93.83
VF	0.00	20.00	20.00
VA	76.00	16.50	92.50
VFAw	77.33	10.83	88.17
VFw	0.00	16.83	16.83
VAw	76.00	10.00	86.00

Experiments were performed in both simulated and real motes environments, showing the same results. Table 4 shows results for TP and TN metrics for experiment E1 for versions VFA, VF and VA of our algorithm. One of the main reasons to explain the results of the TP metric column is that our proposed algorithm depends on at least two shifts (two bits in the CDC with value = 1) caused by the actual presence of damage for deciding about damage detection (TP), or it will assume no damage detection. Considering the cases where only two shifts occur, when one of such shifts occurs in one of the monitored natural frequencies, and the other shift occurs in one of the monitored amplitudes, Version VFA will assume damage detection, and such cases will be accounted as TP (TP = 77,33 percent). However, versions VF and VA will perceive only one shift, not resulting in TP. Still, version VF showed for TP metric, a value much lower than version VA. This happened because version VF assesses only the shifts caused by the variation of natural frequencies, and in the structure reported in [6] the natural frequencies are not sufficiently sensitive to damage. However, in the structure reported in [6], the amplitudes are sufficiently sensitive to damage and, therefore, version VF is able to detect damage in most of the cases where it is actually present (TP = 76.00 percent). Values of CDC obtained by nodes during the experiments that fit in each of the aforementioned cases are shown in Fig. 1. For explaining the results of the TN column in Table 4, the same reason (related to the use of cooperative information fusion) applies. Since version VF (TN = 20 percent) is not sufficiently sensitive to damage, it never assumes damage detection in any case, what is convenient when damage is actually not present in the structure but severely inconvenient when damage is actually present. Versions VFA and VA showed the same values of TN, because both versions assess the shifts in amplitude, which are sufficiently sensitive to damage. From the results in Table 4 we can see that

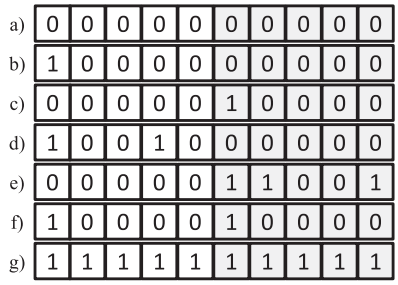


Fig. 1. Examples of CDC values with: a) no shift (undamaged), b) one frequency shift (undamaged), c) one amplitude shift (undamaged), d) two frequency shifts (damaged), e) three amplitude shifts (damaged), f) one frequency and one amplitude shift (damaged), g) All frequency and amplitude shifts (damaged).

our algorithm (version VFA) performed well in terms of reaching values of TP and TN close to the ideal situation (amount of trues = 93.83 percent).

Responding to Q1, the use of cooperative information fusion contributes to ensure higher levels of accuracy, since assessing both frequencies and amplitudes simultaneously allowed a better performance of our algorithm full implementation (VFA). Our algorithm showed a higher amount of trues than versions VF and VA, which assessed only one feature at a time.

Experiment E2 was conducted to answer Q2. This experiment consisted in evaluating the algorithm with and without the use of our collaboration procedure. Therefore, we considered six versions of our algorithm: versions VFA, VF and VA, the same used in experiment E1, which have the collaboration mechanism implemented, and versions VFAw, VFw and VAw that are based on versions VFA, VF and VA, but without the collaboration mechanism implemented. In versions VFAw, VFw and VAw, every SEN forwards its decision to the ATN based only on its local view (without consensus). We considered the same scenario used in Experiment E1. Table 4 shows the results for experiment E2.

Comparing the results from experiments E1 and E2, we observe that the absence of the collaboration mechanism had a great negative impact on the amount of trues (reducing it). Therefore, versions VFAw, VFw and VAw performed worse than versions VFA, VF and VA. In addition, since changes in the rates of TP were not perceived between each respective version, such decrease for Trues was due to a decrease for TN in all versions without the collaboration mechanism. Such a decrease is explained by the fact that, in the absence of the collaboration mechanism, some SENs which perceived changes in their frequencies and/or amplitudes when the structure was actually undamaged (SENs with an undesired behavior), started reporting damage. This fact shows the importance of our collaboration mechanism. Responding to Q2, the use of the collaboration procedure contributes for ensuring higher levels of accuracy, increasing the TN rate.

We conclude that goal G1 was achieved. It is also important to mention that the calculations performed within each SEN in version VFw are, in fact, close to the calculations used for damage detection in Sensor-SHM, our previous work [5]. The results in terms of accuracy obtained in our current work support the following conclusions. Our (new) proposed algorithm is suited for the same structures that Sensor-SHM is, however, due to the use of cooperative information fusion, it is also capable of detecting damage when the natural frequencies of the structure are not sufficiently sensitive, or insensitive to damage, but their amplitudes are sufficiently sensitive to damage. Therefore, our algorithm is more appropriate for detecting damage in more diverse structures than Sensor-SHM.

6.2 Set of Experiments B: Overhead

This section presents the results of experiments E3, E4, E5 and E6, for respectively answering Q3, Q4, Q5 and Q6. As a point in common, experiments E3 and E4 used the same scenarios when varying the number of nodes to assess the scalability of our algorithm when our multilevel decision procedure was configured so that NLev = 1. Because the scalability is directly related to the increase in the network size, we chose to change the monitoring area and the number of nodes, keeping a constant node density, following other works such as [14] that used this approach successfully. All scenarios have a square monitoring area comprising SENs under a flat WSAN topology. The node density was set to 1 SEN/m^2, so every SEN is equally spaced from its immediate SENs by 1m in the y-axis and 1m in the x axis, what is repeated in every variation pattern. The first pattern had the minimum amount of SENs required to perform our collaboration procedure, i.e. two SEN in each side (called pattern P2), which results in a scenario with four SENs. We increased one by one the amount of SENs in each side of the monitoring area, for generating patterns P3, P4, P5, ... , P11. In all patterns, all SENs are within the radio range of each other and so, every SEN, is considered a neighbor of every other SEN in the scenario. For instance, the variation patterns P3, P5, P7, P9 and P11 have 9, 25, 49, 81 and 121 SENs, respectively. Therefore, each SEN has 8, 24, 48, 80 and 120 neighbors, respectively in each pattern. Each pattern also has a SKN at the origin of the Cartesian plane, and an ATN at 1 m from the SKN in the y-axis.

Experiment E3 was conducted to answer Q3. This experiment consisted in evaluating the WSAN Lifetime (WL metric) while varying the number of SENs in the scenario through patterns P2 to P11. In E3, LFreqs and LAmps were set to zero, so that it is assumed that in all periods of the monitoring cycle phase, sites with the presence of damage were found. This is the most demanding scenario in terms of WSAN resource consumption, since the SENs must collaborate for reaching a consensus, spending more communication energy.

Considering that the energy model of the Avrora simulator was sufficiently validated, we measured the power consumption from the MICAz motes only through simulations. We performed one monitoring cycle phase for assessing the average energy consumption of each node in each scenario. Thereafter, we calculated the WL metric, considering one monitoring cycle phase being performed per hour, while during the remaining time all the WSAN nodes are in sleep mode. For patterns P2 and P11 the WL was, respectively, 475 days $\pm$ 1 day and 468 days $\pm$ 2 days (Fig. 2). Such decrease in WSAN Lifetime when increasing the number of SENs is explained by the increase in the communication

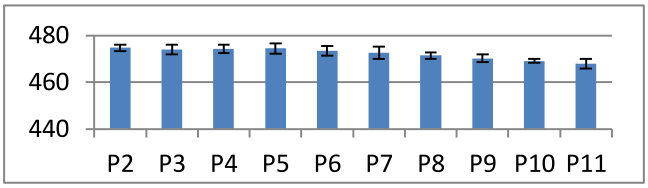


Fig. 2. Results of the WL metric (days) for E3.

overhead during the experiment, as well as the increase in the radio range of SENs, for communication among the neighbors in one hop distance. Responding to Q3, the WSAN can last as long as 468 days, in the worst case, running our algorithm, operating system and protocols in different scenarios (using 121 or less SENs).

Experiment E4 was conducted to answer Q4. This experiment consisted in evaluating ABT, ABR and PL metrics while varying the number of SENs in the scenario. The same criteria used in E4 for varying the number of SENs were used. Results are shown for ABR and PL in Fig. 3. The value of ABT is constant for all scenarios (64 bytes), because every SEN in our algorithm always transmits the same amount of messages during a period of the monitoring cycle phase, regardless of the amount of neighbors in the scenario. However, each message transmitted by a SEN is accounted as received by every other neighbor, and more transmissions occur as the amount of SENs in the scenario increases. Therefore, as shown in Fig. 3a, ABR increases when the amount of SENs increases. In addition, as shown in Fig. 3b, the increase in the ABT caused more packet losses (only due to more collisions of messages, since no noise was present in our simulated environment) in scenarios with more SENs.

We can deduce that, as the PL increases, our collaboration algorithm becomes less capable of reaching a consensus around the correct decision. Such deduction is possible when analyzing the assumption (taken from the original Byzantine Generals Problem) that the number of participating SENs 'N' is at least '3M + 1', where 'M' is the number of participating SENs that send an incorrect decision. In other words, the number of participating SENs that send an incorrect decision must be, at most, 1/3 of the total number of participating SENs, for the algorithm to reach a consensus. The packet losses result directly in the occurrence of false negatives because when a packet is lost the destination SEN understands (incorrectly) that the respective decision was 'False', when it was 'True' in fact. Therefore, considering that in pattern P11 the PL surpasses 1/3, we conclude that our collaboration procedure does not perform properly in a scenario with 121 SENs (P11). Since the PL for scenario P10 is lower than 1/3, for the amount of SENs in pattern P10 our collaboration procedure is still capable of reaching a correct consensus. Responding to Q4, the communication overhead of our algorithm might affect the correct operation of our proposed collaboration procedure (packet losses impact the capacity of the byzantine algorithm to reach a consensus) for scenarios with more than 100 nodes. Scenario P10, with 100 nodes, is the closest to the limit of the allowed amount of SENs with incorrect decisions for reaching a correct consensus through the Byzantine Algorithm in one same domain in the lowest and only level (NLev = 1).

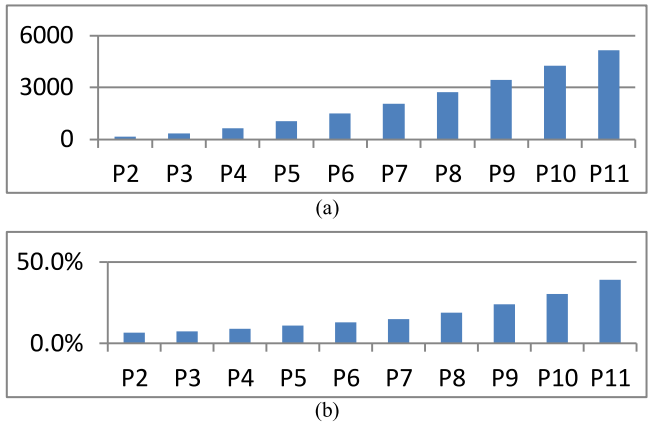


Fig. 3. (a) Results of ABR metric (Bytes) for E4. (b) Results of PL (percent) metric for E4.

Experiment E5 was conducted to answer Q5. It consisted in assessing metrics BF, BR, WL, ABT, ABR and PL when configuring the multilevel decision procedure with NLev = 2 in scenario P10, and comparing these results with the ones obtained for a base scenario (the original P1, with NLev = 1). When configuring NLev = 2 in scenario P10, we considered the existence of four floors in the building (each comprising 25 SENs concentrated at the four corners of the floor, as if the original P10 scenario were divided by a cross) at the first and lowest level of consensus. The second and highest level of consensus comprised the SKNs of each floor located at the center of their respective floors. The results obtained in E7 were normalized for comparison with the base scenario. The results of the base scenario are considered as 100 percent (having the same absolute values obtained for scenario P10 in experiments E3 and E4, i.e. P10 and NLev = 1).

Our multilevel decision procedure affected the overhead of our algorithm as follows. For NLev = 2 we observed a reduction of BF and BR in relation to NLev = 1 (of 0.7 and 6.3 percentage points), due to, respectively, an increase in the size of the program for implementing the multilevel decision procedure, and an increase in the size of the NeNodeIDs, for storing the information of the neighborhood at the second level. The impact over WL was negligible, in the order of 2 hours, so that it did not affect the amount of days achieved for the WL metric in the base scenario (468 days ± 2 days). This fact is explained by the additional amount of messages transmitted to perform the multilevel decision procedure. Such amount is considerably smaller when compared to the amount of messages used in the other procedures of the algorithm. ABT increased only by 0.3 percent, while ABR increased by 3.7 percent. Finally, PL was reduced by 3.6 percent, because the small amount of new messages transmitted for performing the consensus at the second level were not lost, since at this level the communication was performed only among four SENs, avoiding collision of messages. Therefore, the new amount of messages transmitted was accounted as successful transmissions.

Responding to Q5, the multilevel decision procedure (collaboration procedure in Section 3.5) has a tolerable impact in the overhead of our algorithm. Finally, we conclude that G2 was achieved, since our algorithm was analyzed for the purpose of evaluating its overhead in terms of

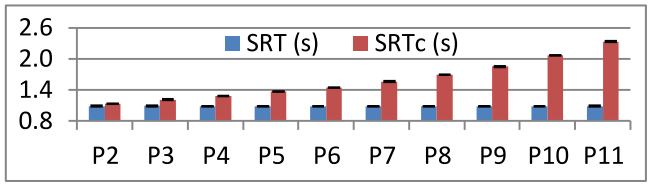


Fig. 4. Results of the SRT metric for experiment E6.

the consumption of computing, communication and energy resources of the WSAN.

Experiment E6 was conducted to answer Q6. It consisted of evaluating the SRT metric while varying the number of SENs. We used the same scenario and variations of experiment E3. In addition, it was assumed that in all periods of the monitoring cycle phase, sites with the presence of damage were found, the most demanding scenario in terms of WSAN resource consumption. Fig. 4 shows results for SRT. For patterns P2 and P11 the SRT was, respectively, 1.091 ± 0.007 and 1.087 ± 0.012 s. Because all the values of the SRT metric are within the confidence interval of each other, we can conclude that there was no significant change in SRT when varying the number of nodes in the WSAN. This constant value is explained by two factors: (i) the 1-hop distance from each node to the SKN and ATN, i.e. all the WSAN nodes are within the radio range of each other, avoiding the time spent with routing; and (ii) the small amount of bytes transmitted in each message by each SEN.

To evaluate the gain of a decentralized approach (our algorithm) compared to a centralized approach, we conducted experiments to measure SRT also for a centralized version of our algorithm, in which SENs collect data, extract the frequency and amplitude and transmit them directly to the SKN. The decentralized version achieves a shorter SRT. The SRT achieved by the centralized approach was 4 and 115 percent higher than the SRT achieved by the decentralized approach, for scenarios P2 and P11, respectively. This difference is related to the fact that the SKN needs to handle all the received messages from all the SENs and perform the process of damage detection for each SEN. There is also an additional cost of time for the SKN to handle all the received messages, because every message received by the radio must be retransmitted via serial communication for a central computer, where the data is stored in a database to be further processed. In the decentralized approach, the SENs perform the prediction process in parallel, and only wait for collaborating over the result obtained by each node. Faster actions can be very important to increase the safety of the operation and the operational performance of the monitored structure. Responding to Q6, the WSAN can respond promptly (1.091 s), triggering actuators, in different scenarios, with the number of nodes smaller than 121. Finally, we conclude that goal G3 was achieved.

6.3 Impact of Non-Assessed Factors

In this section, we discuss the impact of factors that were not assessed in the described experiments on our algorithm's performance. Relevant parameters that deserve discussion are network density and faulty nodes.

Regarding the network density, it potentially affects the performance of our algorithm as follows. In the current setting for the experiments, the density was 1 SEN/m^2. Let us consider the increase of such value. This variation will reduce the sensing area dedicated to each SEN and, consequently, more SENs will transmit data at the same time in the same area. We can foresee two effects caused by such increase in density. One effect regards an increase in the algorithm accuracy. This positive impact happens because during the execution of the consensus algorithm in one level, every node will have more neighbors in its range for collaborating and the consensus will be reached among more nodes. As a negative effect, an increase in node density will increase the PL metric, so more collisions of packets will occur. However, given that nodes were reasonably spaced in the floor of a building during their deployment, the first impact may increase before the second impact. However, reaching higher values of node density will cause the second impact to be stronger. As a conclusion, this situation suggests that an optimum point in node density exists, and it calls for further investigation in future work. Therefore, when performing a deployment in a building, the expert in the structure must consider the node density in one same floor as a potential way of increasing the accuracy of the algorithm, but this should be used wisely.

Regarding the faulty nodes, we need to consider the effect of different types of faults since they affect the algorithm differently. We consider a node as faulty due to factors such as: (i) measurement errors due to sensor malfunctioning, (ii) malicious behavior due to attacks, (iii) low battery level, (iv) environmental influences or (v) electromagnetic disturbances. In our experiments, we considered only packet losses, a fault that could occur because of some aforementioned factors as, for instance, electro-magnetic disturbances. Such losses result directly in the occurrence of false negatives because when a packet is lost the destination SEN understands (incorrectly) that the respective decision was 'False'. However, some faults, such as errors due to sensor malfunctioning may also transform the local decision of a node into true when it should be false, thus increasing false positives. Possible solutions for dealing with all those factors are as follows. Increasing node density in the same area; eliminating the environmental influence by using filters during data collection and processing; repeating the broadcast of decisions several times to increase the probability of delivering messages to all nodes.

It is important to mention that we are discussing the loss of messages carrying decisions as the most important case. However, we can consider the case in which synchronization messages are lost or delayed leading to measurements being taken not at the same time. In this case, some nodes will not partake in the monitoring cycle. Their decision messages will be consequently delayed or never be sent. Therefore, this case may increase false positives, as previously discussed. The global agreement will not take more or less time to be reached, because of how our consensus algorithm was designed: based on a timeout, so that it has a fixed maximum time for reaching the consensus. Therefore, our consensus algorithm will not wait for delayed nodes or messages. Such messages will be dropped and the consensus will happen without such decisions.

It is also worth mentioning that the monitoring cycles of our algorithm should never overlap. The system response

time (duration of a monitoring cycle) is in the order of seconds, as evaluated in the experiments. Therefore, the application expert should schedule the start of monitoring cycles with intervals higher than this response time, i.e. at every minute, hour or higher. Using this rule, and because the consensus has a fixed maximum time for being achieved, messages for scheduling the next monitoring cycle will never arrive while the system did not reach the consensus of the previous monitoring cycle.

7 Conclusion

This work presented a decentralized algorithm for detecting damage in structures by using a WSAN. Overall, the use of the information fusion techniques presented in our work reduced the transmissions of data messages. Moreover, results showed that it is possible to detect a single position of damage using our algorithm in a case where damage on a structure makes its frequencies and amplitudes shift. The information fusion techniques helped to reduce data and, consequently, allowed a faster and less energy consumptive information exchange among WSAN nodes. False positive and false negative avoidance are some of the main reasons to stimulate such collaboration mechanisms among the nodes in our algorithm. In future works, we intend to explore the cooperative information fusion along with data from different kinds of sensors, such as strain gauges. Another future direction is to investigate the existence of an optimal point where data reduction and information fusion should be applied in our assessment, relating in a trade-off the reduction in overhead and the loss of accuracy.

Acknowledgments

This work was conducted during a scholarship supported by CAPES at Universidade Federal do Rio de Janeiro. This work was supported in part by CNPq under grants 304941/2012-3, 473851/2012-1, 477223/2012-5, and 307378/2014-4; by Inmetro/Pronametro; and by FAPERJ under grant JCNE E-26/102.961/2012. Albert Zomaya's work is supported by the Australian Research Council Discovery Grant DP130104591.

References

[1] I. Akyildiz and I. Kasimoglu, "Wireless sensor and actor networks: Research challenges," *Ad Hoc Netw.*, vol. 2, no. 4, pp. 351–367, Oct. 2004.

[2] E. Nakamura, A. Loureiro, and A. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications," *ACM Comput. Surv.*, vol. 39, no. 3, pp. 9/1–9/55, Sep. 2007.

[3] Y. Gao, B. Spencer Jr., and M. Ruiz-Sandoval, "Distributed computing strategy for structural health monitoring," *J. Struct. Control Health Monit.*, vol. 13, no. 1, pp. 488–507, Jan./Feb. 2006.

[4] G. Hackmann, F. Sun, N. Castaneda, C. Lu, and S. Dyke, "A holistic approach to decentralized structural damage localization using wireless sensor networks," *Comput. Commun.*, vol. 36, no. 1, pp. 29–41, Dec. 2012.

[5] I. Santos, L. Pirmez, E. Lemos, F. Delicato, L. Pinto, J. Souza, and A. Zomaya, "A localized algorithm for Structural Health Monitoring using wireless sensor networks," *Inf. Fusion*, vol. 15, pp. 114–129, Jan. 2014.

[6] D. Reddy and S. Swarnamani, "Application of the FRF curvature energy damage detection method to plate like structures," *World J. Model. Simul.*, vol. 8, no. 2, pp. 147–153, May 2012.

[7] S. Meguerdichian, S. Slijepcevic, V. Karayan, and M. Potkonjak, "Localized algorithms in wireless ad-hoc networks: Location discovery and sensor exposure," in *Proc. 2nd ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2001, pp. 106–116.

[8] G. Huang, A. Zomaya, F. Delicato, and P. Pires, "Long term and large scale time synchronization in wireless sensor networks," *Comput. Commun.*, vol. 37, pp. 77–91, Jan. 2014.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *Proc. 9th Int. Conf. Architectural Support Program. Languages Oper. Syst.*, 2000, pp. 93–104.

[10] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, MICAz DataSheet. MEMSIC, USA [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf, 2010.

[11] R. Alberola and D. Pesch, "AvroraZ: Extending Avrora with an IEEE 802.15.4 compliant radio chip model," in *Proc. 11th ACM/IEEE Int. Symp. Model., Anal. Simul. Wireless Mobile Syst.*, 2008, pp. 43–50.

[12] V. Basili, "Software modeling and measurement: The goal/question/metric paradigm," University of Maryland, College Park, MD, USA, Tech. Rep. CS-TR-2956, 1992.

[13] R. Klempous, J. Nikodem, L. Radosz, and N. Raus, "Byzantine algorithms in wireless sensors network," in *Proc. Int. Conf. Inf. Autom.*, 2006, pp. 319–324.

[14] C. Long, Y. Li, and Y. Li, "A multi-hop routing scheme based on different spatial density of nodes in WSNs," *J. Theoretical Appl. Inf. Technol.*, vol. 46, no. 1, pp. 195–200, Dec. 2012.

[15] R. Giffinger, C. Fertner, H. Kramar, and R. Kalasek, "Smart cities: Ranking of European medium-sized cities," Centre of Regional Science, Vienna UT, Vienna, Austria, Oct. 2007.

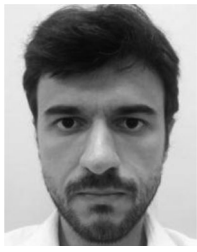
[16] L. Schor, P. Sommer, and R. Wattenhofer, "Towards a zero-configuration wireless sensor network architecture for smart buildings," in *Proc. 1st ACM Workshop Embedded Sens. Syst. Energy-Efficiency Buildings*, 2009, pp. 31–36.

[17] F. Pentaris, J. Stonham, and J. Makris, "A cost effective wireless structural health monitoring network for buildings in earthquake zones," *Smart Mater. Struct.*, vol. 23, no. 10, pp. 193–218, Oct. 2014.

[18] H. Oliveira, A. Boukerche, E. Nakamura, and A. Loureiro, "An efficient directed localization recursion protocol for wireless sensor networks," *IEEE Trans. Comput.*, vol. 58, no. 5, pp. 677–691, May 2009.

[19] D. Agarwal and N. Kishor, "A fuzzy inference-based fault detection scheme using adaptive thresholds for health monitoring of offshore wind-farms," *IEEE Sens. J.*, vol. 14, no. 11, pp. 3851–3861, Nov. 2014.

[20] D. Moss and P. Levis, "BoX-MACs: Exploiting physical and link layer boundaries in low-power networking," Computer Systems Laboratory, Stanford University, Stanford, CA, USA, Tech. Rep. SING-08-00, 2008.



Igor L. Santos received the master's degree in 2013 from the Federal University of Rio de Janeiro (UFRJ), where he is currently working toward the doctorate degree in informatics. He is also a lecturer at UFRJ. His research interests include wireless sensor and actuator networks, cloud computing, information fusion, and structural health monitoring.

Luci Pirmez received the PhD degree in 1996 from Federal University of Rio de Janeiro (UFRJ), where she is a researcher and professor of postgraduation courses in computer science. Her research interests include wireless sensor and actuator networks, network management, and information security.

Luiz R. Carmo received the PhD degree in informatics from the Universite de Toulouse III (LAAS/CNRS) in 1994. He is a senior specialist at the National Institute of Metrology Standardization and Industrial Quality (INMETRO) and a researcher at the Federal University of Rio de Janeiro (UFRJ). His research interests include information security and computer networks.

Paulo F. Pires received the PhD degree in 2002 from Federal University of Rio de Janeiro. He is an associate professor at UFRJ and integrates the Center for Distributed and High Performance Computing at the University of Sydney. His research interests include ubiquitous computing, model-driven development, and software architecture.

Flávia C. Delicato received the PhD degree in 2005 from Federal University of Rio de Janeiro, where she is an associate professor. Her research interests include wireless sensor and actuator networks, middleware, adaptive systems, and Internet of Things (IoT). She is a CNPq fellow level 1.

Samee U. Khan received the PhD degree in computer science from the University of Texas, Arlington. He is an associate professor of electrical and computer engineering at North Dakota State University. His research interests include cloud, grid, and big data computing, wired and wireless networks, smart grids. He is a senior member of the IEEE. For more information, see: http://sameekhan.org/

Albert Y. Zomaya received the PhD degree in control engineering from Sheffield University, United Kingdom. He is the chair professor of high-performance computing and networking in the School of Information Technologies, Sydney University. His research interests include complex systems, parallel and distributed computing, and green computing. He is a fellow of the AAAS, IEEE, and IET (United Kingdom).