Exploiting self-organization and fault tolerance in wireless sensor networks: A case study on wildfire detection application

Felipe Taliar Giuntini¹, Delano Medeiros Beder² and Jó Ueyama¹

Abstract
The wildfires caused by human occupation is one of the factors that most contributes to deforestation of conservation areas, resulting in a number of issues for ecological systems. Premature fire detections lead to the elimination or reduction of the damage that will be caused by a fire incident. Wireless sensor networks have been shown to be a good alternative for environmental monitoring applications, as they can collect and send information in real time, such as humidity, wind, and temperature of various parts of the forest. Due to problems such as power limitation, communication failure, and loss of nodes, the network topology is constantly changing, requiring mechanisms to achieve self-organization and fault tolerance. This article proposes the development of a model and application in self-organizing and fault-tolerant wireless sensor networks for fire detection in conservation areas. To achieve self-organization and fault tolerance, we rely on encouraged local interactions between neighboring nodes that monitor the same region and the coordination of tasks, through a supervisor node, equipped with a framework for developing component-based fault-tolerant applications. For the analysis and validation of both model and application, it was simulated 60 events in the network of sensors in a real scenario using the simulator Sinalgo. The results were classified as True (partial or absolute) or False (partial or absolute). In all, 45% of consensus identified a possible fault in the application and in only 35% there was absolute consensus.

Keywords
Wireless sensor networks for fire detection, self-adaptive wireless sensor networks, fault-tolerant wireless sensor networks, component-based wireless sensor networks, fire detection systems, N-Version applications, N-Version wildfire detecting systems

Date received: 12 May 2016; accepted: 20 March 2017

Academic Editor: Janos Botzheim

Introduction
One of the main environmental problems in Brazil is the high rates of wildfires caused by human occupation, placing the country among the main responsible for the increase in greenhouse gas emissions in the world. Besides contributing to global warming and causing climatic changes, fires cause social and economic losses and precipitate the process of deforestation and loss of biodiversity of the country. Although government

¹Institute of Mathematics and Computer Sciences, University of São Paulo, São Paulo, Brazil
²Department of Computer Science, Federal University of São Carlos, São Carlos, Brazil

Corresponding author: Felipe Taliar Giuntini, Institute of Mathematics and Computer Sciences, University of São Paulo, São Paulo, Brazil. Email: felipegiuntini@usp.br

Creative Commons CC-BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (http://www.creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (http://www.uk.sagepub.com/aboutus/openaccess.htm).
measures attempt to minimize this problem and punish those responsible for environmental disasters, the demand for land is currently the biggest reason of intense wildfire occurrences. Also, fires may also occur naturally in conservation areas due to increased drought and high-temperature events.

According to statistics from the National Institute for Space Research (INPE/Brazil) (http://www.inpe.br/queimadas/estatisticas.php), only in the year 2015 the reference satellite detected 236,371 fires. These fire outbreaks occurred 203,411 in conservation areas, with most of the biome Mata Atlântica (51%) and the biome Cerrado (37%)—see the full distribution in Figure 1. The average annual outbreaks detected since the start of monitoring (1998) is 175,870. These numbers may be even greater because the satellites detect fires from just 30 m long by 1 m wide. Figure 2 shows a comparison of the current year’s data with the maximum, medium, and minimum values, from 1998 to 21 January 2016. During the months from July to October, it is perceived that there is a greater number of fires, this is due to low humidity and high temperatures, very common in this season. However, the existence of satellite detection system is prominent to have approaches such as WSNs which perform the early detection of outbreaks of fires and notify in a timely manner managers and combat staff of the affected area.

WSNs can be defined as a large number of sensor devices connected through a wireless communicative technology in order to perform monitoring of an area in a cooperative manner, and it has been employed in various scenarios such as control, monitoring of environment and traffic, security, medicine, and biology. It is peculiar of a WSN to depend on the self-organization and adaptive algorithms, as this makes them less dependent on a centralized control, making it the most scalable, adaptable, and robust.2

This article aims to discuss the main research challenges related to the development of self-organizing and fault-tolerant WSN applications, especially some fire detection applications. Moreover, it presents a model that aims to fill the gap in the area, which is a model that responds to both low-level concerns and challenges regarding the application itself. As far it is known, research papers do not deal with both characteristics, or concentrate on the development of routing protocols or applications that do not take into account the fault tolerance and self-organization. The approach discussed in this article aims to address these two research challenges with a model that integrates both research areas.

**Background and research challenges**

With the continuous evolution of technology and the increasing software demand, computer systems have operated in more changeable environments, resulting in a continued increase in the administrative burden for managing these systems. The implication of this continuous evolution is that software systems need to become more versatile and flexible, more efficient in terms of energy consumption, fault tolerance, and self-optimization, adapting themselves according to the context they are inserted. This concern has been demonstrated by Garlan et al.3 and Cheng et al.4 and further increased over the years, due to high demand. Nevertheless, they need innovative approaches for building, running, and managing these systems.

WSNs can suffer several changes, which can be caused by (1) destruction or lack of energy of nodes, (2) problems in communication between the sensors, and (3) the network management algorithm, for example, to save energy or to have a neighboring node which already collects the same required data. Furthermore, inactive sensors can become active and new sensors can be inserted into the network. However, to manage all these changes and the WSN continue perform its function, it is essential to have self-organizing and fault-tolerant mechanisms. Section “Self-organizing mechanisms” presents the concepts and self-organization mechanisms.
while section “Fault-tolerant mechanisms” describes the fault-tolerance approaches.

Self-organizing mechanisms

Self-organization is a concept that enables systems made up of many subsystems operate autonomously to perform a group task. Moreover, self-organizing systems demonstrate an overall behavior, which can hardly be predicted or pre-programmed in a scalable manner. Investigations of properties of self-organization in natural systems began in the 1960 and since then have developed many techniques intrinsic to the fundamental concepts of self-organization.5

The general technological systems become organized by external commands, such as human intentions leading to the construction of machines and structures. However, self-organizing systems are those that become structured through their own internal processes, in which the emergence of order in itself is a complex event that intrigues scientists from different fields.6

Afterward, self-organization is defined as a standard process in which the overall level of a system arises only from the many interactions between the lower level elements of the system. Furthermore, it is specified by rules that the interactions between system components only be performed using only local information, that is, without reference to the overall pattern.7

This section summarizes the idea of self-organization as the creation of a global behavior from local interactions between system elements. In all, some self-organizing properties are summarized in Table 1 by Dressler.5

| Property            | Description                                                                 |
|---------------------|-----------------------------------------------------------------------------|
| No central control  | There is no global control system or global information available. Each subsystem must perform completely autonomously |
| Emerging structures | The global behavior or functioning of the system emerges in the form of observable patterns or structures |
| Resulting complexity| Even if the individual subsystems can be simple and perform basic rules, the resulting overall system becomes complex and often unpredictable |
| High scalability    | There is no performance degradation if more subsystems are added to the system. The system should perform as requested, regardless of the number of subsystems |

Self-organizing mechanisms

Self-organization is a concept that enables systems made up of many subsystems operate autonomously to perform a group task. Moreover, self-organizing systems demonstrate an overall behavior, which can hardly be predicted or pre-programmed in a scalable manner. Investigations of properties of self-organization in natural systems began in the 1960 and since then have developed many techniques intrinsic to the fundamental concepts of self-organization.5

The general technological systems become organized by external commands, such as human intentions leading to the construction of machines and structures. However, self-organizing systems are those that become structured through their own internal processes, in which the emergence of order in itself is a complex event that intrigues scientists from different fields.6

Afterward, self-organization is defined as a standard process in which the overall level of a system arises only from the many interactions between the lower level elements of the system. Furthermore, it is specified by rules that the interactions between system components only be performed using only local information, that is, without reference to the overall pattern.7

This section summarizes the idea of self-organization as the creation of a global behavior from local interactions between system elements. In all, some self-organizing properties are summarized in Table 1 by Dressler.5

As seen above, the WSN can be used in various scenarios and applications, and the sensors can be distributed manually at predetermined sites or thrown in the area that you want to monitor. The latter one should be applied where the network contains hundreds of sensors or in case that the monitored site is remote and inaccessible. WSNs can change their constantly topology for various reasons, such as lack of battery or destruction of nodes, new sensors can be inserted, and sensors can switch on and off to save power, among others. Therefore, these networks need to be self-organizing, that is, you must have the ability to adjust to change without human interference.2

The main difficulties of a WSN in terms of self-organization would be easily solved with a global view of the network, so it would be possible to manage groups of sensors that need to interact, establish more intelligent data transmission routes, and check the sensors, which they can be turned off at certain times in order to save energy and therefore allow the long suffering of the network.

However, with respect to most of the applications in WSN, in practice it is required that they have autonomy, that is, work without human intervention as long as possible, and are scalable and robust. Thus, it is considered to be most beneficial to use algorithms located in the design of these networks, in which the sensors interact only with each other and collectively to achieve the overall goal.5 Thus, self-organization in WSN is characterized by Loureiro et al.2 as the ability to make changes in its structure without human action, so make them scalable and robust before the dynamic characteristics intrinsic to the type of the network.

In general, the methods of self-organization try to reduce or even extinguish the centralized control, taking into account the power constraints, processing power, storage, and mobility.5 However, the most important concerns that must be addressed are scalability and network lifetime, briefly described below:

**Scalability.** This parameter portrays how many communicating nodes are supported by the network, with the worst case, a linear increase in the consumption of resources. As implementation scenario, scalability may be the limiting agent in sensor networks, such as networks with a large number of nodes.5

**Lifetime.** The lifetime of a sensor network is nothing more than the ability of the network to meet application requirements. In general, the fact that some of nodes can fail is ignored while the overall goal is always respected. Moreover, this parameter is the question of whether the sensor network is always available, or if there are special maintenance periods.5
In context of complex and massively distributed systems, approaches such as feedback cycles, interactions between system components, and probabilistic techniques can be applied to achieve self-organization. Considering that the domain of WSN and ad hoc network is very specific, a brief description of possible applications of these methodologies is given. Furthermore, some works related to this topic are presented in section “Related work”:

**Feedback loops.** Used to update the operating parameters of the systems or groups consisting of several subsystems in accordance with the current state, adaptive algorithms can be constructed using feedback loops negative and positive, the first being used for the amplification and the second for control of the adaptation process. Overall, in WSNs, feedback is given by means of observation and evaluation of various system parameters. If the focus is changing environmental conditions, sensors can be used locally to accomplish these measurement conditions. Thus, the analysis of periodic reading of the sensors can start the adjustment of the parameters if necessary. Moreover, feedback may also be given through neighboring nodes that measured or detected some environmental change, in which the system or group of systems must adapt. Nevertheless, neighborhood information and the local state are related to the application of adaptive algorithms, which monitor the individual sensor node performance.

**Interactions between system components and the environment.** There are three types of interactions, which are interactions with the remote nodes with direct neighbors or through indirect interactions. Interactions with remote nodes may be needed to achieve a common goal. Examples which fall into this category are the routing optimization algorithms or coverage mechanisms, since they depend on interactions with remote nodes. Since the interactions carried out with direct neighbors are highly necessary for the efficient use of radio resources, collision avoidance systems can handshake and synchronization schemes. In addition, it is possible to perform direct interactions with the environment in which node is inserted, or even, indirect interactions with other nodes in the network, as changes occur in the environment, since it is possible to explore the environment and adapt the local state according to the measurement of physical phenomena, selecting operating parameters and algorithms accordingly.

**Probabilistic techniques.** Probabilistic techniques have been used in different contexts. Two examples in sensor networks are probabilistic routings, as gossipping-based, and the allocation of probabilistic tasks. Additionally, the randomness is used in various protocols and mechanisms to avoid the undesirable effect synchronization. Probabilistic methods can also be used to prevent bottlenecks in the network.

### Fault-tolerant mechanisms

At the beginning of computer science, the concern in developing fault-tolerant systems were exclusively of critical system designers, such as industrial control systems, aircraft, and aerospace. However, with the popularization of the services offered on the network, the simplest of computers must present a minimum of reliability. In the case of WSN applications, it is crucial due to changes that may occur in the network and also the inherent applications.

The main goal of fault tolerance is to achieve dependability, which indicates the quality and confidence in the service delivered by a system. The concept of tolerance failure was first presented by Avizienis, which says the following: a system is fault-tolerant if its programs can be run properly, despite the occurrence of logical flaws. That is, with fault tolerance can deliver the same service expected in the presence of faults.

All fault-tolerance techniques involve some type of redundancy, requiring additional components or special algorithms. They are generally divided into two categories: (1) masking or (2) detection, location, and reconfiguration. Adopted for critical real-time systems, often the first category is more employed than the second form of redundancy because it does not involve the time spent to detection, localization, and reconfiguration.

Replication hardware identical components to achieve fault tolerance are interesting, but in the case of software component, this approach is inadequate because identical software components will present identical errors. Therefore, copying a program and running it in parallel or running the same program twice does not achieve fault tolerance. Among the proposed techniques of redundancy to achieve software fault tolerance, it is worth to mention the N-Version Programming Technique proposed by Avizienis and Recovery Blocks, devised by Randell. The first consists in the implementation of different alternatives, so that the system response is defined by vote. This technique does not take into account that errors in alternative solutions may originate from the same cause, such as error in the interpretation of the specification. Thus, so that errors are detected, it is necessary to express differently from other implementations. Second, though it is similar to the previous one, it differs in the fact that it involves an acceptance test. In this case, the minor versions will only be performed if an error is detected in the primary version, that is, the algorithms are executed one by one until the first pass the acceptance test.
Related work

This section presents the research work related to self-organization and fault tolerance, as well as wildfire detection applications developed for WSNs.

Works on self-organization and fault tolerance in WSNs

D Chen et al.\textsuperscript{12} presented a model of a WSN application for monitoring emergency events. The model operates in two states, vigilant and detection, and the system remains in the first mode while there is no occurrence. To save power, the system works in an alternative way, since some nodes are selected to operate with low sampling frequency at a given time interval, while other nodes continue sleeping. At each time interval, the roles are swapped. Once an emergency event is detected, the system will operate in the state detection, since the node immediately transmits the information throughout the network, so that all nodes wake up and become aware of the occurrence. In time, the system finds and marks the exact spot where the event started and provides the scope, that is, the possible infected area. Finally, the control room receives the data from the sink node, which maintains contact with the situation, thus allowing first responders to be sent to the infected area, the steps that nodes reported the situation in real time through multiple hops.

K Tatara et al.\textsuperscript{13} propose a decentralized solution to handle with the self-organization problem in many applications composed of a multitude of autonomous robotic sensors. This decentralized solution allows each robot to build and maintain the network, through local communication interactions with their neighbors, allowing robots choose as neighbors those with greater connectivity, adapting to the topological changes of the network through the robot’s movements. The proposal was validated through extensive simulations and realistic considering communication delays and robot failures.

M Vilela and R Araujo’s\textsuperscript{14} work proposes a routing solution for mobile WSN consisting of fixed nodes and mobile. The solution aims to collect data from sensor nodes that send data to the sink, through intermediate nodes and different mobility. Basically, the solution consists of three phases: in the network training phase, all nodes receive the sink location information. In the second phase, an election of cluster-head is performed when an event occurs, considering the levels of mobility, away from the sink and energy. Thus, all data collected by the sensors are sent to the cluster-head. In the third phase, the data received by cluster-head are delivered to the sink. According to the research, the solution can be efficient in terms of messages of overhead and transmission of data packets. In addition, the main goal is that this solution is adaptable to different domains.

Y Joshi and M Younis\textsuperscript{15} presented a distributed algorithm for autonomous repair (AUR) of damaged WSN topologies due to failures of sensor nodes. First, it is assumed that the nodes are aware of its position monitored area and by means of contemporary location schemes. It also assumes that all nodes have the same transmission range, which is much smaller than the monitored area. Failure of a node of the network may result in loss of connectivity and coverage. From this, the algorithm focuses on reestablishing connectivity, and the recovery process is started locally in the neighboring of failed nodes, moving them in the direction of loss, to reconnect with the other nodes. Moreover, to restore some coverage is used in self-propagation optimized by moving them to the center of the monitored zone. The simulation results show that the AUR exceeds the current solutions in terms of recovery of average degree of fault nodes and overhead coverage.

LA Villas et al.\textsuperscript{16} present a Data Aggregation Aware Routing Protocol (DAARP) to select routes with the highest aggregation rate and perform reliable data aggregation transmission. The protocol considers four types of roles that sensor nodes can have: Collaborator, Coordinator, Sink, and Relay. It performs three phases: first, the node starts sink building the hop tree, which will later be used by the Coordinator to forward the data. The second phase consists of the cluster formation between nodes that detected the same event on the network; thereafter, it is held the election of Coordinator of the cluster, responsible for gathering the data of Collaborator and notifies them. The last stage is in charge of creating a new route for reliable transmission of data and updating the hop tree.

Wildfire detection applications of WSNs

M Hefeeda and M Bagheri\textsuperscript{17} present a WSN project for early detection of fires from an analysis of the Fire Weather Index System (FWI), one of the most comprehensive systems risk notification of forest fires in North America. From this analysis, this work emphasizes the main aspects that should be considered in the modeling of fires and how different components of the FWI can be used in the design of efficient fire detection systems. The fire detection problem is solved with a distributed algorithm with K-coverage. Furthermore, a data aggregation scheme is proposed that extends the life of the network. The project was validated by simulation.

Y Liu et al.\textsuperscript{18} propose a fire detection system that uses WSN application of multi-criteria to increase the detection accuracy of a wildfire because an alarm decision needs many attributes of a fire. The multi-criteria detection is implemented by an artificial neural network that combines the sensed data of various attributes of a
fire into an alarm decision. The use of neural network allows self-learning and low network overhead. They assume that each node is aware of its location, provided by base station or Global Positioning System (GPS) device, and has various sensors, such as humidity and temperature. Using the prototype developed, experiments were conducted to evaluate the performance of the proposed system. The results show that the prototype fire alarm with high accuracy. Furthermore, a solar battery has been developed and proved to be able to meet the network for years.

In Yoon et al.19 is presented a reliable monitoring system based on WSN deployed in adverse conditions, wherein asymmetric, irregular, and unreliable wireless links, routing problems, among others. Reliable communication systems is used in a fault-tolerant topology, ensuring the delivery of data to the base station and allowing real-time data display. In this work, 20 sensors were implanted very sparsely to collect temperature and humidity data. The nodes are designed to turn off their transceivers to save power and synchronize their sleep and wake-up cycles to simplify communication. Nodes that are awake may operate in normal mode or in warning mode. Alert mode implies that the fire was detected, which requires that the sensory data should be delivered as quickly as possible even though it consumes more energy. The end-to-end delay in data transmission to the sink node is reduced using a modified PISA-I routing algorithm (A priority-based routing algorithm for solar-powered WSNs).20

Table 2 presents a summary of the related work highlighting the objectives, main approaches, and whether or not there was an application. Through the related works, it is possible to perceive that some of them employ common technical approaches, for example, it is noticed that the location of the sensor nodes is used in all the works, by exploring the routing tree or by GPS, as in case of Y Liu et al.18

In addition, the works of D Chen et al.12 and I Yoon et al.19 in order to achieve self-organization, use features such as switching status of sensor nodes and timing mechanisms, that is, sensors alternate between on and off, reducing the consumption of energy. The works of M Vilela and R Araujo14 and LA Villas et al.16 in addition to the location, employ clustering of sensor nodes and leader election approaches. While the first one performs the clustering considering the connectivity of the mobile or fixed nodes, the second performs clustering based on the routing tree. In addition, the second one stands out because we can assume several roles and have a data aggregation mechanism based on routing.

This section emphasizes that it is scarce to find a WSN-related work that reunites self-organization and fault tolerance concerns in the same project, especially at different levels, such as in network and application layer. Also it is worth to mention that self-organization in WSN is firmly linked to exchange of sensor node states (or roles) and fault tolerance at the application level is achieved through diversity. Thus, the proposed model attempts fill this gap—a model to guide the development of WSN applications taking into account concerns at different levels, such as routing and application. In critical applications, such as fire detection, this is crucial.

Among the related works presented, those that allow some comparison with the following proposal, that is, the most similar works, are Hefeeda and Bagheri17 and Liu et al.18 by the application in fire detection and

| Authors                  | Objective                                                                 | Approaches                                             | Application |
|--------------------------|---------------------------------------------------------------------------|--------------------------------------------------------|-------------|
| D Chen et al.12          | Model of a WSN application for monitoring emergency events                 | Status switching, location and timing                  | No          |
| K Tatara et al.13        | Self-organizing ad hoc robotic sensor networks based on locally communicative interactions | Location                                               | No          |
| M Vilela and R Araujo14  | A routing solution for mobile WSN consisting of fixed nodes and mobile   | Location, clustering, election                          | No          |
| Y Joshi and M Younis15   | Distributed algorithm for autonomous repair of damaged WSN topologies    | Location                                               | No          |
| LA Villas et al.16       | Data Aggregation Aware Routing Protocol                                  | Sensor node roles, locations, clustering, election, data aggregation | No          |
| M Hefeeda and M Bagheri17| WSN project for early detection of fires                                  | Data aggregation, distributed algorithm with K-coverage | Yes         |
| Y Liu et al.18           | Fire detection system                                                    | Location, multi-criteria, artificial neural network     | Yes         |
| I Yoon et al.19          | A reliable monitoring system based on WSN deployed in adverse conditions | Status switching, timing                               | Yes         |

WSN: wireless sensor network.
Villas et al.\textsuperscript{16} by the self-organized communication protocol. All these works use aspects of location and aggregation in some way. As for the solution, while Hefeeda and Bagheri\textsuperscript{17} use an algorithm distributed with k-coverage, Liu et al.\textsuperscript{18} use a multi-criteria approach in conjunction with an artificial neural network. In our proposal, a fault-tolerance technique based on diversity of solutions and majority voting is applied. In addition, unlike the previous approaches in which the detection solution is located on the server, our solution is allocated on all sensor nodes and are started on demand, that is, the processing occurs in the network. Details of our proposal can be found in the next section.

A self-organized and fault-tolerant model

This section presents the model to achieve a WSN self-organizing and fault tolerant in wildfire detection applications in forests. The model consists of the following phases: (1) implementation and establishment of the network; (2) announcement event, clustering, and coordinator election; (3) coordination tasks and detection of wildfires, and (4) aggregation and sending data to the sink. Each step is explained in detail below.

Implementation and establishment of network

Initially, the nodes are randomly dropped on the area of interest and awaken to network formation. For this, the sink sends a control message on broadcast to the entire network in order to build the routing tree. That is, each node that receives the message calculates and stores the number of hops and sends the message to their neighbors. They continue doing this until the entire network is achieved. In accordance with Busulu et al.,\textsuperscript{21} the recommended density of a WSN is close to 21.7 and its distribution, that is, the number of nodes required to monitor a region, can be calculated as 

\[ \left( \frac{N \pi R^2}{4} \right)^{1/3} \]

where \(N\) is the number of scattered sensor nodes in region \(A\) and \(R\) is the radio transmission range.

Announcement event, clustering, and coordinator election

The trigger event can occur in various ways depending on application requirements. An example of a fire detection application is when there is a considerable increase in temperature. A characteristic of WSNs, especially those applied in environmental monitoring, is to be dense networks, which increases network reliability and guarantees packet delivery even in case of failure of some nodes. However, it is common that multiple nodes detect the same event. Thus, an approach to prevent this and sending duplicate data to the Sink is the clustering of these nodes and the election of a node with the coordinating role. Thus, each node after collecting the data send them for analysis coordinator (Figure 3).

The election of a coordinator to be given in different ways. The model proposes a new algorithm for coordinator election (Algorithm 1). This algorithm is based on the leader election algorithm proposed by the DAARP protocol\textsuperscript{16}, with the differential to consider the energy level of nodes in the choice for a better network load balancing.

Initially, the algorithm takes as input all nodes belonging to the cluster, that is, nodes which have detected the same event, since all are eligible. If the first event of the network will be elected Coordinator, the node has the shortest distance to the Sink. If the number of hops is equal, the node that has higher energy level will be elected. Finally, if there is a tie in both instances, the ID will be used as tiebreaker. Already from the second occurrence of the event, will be elected coordinator node, the node that has the smallest number of hops in the existing routing tree. Similarly, the first event will be considered the energy level and the ID to tiebreak, if necessary. All other nodes that detect the same event are nodes that own the collaborator role. The collaborators nodes transmit data detected for evaluation coordinator.

In this way, it is possible to ensure a better network energy balancing, because as the coordination tasks consume more energy, which is considered in the election. Thus, hardly a node will be elected coordinator more than once in a row.

Coordination tasks and detection of wildfires

In most of the research papers found in the literature, coordinating tasks are summarized to just collect data
from collaborators aggregate and send to Sink, and send to Sink that is in charge of data analysis. This is due to the application being allocated only in the sink or the base station, not distributed by the network. Consequently, many data end up being sent to the sink, consuming unnecessary energy, since not all announced events are actually considered fire, as false alarms may occur and the application must be prepared for this.

In addition to the common tasks relevant to the coordinator, the model suggests that the detection of fire is performed on the coordinator, that is, the solution would be allocated to the sensors that are not in the base station. To do this, it is assumed that the solution was previously deployed in all sensors. However, it will only be instantiated only when the sensor is in the role of coordinator. Once the sensor is back to collaborator role, the application will be deallocated from memory. It is worth to say that this happens in an adaptive approach.

After all, in order to avoid overloading the network and make possible low power consumption, the proposed model encourages the utilization of software componentization. It is understood that solutions are prone to failure and therefore fault tolerance must be considered since the design of application. One way to implement fault tolerance in software is through diversity. That is, different algorithms address the same problem.

However, it is proposed that the fire detection is performed by various solutions, where in each solution is built as a component, which would load and unload at runtime. To manage this dynamic and achieve fault tolerance and give an accurate result, the N-Version Programming technique was implemented as discussed before (section “Fault-tolerant mechanisms”). That is, various fire detection solutions proposed by domain experts are executed at runtime, generating results true or false and an error probability. After that, a vote is performed, that is, a consensus of all the results. For example, there are three decision trees and the results of each decision was true, respectively, True 6%, False 10%, and True 7%. Our approach counts the number of trues and falses and calculates the average of the probabilities. That is, in this case, the result of consensus would be True 6.5% ((6 + 7)/2). The result is True because two of decisions tree provide True as its outcome. The probability result is 6.5% because it is the probability average of them (decision trees that provide true as its outcome).

---

Algorithm 1: Coordinator election

**Input:** S // Set of nodes that detect the event  
**Output:** u // A node of cluster is elected coordinator  

**Initialization:**
1. for each u in S do  
2. roleu ← Coordinator  
3. u sends message in broadcast to neighbors (w) belonging to the cluster  
4. if EventNum = 1 then  
5. if HopstoSinku > HopstoSinkw then  
6. roleu ← Collaborator  
7. Node u retransmits the message received from node w  
8. else if HopstoSinku = HopstoSinkw and Energyu < Energyw then  
9. roleu ← Collaborator  
10. Node u retransmits the message received from node w  
11. else if HopstoSinku = HopstoSinkw and Energyu = Energyw and IDu > IDw then  
12. roleu ← Collaborator  
13. Node u retransmits the message received from node w  
14. end if  
15. else  
16. if EventNum > 1 then  
17. if HopstoEventu > HopstoEventw and Energyu < Energyw then  
18. roleu ← Collaborator  
19. Node u retransmits the message received from node w  
20. else if HopstoEventu = HopstoEventw and Energyu ≤ Energyw and IDu > IDw then  
21. roleu ← Collaborator  
22. Node u retransmits the message received from node w  
23. end if  
24. end if  
25. else  
26. Node u discards the message received from w  
27. end if  
28. end for
Aggregation and sending data
After completion of the consensus, that is, the final decision, the coordinator might or might not send the data to the Sink. The domain expert can set up as desired. For example, the configuration should be to send only positive results. As a consequence, there is a network energy saving, since there is no transmission of false alarms. However, if the coordinator wants to store all data for future statistical studies, it can be configured to send all events, trues and falses.

Furthermore, identical results may be aggregated to reduce the number of duplicate transmissions. Finally, Figure 4 summarizes the main activities of the proposed model steps.

Implementation details and evaluation
To evaluate and validate the model, a simulation was implemented in Sinalgo v. 0.75.3.22 a framework developed in Java (https://www.java.com/) for testing and validating network algorithms.

The simulation was performed on the area of environmental preservation of the Botanical Garden of Brasília, which is a nature conservation unit attached to the Secretariat of Environment and Water Resources of the Federal District. The institution owned 500 hectares of public visitation, covering various vegetation types Cerrado biome. It also has 4500 hectares of ecological reserve for research and conservation of the Cerrado.

To achieve the density 21.7 and applying the density calculation presented in section “Implementation and establishment of network” for an area of 4500 hectares with radio communication of 100 m each node, approximately 31,082 nodes were necessary to monitor all ecological reserve.

Initially, the model sensors were able to monitor humidity, temperature, and wind and distributed over the area considering the calculation of the density shown in phase A of the model. For low-level tasks, such as routing, any communication and clustering, the DAARP16 was employed with some modifications. As mentioned in section “Announcement event, clustering, and coordinator election,” the proposed model uses a coordinator election algorithm that takes into account the energy level of eligible nodes. In addition, the message structure was changed to support data detected from sensors and application. This protocol has been very efficient as compared to other protocols in the literature, it reduces the number of messages needed to set up a routing tree, maximizes the number of routes that overlap, selects routes with a high rate of aggregation, and possesses reliable delivery of data.

At coordination of task phase (section “Coordination tasks and detection of wildfires”) resides the main differential of this work—to implement the solutions as components, allowing them to be loaded (and unloaded) at runtime and employing the N-version technique through the framework FlexFT.23 FlexFT consists of a general structure, based on components and written in Java, for the construction of adaptive fault-tolerant system that integrates the various devices and technologies, both software and hardware. The framework consists of a small core where fault-tolerance policies are inserted on demand, as they are implemented as components and are instantiated as needed. FlexFT is built on the OpenCom,24 which allows for the load, unload, and destroy operations at runtime and adaptive manner. Among the benefits mentioned by authors are flexibility, reusability, skills transfer, and support to independent and interoperable technologies.

As shown in Figure 5(a), the architecture of FlexFT is composed of two layers, fault-tolerant component frameworks and Component Runtime Kernel Layer. The first layer provides mechanisms for the development of systems based on reliable components. The
second layer supports the development of reliable systems based on components. As shown in Figure 5(b), the approach implements different versions of reliable components. The versions (FirstVariant, SecondVariant, ...) are functionally similar components, which are constructed independently based on the same initial specification. ReliableComponent has the task of coordinating the implementation of the versions and call the relevant operations such the vote operation employed by the N-Version technique (section “Fault-tolerant mechanisms”). The binding mechanism has the task of calling the provided and required interfaces.

Prototype fire detection solution

The fire detection solution consists of three decision trees that were implemented as components: (1) DT1—Figure 6(a), (2) DT2—Figure 6(b), and (3) DT3—Figure 6(c).

These decision trees will be only loaded when the sensor node is playing the coordinator role, as mentioned before (section “A self-organized and fault-tolerant model”). In order to develop the decision trees, Forest Fires Data Set was used with 517 instances of UCI Machine Learning Repository and the data mining tool Weka. The first step was to classify the database together with a specialist in forest fire detection, that is, a manager responsible for the control team, management, and firefighting in the Brasilia Botanical Garden. Of all attributes of the dataset, it used only those that our sensor model is intended to detect, that is, wind, relative humidity, and temperature. The rain attribute was not used because it does not have any impact on decision trees, that is, it is an irrelevant attribute. The decision trees have been generated with different instances. Decision trees have different cross-validation rates, which are, respectively, 84.05%, 61.24%, and 73.66%.

Details about the accuracy are presented in Tables 3–5. Their confusion matrix is presented in Tables 6–8, respectively.

The purpose was to demonstrate that the decision trees can be different solutions with different classification rates, built by groups of features and levels of
knowledge and different specialties. Without a doubt, the best solution is that built by a set of domain experts. Therefore, it is emphasized that the proposed model is effective in case of solutions built by these experts. However, these detection models can work together in order to increase the dependability.

As shown in Figure 7, the proposed component model diagram contains the DecisionTreeNV component and its variants DecisionTreeVariantOne, DecisionTreeVariantTwo, and DecisionTreeVariantThree. Each of these variants consists of the implementation of a fire detection solution, that is, respectively, the decision trees DT1, DT2, and DT3.

As shown in Figure 8, the class diagram model contains the class DecisionTreeNV that represents the FlexFT ReliableComponent (Figure 5(a)) by extending the abstract class NVComponent and implementing the execute() method, taking into account the peculiarities inherent to N-Version programming technique. In addition to invoking the variants (load and unload OpenCOM operations), the method run() is in charge of obtaining the decision result provided by the method decide() (NVInterface interface).

Since the FlexFT framework offers a standard decide algorithm (majority) implemented by the NVComponent class, to meet the application requirements the method decide has been overridden in DecisionTreeNV class, which is responsible for calling

| Table 3. Detailed accuracy by Class DT1.
| Class | TP rate | FP rate | Precision | Recall | F-measure | Roc area |
|-------|---------|---------|-----------|--------|-----------|----------|
| Not   | 0.846   | 0.163   | 0.759     | 0.846  | 0.8       | 0.873    |
| Fire  | 0.837   | 0.154   | 0.9       | 0.837  | 0.867     | 0.873    |
| Average | 0.841 | 0.157   | 0.847     | 0.841  | 0.842     | 0.873    |

TP rate: true positives; FP rate: false positives.

| Table 4. Detailed accuracy by Class DT2.
| Class | TP rate | FP rate | Precision | Recall | F-measure | Roc area |
|-------|---------|---------|-----------|--------|-----------|----------|
| Not   | 0.682   | 0.45    | 0.578     | 0.682  | 0.625     | 0.632    |
| Fire  | 0.55    | 0.318   | 0.656     | 0.55   | 0.598     | 0.632    |
| Average | 0.612 | 0.381   | 0.619     | 0.612  | 0.611     | 0.632    |

TP rate: true positives; FP rate: false positives.

| Table 5. Detailed accuracy by Class DT3.
| Class | TP rate | FP rate | Precision | Recall | F-measure | Roc area |
|-------|---------|---------|-----------|--------|-----------|----------|
| Not   | 0.872   | 0.544   | 0.769     | 0.872  | 0.817     | 0.634    |
| Fire  | 0.456   | 0.128   | 0.632     | 0.456  | 0.529     | 0.634    |
| Average | 0.737 | 0.409   | 0.724     | 0.737  | 0.724     | 0.634    |

TP rate: true positives; FP rate: false positives.

| Table 6. Confusion matrix DT1.
| a        | b        | as classified as |
|----------|----------|-----------------|
| 22       | 4        | a = not         |
| 7        | 36       | b = fire        |

| Table 7. Confusion matrix DT2.
| a        | b        | as classified as |
|----------|----------|-----------------|
| 167      | 78       | a = not         |
| 122      | 149      | b = fire        |

| Table 8. Confusion matrix DT3.
| a        | b        | as classified as |
|----------|----------|-----------------|
| 143      | 21       | a = not         |
| 43       | 36       | b = fire        |
the static method decide() in the class of interface IDecisionTree passing a list of results.

This results list is built by the algorithm decision() in DecisionTreeNV, which receives the data from the sensors and invokes the method run() passing as parameters the data for each variant. The returned results are stored, which subsequently are passed as parameters to the method decide(). In this case, the latter method counts the number of true and false results and calculates the average of error probabilities. Finally, the final consensus of all occurrences is returned with the mean of probabilities. This functionality is described in the source code 1.

It is worth to say that the IDecisionTree interface, shown in the class diagram model, is implemented by the DecisionTreeNV class.

**Java Code:**

```java
static public Object decide(List<?> results) {
    int counter[] = new int[2];
    double probability[] = new double[2];
    for (Object result: results) {
        Decision d = (Decision) result;
        if (d.isFire()) {
            counter[1]++;
            probability[1] += d.getProbError();
        } else {
            counter[0]++;
            probability[0] += d.getProbError();
        }
    }
    if (counter[1] >= counter[0]) {
        return new Decision(true, probability[1]/counter[1]);
    } else {
        return new Decision(false, probability[0]/counter[0]);
    }
}
```

**Figure 7.** Proposed component model diagram.

**Figure 8.** Solution class diagram.
Solution deployment in DAARP sensor node

To comply with the application requirements, it was necessary to reimplement a method of sensor node class implemented by the DAARP protocol. Previously, only the node received the data and headed for the sink node through the route established by the protocol. Because the application is allocated on the sensor node when the SentInformationEvents method is called, it is verified if the node has the role of coordinator and if it has already received all the data, that is, when the variable send data equals true. If this is true, the application will start running. First, create the OpenCOM runtime environment and request the IOpenCOM interface. Soon after, the DecisionNV component and decision components are created, that is, decision trees. Afterward, the IDecisionTree interface is required and finally the decisions are executed, considering values of wind, humidity, and temperature provided by the sensor nodes. In this case, the message was passed by parameter to the method.

Subsequently, the results of the consensus are stored in a list and only positive results are sent to the Sink node. Note that this should be configured by the domain expert. In addition, you can configure to send all results or some adaptation taking into account expert guidance.

Results and discussion

The simulation was replicated 10 times with 6 events each, generating a total of 60 events. The values of wind, humidity, and temperature of each node were also generated randomly, considering a range of minimum and maximum values of the database.

The simulation was performed on a Notebook with Intel i5 2.4 GHz and 8 GB of RAM. A large number of sensors were dedicated 4 GB of RAM for the Java virtual machine and the simulation was performed via console, that is, without using the graphical interface Sinalgo. The simulation setup parameters are shown in Table 9.

For analysis purposes, it was chosen by sending all results. The consensus of each event was analyzed and classified as:

- Total False. Fire not detected and there is an absolute consensus of the decisions trees.
- Total True. Fire detected and there is an absolute consensus of the decisions trees.
- Partial False. Fire not detected and there is a partial consensus of decisions trees.
- Partial True. Fire detected and there is a partial consensus of decisions trees.

The results of the consensus, presented in Figure 9, show that there are two kinds of decisions: (1) unanimous and (2) consensual majority:

- Unanimous (Total True or Total False). When the decision is unanimous, all decision trees showed the
same result. That is, any failure is detected. As shown in Figure 9, 35% of events were announced as consensual fire and 20% of events were consensual fake events (no fire).

**Consensual Majority (Partial True or Partial False).** When the decision is consensual, the consensus process detects a possible failure. That is, there is a solution result that is different from the other two. As shown in Figure 9, 45% (35% Partial False + 10% Partial True) of the consensus detect a possible failure. This shows that proposed model is effective in detecting failures in case the detection solutions are constructed by a group of specialists in wildfire detection.

In addition, if the application was configured in a such way that the coordinator node only send to the sink the events that produced *Total True* consensus, it would be a saving of 65% in the number of transmissions if compared only with the results of the implementation of the DAARP protocol. Since the literature have shown that transmission data are the main energy consumer in a WSN, thus the network lifetime would be higher. Likewise, even in case of allowing the transmission of events with partial consensus (*Partial True*) to be analyzed by an expert, it would be a saving of 55% in the number of transmissions. Finally, if it is desired to send all probable events of fire, that is all consensus result except the *Total False*, it would be a saving of 20% in the number of transmissions.

Furthermore, by looking at Figures 10 and 11, it is possible to see the effectiveness of the N-Version fault tolerance technique by means of observing (1) the variation of each consensus per round and (2) the variation of each consensus in all rounds respectively. It is possible to see that in almost all rounds there is the presence of partial consensus, that is, at least one detection model (decision tree) disagreed with the others, indicating a possible detection failure. Moreover, it is possible to note that in regard to the 10 rounds, the partial false appeared in 9 of them. Similarly, the true partial consensus appeared in 5 of them, showing a large distribution and high amount of consensus by majority votes, that is, without unanimity.

For the purposes of comparison and considering the accuracy rates, the work of M Hefeeda and M Bagheri presents in their approach a 95% fire detection rate, considering only the temperature. However, our work shows that analyzing only temperature data in the fire detection process is not enough and can lead to unequivocal detection. This is due to the fact that 20% of the events initially announced, considering the increase in temperature, were considered false events, when wind and relative humidity data became part of the detection. In addition, Y Liu et al., using infrared camera and multi-criteria approaches, achieved detection rates of up to 100% when at a distance of 10 cm from the focus. However, this accuracy rate decreased dramatically as the distance between the sensor node and focus increases. For example, the accuracy decreases to 30% when the sensor is within 30 cm of the focus, showing to be an inefficient approach in monitoring large areas. Regarding the other related works presented, they are focused on the self-organizing mechanisms in the protocol or low-level layers, focusing on increasing network lifetime and saving energy. Considering the different aspects, this work contributes to the fact that in this sensory context, if one considers the application of the network, it is possible to obtain an economy in the number of transmissions of up to 65%.

In addition, unlike the approaches presented in section “Related work,” this work stands out for presenting a fire detection model that uses a set of decision trees, which consider data of wind, humidity, and temperature for decision making. Beyond which, it stands out for fault tolerance in the WSN application.

**Concluding remarks and future works**

This article presents a self-organizing and fault-tolerant model for wildfire detection in areas of environmental
preservation. The main objective was to develop a model which aggregates concerns at various levels in a WSN. The main differential of this study was to explore how the componentization of software, mixed with software diversity techniques (N-Versions programming), can be used in WSN both to create self-adaptive low-power applications and to develop fault-tolerant applications.

To achieve different levels of self-organization and fault tolerance, it was necessary to integrate various approaches and solutions previously proposed, among them, the protocol DAARF\textsuperscript{16} and the frameworks: FlexFT,\textsuperscript{23} OpenCOM,\textsuperscript{24} and Sinalgo.\textsuperscript{22}

The objective of this work was not to develop the best fire detection solution, considering that a strong group of heterogeneous professionals is required in the development process. However, this can be considered a step of every journey. Moreover, the differential of this application is to be allocated in the sensor nodes that form the network, being instantiated when the node assumes the coordinator role, avoiding the unnecessary sending of data to the base station, where a large part of the applications are allocated in the literature.

Through the simulation on the area of the Botanical Garden of Brasilia, it was possible to show the model of implementation feasibility in a real scenario. Using the N-Version Programming technique through FlexFT framework, it was possible to demonstrate that there are possible evidence of failures in decision trees, which is true, since the class rate are not very high. Moreover, it was considered that the general solutions can not perfect and that the organization of different solutions can increase confidence in the application.

It is also noticed that many of the detected events were just false alarms, considering that the announcements of events took into account only an increase in temperature. That is, wind and relative humidity will greatly contribute to the process of wildfire detection. Thus, no transmission of alarms can have a considerable saving of energy and hence an increase in network lifetime.

Finally, by what is known in the literature, it is possible to list the following contributions of this work:

1. A self-organized and fault-tolerant model of WSN for wildfire detection. So far it is known that a model in fire detection context has not been reported in the literature that systematizes the process of a WSN.

2. A software solution prototype for fire detection. The proposed solution stands out for the following reasons: (1) is allocated in the sensor nodes, not on the server as the work in the literature, allowing early detection and saving transmissions, given that false alarms are not sent to the sink node and base station; (2) use of wind, temperature, and humidity data, which makes the process less expensive than the solutions using video, in which case the majority are in the literature; (3) constructed using an adaptive approach; (4) has a higher fire detection range and accuracy compared to infrared camera-based approaches.

3. Brings together aspects of self-organization and fault tolerance at different levels. So far it is known in the literature that there are no studies that address together aspects at different levels in WSN for fire detection. The proposed model addresses both these aspects in the routing and the application level.

4. Application of component redundancy technique, in particular, N-Version Programming, whose development was based on components in a sensor network application. Through the consensus of different solutions implemented as components was possible to identify possible evidence of failures in the application level, increasing confidence in the service.

5. Framework Integration FlexFT\textsuperscript{23} to Sinalgo\textsuperscript{22} can be used in the construction of new fault-tolerant solutions in Sinalgo. This goes for the application and routing levels.

6. Classification of Forest Fires Date Set.\textsuperscript{25} Instances were classified as “Fire” and “No Fire” with monitoring and validation of a domain expert. In this case, the manager of a firefighting team.

7. Simulation on the environmental reserve of the Botanical Garden of Brasilia. The simulation took place over an area of 4500 hectares. This enabled to be aware of the number of nodes necessary by the distribution density calculation.

Finally, the following studies and future works are suggested:

1. Within the scope of fire detection application, develop solutions to analyze the side that the fire is growing. This greatly helps the team fight to take action and prevent at risk to human life.

2. To develop other applications using the model, such as flood monitoring applications, pollution, and various applications of smart cities.

3. Using other data mining models and artificial intelligence in our detection model, such as Bayesian networks and neural networks.

4. Building an application server side, to analyze statistical data to assist in the development of new applications and facilitate the management of environmental preservation.
Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has been financially supported by CAPES, and FAPESP (São Paulo State Research Foundation) under grant no. 2015/21642-6.

References

1. INPE. Monitoramento dos Focos Ativos no Brasil, 2015, http://www.inpe.br/queimadas/estatisticas.php (accessed 21 January 2016).
2. Loureiro AA, et al. Redes de Sensores sem Fio. In: XII Simpósio Brasileiro de Redes de Computadores, 2003.
3. Garlan D, et al. Rainbow: architecture-based self-adaptation with reusable infrastructure. Computer 2004; 37(10): 46–54.
4. Cheng BH, et al. Software engineering for self-adaptive systems: a research roadmap. In: Cheng BH, et al. (eds) Software engineering for self-adaptive systems (Lecture Notes in Computer Science). Berlin: Springer, 2009, vol. 5525, pp.1–26.
5. Dressler F. Self-organization in sensor and actor networks. Hoboken, NJ: Wiley, 2007.
6. Yates FE (ed.). Self-organizing systems: the emergence of order. New York: Springer, 1987 (copyright Holder: New York: Plenum Press).
7. Camazine S, et al. Self-organization in biological systems. Princeton, NJ: Princeton University Press, 2001.
8. Avizienis A. Design of fault-tolerant computers. In: Proceedings of the November, fall joint computer conference, Anaheim, CA, 14–16 November 1967, pp.733–743, New York: ACM.
9. Avizienis A. The n-version approach to fault-tolerant software. IEEE T Software Eng 1985; SE-11(12): 1491–1501.
10. Randall B and Xu J. The evolution of the recovery block concept. In: Software fault tolerance, trends in software, 1994, pp.1–22.
11. Randall B. System structure for software fault tolerance. IEEE T Software Eng June 1975; SE-1(2): 220–232.
12. Chen D, Tao Z and Ma G. Application of wireless sensor networks for monitoring emergency events. In: WICOM ’08, 4th international conference on wireless communications, networking and mobile computing, Dalian, China, 12–14 October 2008, pp.1–4, New York: IEEE.
13. Tatara K, Lee G and Chong NY. Self-organizing ad-hoc robotic sensor networks based on locally communicative interactions. In: 2011 8th international conference on ubiquitous robots and ambient intelligence (URAI), 23–26 November 2011, pp.485–490, New York: IEEE.
14. Vilela M and Araujo R. RAHMoN: routing algorithm for heterogeneous mobile networks. In: 2012 second Brazilian conference on critical embedded systems (CBSEC), São Paulo, Brazil, 20–25 May 2012, pp.24–29, New York: IEEE.
15. Joshi Y and Younis M. Autonomous recovery from multi-node failure in wireless sensor network. In: 2012 IEEE global communications conference (GLOBECOM), Anaheim, CA, 3–7 December 2012, pp.652–657, New York: IEEE.
16. Villas LA, et al. A reliable and data aggregation aware routing protocol for wireless sensor networks. In: Proceedings of the 12th ACM international conference on modeling, analysis and simulation of wireless and mobile systems (MSWiM ’09), 2009, pp.245–252, New York: ACM, http://doi.acm.org/10.1145/1641804.1641846
17. Hefeeda M and Bagheri M. Wireless sensor networks for early detection of forest fires. In: IEEE international conference on mobile adhoc and sensor systems (MASS 2007), Pisa, 8–11 October 2007, pp.1–6, New York: IEEE.
18. Liu Y, et al. A novel accurate forest fire detection system using wireless sensor networks. In: 2011 seventh international conference on mobile ad-hoc and sensor networks (MSN), 16–18 December 2011, pp.52–59, New York: IEEE.
19. Yoon I, et al. Reliable wildfire monitoring with sparsely deployed wireless sensor networks. In: 2012 IEEE 26th international conference on advanced information networking and applications (AINA), Fukuoka, Japan, 26–29 March 2012, pp.460–466, New York: IEEE.
20. Noh D, et al. Priority-based routing for solar-powered wireless sensor networks. In: 2007 IEEE 2nd international symposium on wireless pervasive computing (ISWPC’07), 5–7 February 2007, IEEE.
21. Bulusu N, et al. Scalable coordination for wireless sensor networks: self-configuring localization systems. In: International symposium on communication theory and applications (ISCTA 2001), Ambleside, 2001, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.183.5854&rep=rep1&type=pdf
22. Sinalgo, 2016. http://disco.ethz.ch/projects/sinalgo/index.html (accessed 20 March 2016).
23. Beder DM, et al. FlexFT: a generic framework for developing fault-tolerant applications in the sensor web. Int J Distrib Sens Syst 2013; 2013: 358592.
24. Coulson G, et al. A generic component model for building systems software. ACM Trans Comput Syst 2008; 26(1): 1–42.
25. Lichman M. UCI machine learning repository, 2013, http://archive.ics.uci.edu/ml
26. Hall M, Frank E, Holmes G, et al. UCI machine learning repository, 2013, http://archive.ics.uci.edu/ml