Des Autom Embed Syst (2016) 20:269–287
DOI 10.1007/s10617-016-9177-1

Application modeling for performance evaluation on event-triggered wireless sensor networks

Lisane Brisolara[1] · Paulo R. Ferreira Jr.[1] · Leandro Soares Indrusiak[2]

Received: 19 February 2016 / Accepted: 8 August 2016 / Published online: 11 August 2016
© Springer Science+Business Media New York 2016

Abstract This paper presents an approach for event-triggered wireless sensor network (WSN) application modeling, aiming to evaluate the performance of WSN configurations with regards to metrics that are meaningful to specific application domains and respective end-users. It combines application, environment-generated workload and computing/communication infrastructure within a high-level modeling simulation framework, and includes modeling primitives to represent different kind of events based on different probabilities distributions. Such primitives help end-users to characterize their application workload to capture realistic scenarios. This characterization allows the performance evaluation of specific WSN configurations, including dynamic management techniques as load balancing. Extensive experimental work shows that the proposed approach is effective in verifying whether a given WSN configuration can fulfill non-functional application requirements, such as identifying the application behavior that can lead a WSN to a break point after which it cannot further maintain these requirements. Furthermore, through these experiments, we discuss the impact of different distribution probabilities to model temporal and spatial aspects of the workload on WSNs performance, considering the adoption of dynamic and decentralized load balancing approaches.

Keywords Sensor networks · High-level simulation · Application · Modeling · Load · Workload characterization · Load balancing

✉ Lisane Brisolara
lisane@inf.ufpel.edu.br

Paulo R. Ferreira Jr.
paulo@inf.ufpel.edu.br

Leandro Soares Indrusiak
lsi@cs.york.ac.uk

1 Centro de Desenvolvimento tecnológico (CDTEC), Universidade Federal de Pelotas (UFPel), Pelotas, Brazil

2 Department of Computer Science, University of York, York, UK

1 Introduction

Wireless Sensor Networks (WSN) are used to monitor some environment condition and basically can work in two different ways: proactive and reactive [4]. Following a proactive way, the sensor nodes periodically sense the environment and send each new value to a gateway. Such periodic behavior generates several packages and transmissions that could be avoided in certain applications [30]. Considering that end-users are only interested in hearing from the network when certain events occur [29,30], event-driven or reactive applications can be designed.

Sophisticated reactive WSN applications have complex requirements, demanding that the application experts are also involved in its design [27]. In the end-user point of view, reactive applications can be abstractly specified as triggered events that are generated by the environment and must be handled by the system. As these professionals are not trained to specify computational applications in detail, the application should be also abstractly specified [26].

Usually, application-specific WSN must be designed to meet non-functional application requirements. As an embedded system, WSNs have limited resources (energy source, memory, processing capability) and additionally, the WSNs designers should define required redundancy and number of nodes to achieve desired coverage area, as well as determine network topology, which significantly impact on their design [35]. As large-scale and complex networked systems, its design requires several decisions, which influence systems efficiency, costs, and coverage [9]. A critical design aspect is the evaluation of a WSN configuration when running a given application, which cannot be based on prototypes or low-level simulations that are costly and time consuming.

WSNs are used to monitor real environments, where different events can happen at different times and at different places. Thus, to evaluate a WSN configuration, several application scenarios should be considered varying the load distribution in the network area and the frequency of events using the appropriated probabilistic models [10]. The definition of these scenarios is very important in the evaluation of the network efficiency as well as in the evaluation of dynamic management strategies. In the embedded systems domain, high-level modeling primitives have been used to model complex applications, abstracting details [18], and enabling early systems behavior evaluations.

As highlighted in [5], the physical environment plays a fundamental role especially when the energy consumption of WSNs is considered. Considering the event-triggered applications, the computational load is also defined by the events detected from the environment. In this context, the application workload specification should be centered on the description of these events, when and where they happen.

On event-driven WSN, a relevant dynamic management strategy is load balancing, since commonly network solutions include redundancy to achieve desired coverage area. As more than one node can sense the same event, load balancing techniques can improve the network lifetime and service availability avoiding that the same event be processed by more than one node [3,15]. Temporal and spatial aspects of event-triggered WSN applications can impact significantly in the efficiency of load balancing techniques, which are not widely explored in previous work.

Following [21], several works have contributed to facilitate the development of WSN applications using high-level abstractions, models and simulation frameworks. However, these works address challenges on WSN programming and code generation, or on WSN simulations focusing mainly on network aspects [11,19,24]. Moreover, the mentioned and

several others simulators are centered on node, which means that the application are composed of simple periodic tasks (sensor reading and send messages, e.g.) programmed to run on each node. This approach is suited for periodic (or proactive) applications, where the nodes are only responsible to read values and send to the gateway. However, to support event-triggered (reactive) applications, the cooperation with environment simulators is necessary to generate the load, turning the whole simulation more complex.

Here, we propose primitives for application load modeling, addressing temporal and spatial aspects and targeting high-level simulation of event-triggered WSN. Such primitives help end-users to characterize their application workload based on probabilistic models and check the performance of specific WSN configurations when running realistic scenarios. Experiments explore different aspects of events and show how they impact on network lifetime and service availability, evaluating scenarios without and with dynamic load balancing techniques. Thus, our experimental results also shows the impact of these aspects on the efficiency of state-of-the-art load balancing algorithms [3,8].

The paper is organized as follows: Sect. 2 discusses previous work on WSN simulation. Section 3 presents the state-of-the-art on load balancing for event-triggered WSNs. Section 4 introduces the proposed approach. Section 5 presents the experimental work and respective analysis. Section 6 concludes and points out directions for future work.

2 Related work

2.1 Modeling and simulation

Several works have addressed WSN simulation [11], resulting in simulators or frameworks like Semsim [24], TOSSIM [19], J-Sim [33] and TikTak [20]. The most of these focus on the WSN hardware simulation or do not yet consider event-triggered applications. In such works, application tasks are often periodic and simulated over a single node, or they are not modeled at all and only the low-level aspects of the network protocols and radio channel are taken into account.

Furthermore, there is little support to the modeling of realistic scenarios where the application load varies according to physical and natural phenomena models (with a few exceptions based on complex application-specific simulators are used to mimic the environment behavior, as proposed in [14] for fire monitoring). Even most recent works that focus on WSN performance analysis adopt workload based on uniformly distributed random variables [3]. This simplification is not suitable for all natural phenomena and for event-triggered applications, where appropriated probabilistic models should be used to determine the event frequency and position. Recently, probabilistic models were used in [10] to represent the event frequency for dynamic data storage estimation.

When evaluating an application whose objective is the monitoring of movement (e.g. animal tracking, ocean currents, and gas leak) the movement pattern should be also considered [31]. For instance, an animal mobility model based on probabilistic behavior was proposed in [23] for wildlife tracking and monitoring applications. However, this model does not consider all aspects of the WSN application load (such as software tasks used to process data generated by the environment). Similar to [10,23], our work proposes modeling primitives to facilitate the adoption of probabilistic distributions to describe all aspects of an event.

As part of Ptolemy II [2], VisualSense [28] is a framework for high-level WSN modeling and simulation. Although VisualSense primitives focus mainly on platform modeling, it is a

relevant effort since can be easily extended. We extend the VisualSense, including application modeling primitives based on probabilistic models in order to support the performance evaluation of WSN when running realistic scenarios.

This paper expands upon [1], where the impact of temporal and spatial aspects from event-triggered applications on WSNs was discussed. That work was the first to integrate probabilistic application load characterisation into WSN simulation, but without consider any load balancing mechanism. Here, we explore this integration to better analyse the efficiency of dynamic load balancing strategies for event-triggered WSNs. Thus, several new experiments were conducted with the proposed application workload model combined with two prominent distributed dynamic load balancing techniques [3,8].

2.2 Load balancing

Works commonly faces the load balancing in WSN as a problem of communication among nodes, presenting methods to better routing the data traffic through the network [6,13,17,22]. It happens because they consider that the network senses the environment in a proactive way, which means, all nodes have the same tasks to process. We focus here on reactive WSNs where the events trigger the network which generates tasks to be processed by the nodes.

When targeting event-triggered WSN applications, the load balancing should explore the sensor redundancy since several nodes can be triggered by the same event but only one has to process the tasks it produces. The problem of task mapping on distributed systems has investigated, considering different platforms as multi-cores, and as well as sensor networks. The approaches for task mapping can be divided in static as in [25,36] and dynamic as in [3, 8,15,32]. Static approaches are limited since cannot effectively adapt to network conditions.

Different approaches have proposed for dynamic WSN load balancing, which mainly differs among each other from the adopted heuristics like genetic algorithm [15] and bio-inspired [32]. In these works, centralized solutions have proposed to balance energy usage while extending the network lifetime.

Moreover, recently works have focused on distributed approaches given the distributed nature of WSN. Ferreira et al. [8] and Ipek et al. [3] have proposed different distributed dynamic solutions for WSN load balancing based on social insects colonies behavior. In [3], an interaction process is used to determine which node will process a sensed event inspired in queens differentiation mechanism through signal pheromones. In [8], nodes mimic ants division of work behavior to dynamically choose nodes that will be responsible to process sensed events. As said before, here we have evaluated these two most recent and prominent dynamic load balancing approaches.

3 Distributed dynamic load balancing in WSNs

In WSN, the network lifetime or its service availability depends on the discharge of nodes batteries. As WSN solutions include redundancy to achieve desired coverage area, more than one node sense the same area at same time. This redundancy motivates the adoption of dynamic coordination mechanisms to achieve load balancing, saving energy and consequently improving the network lifetime. When targeting event-triggered WSN applications, the load balancing can be more useful, since several nodes can be triggered by the same event.

The ant-based approach [3] aims to allow WSN nodes to decide individually which event to process triggered by the events emergence on the environment. In this approach, nodes decide probabilistically, on-the-fly, which events to perform applying a decision process

inspired by the ants theoretical model of response threshold. This approachs general idea is to divide the work among the nodes sensing the same area, considering the number of nodes who sensed an event at the same time; and the number of times a given node was previously engaged in events processing.

When an event occurs, it produces stimulus for the nodes sensing the event's location area. The produced stimulus is computed based on the number of neighbors which sensed the event and the maximum number of neighbors that could have sensed the same event. This information is determined by communication, as nodes broadcast to the neighborhood a simple message indicating that an event was sensed. To capture the nodes working history, each one has an internal response threshold that changes over time. This threshold manages the node sensitiveness in such way that nodes become more sensitive to the stimulus of an event it is performing and less sensitive to the next events. Thus, when nodes detect events, they will decide which ones to process according to their probability computed using events stimulus and their internal response threshold.

The Pheromone Signaling (PS) approach [8] is based on the bees hormonal system, which ensures every beehive has only one queen. Through this approach, nodes are periodically differentiated from other nodes to indicate their duties. Some nodes, called Queen Nodes, are allowed to process the sensed events, while remaining ones, called Worker Nodes, stay in stand-by.

Nodes differentiate themselves through a periodic transmission of pheromone by Queen Nodes and its retransmission by recipients to their neighborhood. The retransmission is limited to a number of hops and the amount of pheromone is decreased each time it is propagated. Through this process PS limits the range of influence of each Queen Node, aiming to keep, as much as possible, the coverage of the WSNs sensing area. Nodes became a Queen Node based on their internals threshold and pheromone level. The differentiation happens whether the node does not receive enough pheromone from their neighbors to keep their internal pheromone level above their internal threshold. The internal pheromone level increases over time as the node receives pheromones from their neighbors. To compensate it, the internal pheromone level of all nodes is decreased periodically.

4 Proposed approach

Basically, the design of an application-specific WSN can be divided in two layers: platform and application. When designing an application-specific WSN, the platform should be designed and configured according to the application requirements. For fast and easy design space exploration (DSE), it should be possible to change the platform or its configurations (e.g. number of nodes, network topology, node positions, radio transmitting power) and evaluate the system efficiency when running the target application.

In order to facilitate the evaluation of WSN configurations, we provide high-level primitives to model computation and communication load of reactive WSN applications. As illustrated in Fig. 1, environment aspects are incorporated to the application model in our approach, since that load is generated by events captured from the environment. Events represent the cost of the computation and communication tasks required by the application. Such costs are directly related to the time tasks take to execute, the energy they require, and the platform resources they occupy. These costs are mostly platform-specific, but can be obtained through platform profiling, and then used within a simulation model in order to analyze the efficiency of a given design solution.

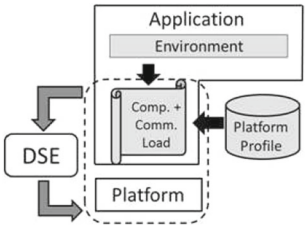


Fig. 1 Proposed approach for event-driven application workload modeling

4.1 Application modeling

The proposed application modeling is focused on non-functional aspects of the application, so the computation tasks are abstracted away, and our model only represents the costs (time, energy, resources) associated with them. Our approach differs from others, incorporating temporal and spatial aspects from the environment into the application level in order to better characterize the workload of reactive WSN applications.

WSN reactive applications commonly monitor events, whose types depend on which kind of natural phenomenon the application end-user is interested in observe, as for instance temperature dynamics, humidity, moving objects (as vehicles and animals), among others. Flood monitoring, forest fire detection, intrusion detection, habitat monitoring are examples of applications with different phenomena of interest. In a fire detection application, sensors read temperature, humidity, and smoke in order to determine if there is a fire event or not. Considering the habitat monitoring, an application subdomain is the movement tracking of animals in a given monitoring area, where the interesting event is the movement. In this case, end-users want to know when the animal is moving, in which direction, and yet in which frequency it changes its position.

An event occurs into the environment in a given spatial location, in a given time, and is related to a natural phenomenon (e.g. temperature, humidity, sound, and light). Events are only sensed by a given sensor node if this node is active at the events time, the event occurs in its sensing area and if the event type is supported by this node. Thus, to represent events, three aspects are relevant, which are: spatial, temporal, and functionality. Spatial is related to events location, temporal is related to the time when this event occurs, while functionality indicates the type of sensor enabled to capture this phenomenon. Varying and combining these three aspects, one can define different events covering the most interesting natural phenomena. For example, we define that an event is static if there is no spatial variation, and atomic when it occurs in a specific time (without temporal variation) and is not repeated. A periodic event is that recurs at intervals.

Uncertainty and variability are present in the modeling of natural phenomena [34], which motivates the use of probabilistic models to model these phenomena. In this work, we propose to represent WSN interesting events using stochastic models, which are built through field observation of the correspondent natural phenomenon by the end-user. These stochastic models determine the probability of a given event occur at a given time and at a specific position to simulate the spatial and temporal variations from the real-world events. These models can be represented using histograms or spectrograms and should be loaded into the simulator.

Based on this event classification and definitions mentioned before, we propose high-level primitives to model events in a WSN reactive application. Figure 2 illustrates the proposed

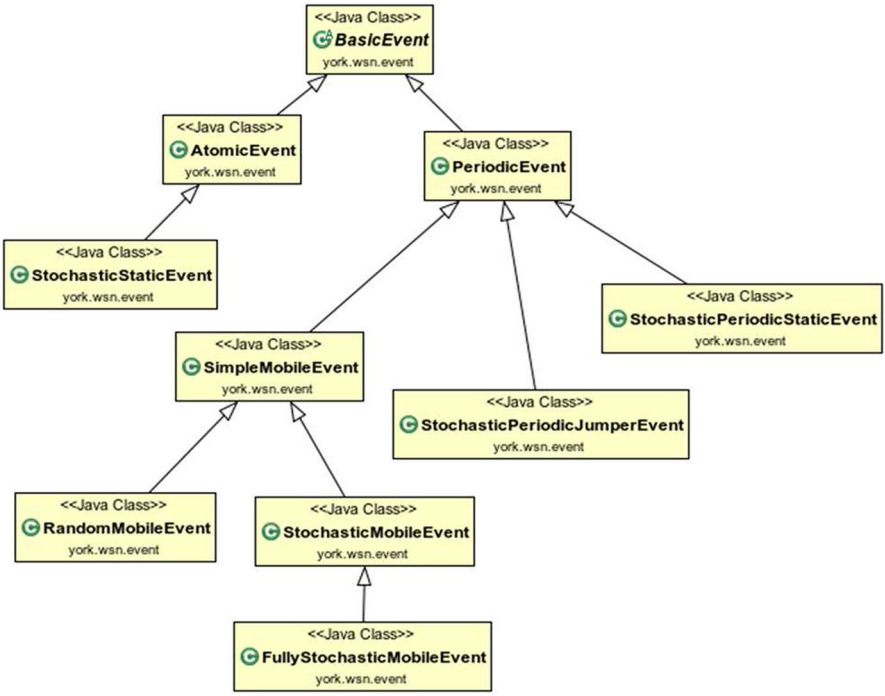


Fig. 2 Proposed primitives for application modeling

primitives organized in a class hierarchy like a metamodel. This hierarchy addresses the temporal, spatial and functionality aspects from events. On the top of this hierarchy is the *BasicEvent*, which is an abstract class that generalizes our concept of event, defining the events attributes (e.g. position, and trigger time). To address the functionality aspect, any event has a type represented by the *EventType* class.

BasicEvent is specialized by *AtomicEvent* and *PeriodicEvent* classes, which represent atomic and periodic events, respectively. An atomic event is static for nature in our hierarchy, but we propose to specialize it to represent an atomic static event whose position and trigger time are determined by a stochastic model, which is named as *StochasticStaticEvent*. Stochastic models can use any probability distribution, and be depicted by a histogram (2D variables) or a spectrogram (3D variables). A periodic event is that recurs at intervals, thus the *PeriodicEvent* class has an extra attribute named period, which represents the duration of one cycle in a repeating event.

PeriodicEvent can be used to represent static or mobile events. The *StochasticPeriodicStaticEvent* subclass represents a recurrent static event. When an event changes position, it is modeled using the *SimpleMobileEvent* or some of its subclasses. The *StochasticPeriodicJumperEvent* can be used to model a big set of non-simultaneous static events whose spatial distribution can be defined by stochastic models, accelerating the simulation.

A mobile event is generalized by the *SimpleMobileEvent* class, which defines attributes to describe the movement parameters, such as direction, speed and time between changes. Such attributes are used to determine the new position for a mobile event, and can be also represented by probability distributions, for instance using *StochasticMobileEvent* or *FullyStochasticMobileEvent* primitives. *StochasticMobileEvent* has initial position, direc-

tion, speed, and time between direction changes determined by stochastic models, but trigger time fixed in zero, which means that the event starts on the beginning of simulation. In *FullyStochasticMobileEvent* all these parameters including triggertime are based on stochastic models. The variability of these parameters is represented by spectrograms and histograms, which define the probability distributions. For example, when representing events associated with birds monitoring, a 3D Normal function spectrogram can express the presence of young birds around the nest. A histogram with a Poisson function can represent for example the probability of each one move at the interval of 1 h. The data represented by these curves are used to raffle a new value for the parameters during the simulation. For the experiments ran in this paper, we defined also the *RandomMobileEvent*. This subclass is a simplified version of *SimpleMobileEvent*, where the event new position is given by a small random decrease or increase on its current position.

As referenced before, an event is the basis of our approach for modeling WSN reactive applications load. Using the proposed primitives is possible to represent these events, covering the three main aspects. Although to evaluate efficiency, another aspect should be specified that is related to the cost to process and communicate this event. The event computation load can be yet represented as a graph of tasks, where each task can have a different cost. These costs are platform-dependent and will be discussed in Sect. 4.2.

4.2 Platform modeling and simulation

To simulate and observe the behavior of an application-specific WSN, we also defined a high-abstraction platform model for timing and energy consumption evaluation. Our platform model is composed of nodes connected by a wireless communication channel with limited range to form a given network topology. Nodes can be sensors nodes, intermediate nodes or sink nodes. Sensor nodes have a limited sensing area, which defines whether they can be affected by a specific environmental event or not. According to a sink-to-gateway-based architecture [26], a sink is an intermediate node among the gateway and the wireless sensor network itself. This sink is connected on the same radio channel as the sensor nodes.

Nodes are distributed across the area of interest and end-to-end connections among then are built, respecting the channel range and considering a way to every node to achieve the sink in a multi-hop fashion. Such information is then stored locally by each node in routing tables. If a node is inactive or dead (e.g. faulty, out of battery), connections can be dynamically rebuilt, finding another way to connect nodes avoiding inactive nodes. Our approach is flexible enough to model many routing algorithms and rerouting policies.

When an event is captured by a node, its corresponding processing tasks are executed and a message is sent to the sink. To simulate this processing, our sensor nodes have a CPU, which abstractly models the execution of tasks. Currently, we assume a FIFO-scheduled CPU, but it would be trivial to support e.g. round-robin, priority preemptive or non-preemptive schedulers. As mentioned before, an event can be associated to a graph of tasks and each task can have different costs. Such costs then determine for how long a CPU is busy and how much energy it dissipates while processing a given task. Each node has a gateway, which determines its way to achieve the sink. Thus, if the node A captures an event, it sends a message to node B, its gateway, that will forward it for its gateway, until achieve the sink. Every event-detection triggers one or several message transmissions, depending on the sensor position and its distance to the sink.

Both sensor and sink nodes have a source of energy (in the current model, a non-rechargeable battery) and parameters are used to model nodes initial and current energy levels. To simulate the nodes battery discharging, during the simulation, sensors and sinks

will have their current battery recalculated according to their activities. This discharging process is based on CPU cycles and costs defined for each operation mode. A node can be operating in the modes idle, processing or communicating, where idle means also sensing without any processing, assuming that the components responsible for sensing are always working. We assume that the reception cost for the radio is included into the idle cost. The communication cost is related only to the send message operation, thus only the node that send the message discharges its battery, but a message that take several hopes cost for every node that forward it.

In order to simulate an application running in different platforms, one can change the number of sensor nodes, the number of sinks, the position of the all nodes, the range of channels, and dimension of the interesting area. For fast simulation, we adopted a discrete-events based approach, and in this case, the mentioned battery discharging occurs only when some event is triggered, accelerating the simulation. Alternatively, the user can adopt a simulation based on wall-clock time. In this case, the simulation cycle will respect the wall-clock, facilitating the behavior visualization graphically, but the simulation will run slower than in the discrete-event based one.

The approach described in this paper was implemented as EBORACUM [7], an open-source extension of Ptolemy II and VisualSense.

5 Experiments

5.1 Hypothesis

The energy consumption in the WSN and consequently its lifetime and service availability depend on the application workload, which means that depending on events frequency and their position, the WSN service availability will vary. Commonly the WSNs are evaluated using hypothetical workload based on uniform distribution [14]. However, as our experiments will show, the use of different probability distributions to model temporal and spatial aspects of the load can enormously impact on the network service availability and also affect the efficiency of load balancing strategies.

In the experiments, we demonstrate it varying the distribution for event spatial aspects (Uniform, 3D Bell curve, 3D Inverted Bell curve), temporal aspects (Uniform, Normal and Poisson), and also modifying the movement pattern. Firstly, we are discussing how it impacts on the network service availability without consider any load balancing strategy. Furthermore, we discuss also the after-effects of the same probabilities models on the efficiency of ant-based and pheromone signaling load balancing algorithms.

5.2 Experimental setup

The proposed modeling primitive hierarchy was built based on Ptolemy II and all concepts are derived from the TypedAtomicActor. Our approach facilitates the generation of simulation parameterized scenarios which are loaded on Ptolemy II GUI through a benchmark generator written in Java. This generator was used to validate our primitives and to automatically generate the experimental scenarios. A scenario describes an application-specific WSN that can be based on a real-life use-case or artificial test environments. The network behavior can be analyzed by data reports or graphically on Ptolemy II GUI.

In our experiments, we consider a simulation model composed of 49 sensor nodes uniformly scattered (or 7×7 mesh) through a square 810 km^2 area (900 m $\times$ 900 m) and one

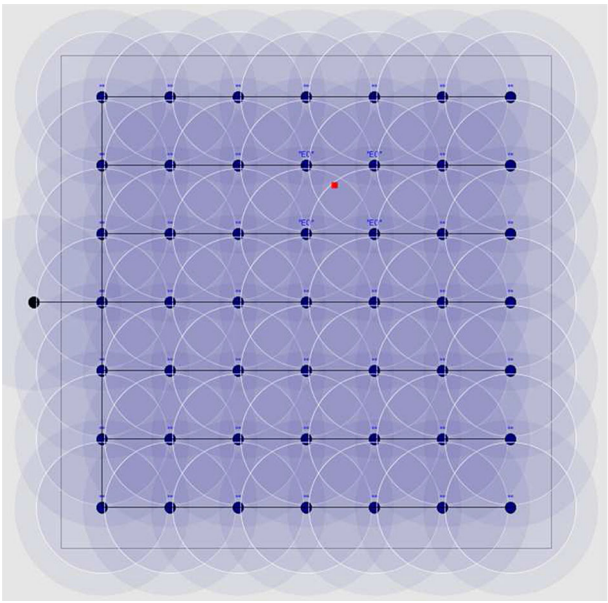


Fig. 3 Screenshot of the simulation model

sink located on the side, as shown in Fig. 3. Nodes are depicted as small blue circles. Nodes sensing and communication radius are depicted as the translucent greater circles with (120 m) and without a white border (160 m), respectively. Communication paths are depicted by the black lines and the area of interest is defined by the square enveloping the mesh. The sink is the small black circle out of the area of interest. As we will discuss further, it is important to remark that there are 24 nodes in the borders with part of its sensing area outside of the area of interest, 4 of them in the corners. Corner nodes have almost twice the area outside the area of interest than the other border ones. The red square depicts an event sensed by 4 nodes simultaneously as it happens inside these nodes' sensing radius.

The nodes are connected to one of its four neighbors (the closest to the sink) or directly to the sink. Nodes are spatially distributed such that 100 % coverage of the area of interest is achieved. This configures a network with limited density or overlapping, once any occurring event can be detected by more than one sensor, but at most by four sensors (with a lower probability). Unless you have specific information to the contrary, the adopted event spatial distribution is the Uniform one. For the first two experiments, our *StochasticPeriodicJumperEvent* is adopted and in the last one, *RandomMobileEvent* and *FullyStochasticMobileEvent* are used.

Each sensed event generates a processing workload equivalent to 14 tasks and the sent messages have 3 Bytes. We adopted this configuration to simulate a network of nodes equipped with acoustic sensors to capture sounds generated by the entities under surveillance (usually animals) on a particular area [3]. The adopted energy costs and also the battery capacity are defined in Table 1, according to IRIS Motes datasheet [12]. We assume that the battery discharge is linear and when the battery is in the half of its charge, the node stops to work.

For all experiments, the same configuration was executed 30 times and average results are used in the comparisons (t tests were run with 0.95 of confidence). The simulator ran in a i7, 8GB of RAM, ordinary PC.

Table 1 Energy costs

Energy-related parameter	Value
Battery capacity	5,400,000 mAs
Idle discharge rate	0.3 mAs
Task computation discharge rate	3.57 mAs
Discharge rate per message (3 bytes) at 30 kbps	0.0018 mAs

5.3 Experimental results

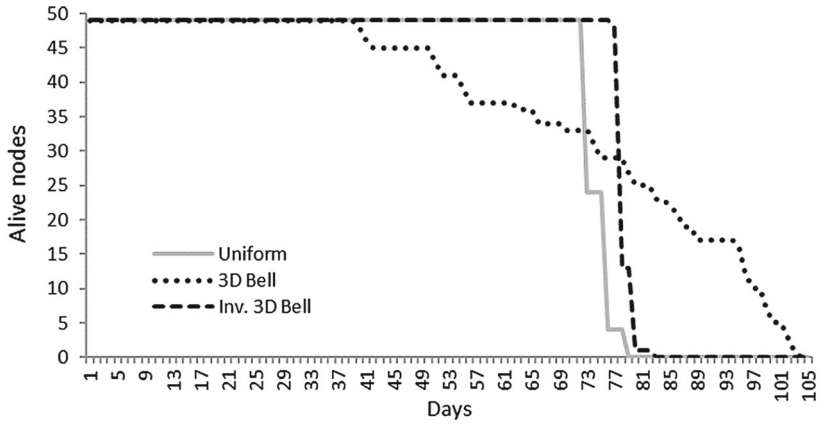
5.3.1 Spatial

In this section, the variation of spatial events distribution is evaluated, considering Uniform, 3D Bell curve and Inverted 3D Bell. Regarding temporal distribution, a Poisson function with an interval between 1 and 100 s was adopted. Firstly, we discuss the impact on a WSN without load balancing strategy. Figure 4a depicts the results for alive nodes over time in this scenario.

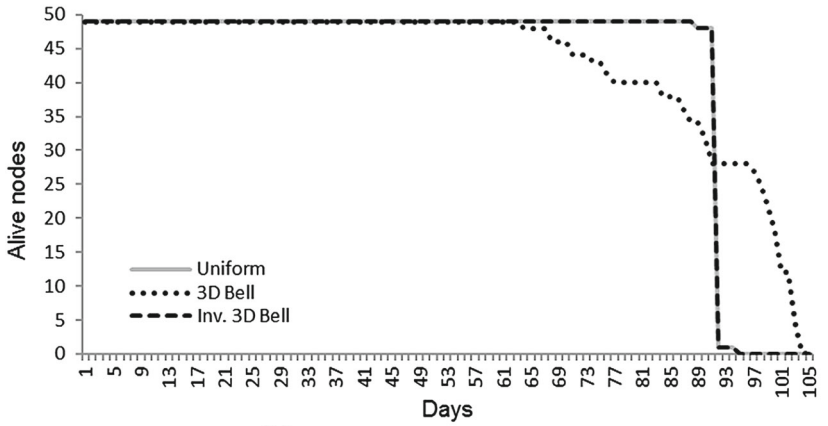
As can be observed in Fig. 3, the nodes at the borders of the area of interest have lower probability to detect events and preserve the battery for more time, since part of its sensing areas are outside of the area where events occur. This effect is even more visible in the four corner nodes. Thus, when a Uniform distribution is used, three steps can be observed in the curve, once the border nodes will detect fewer events and will be alive for longer time. The last step shows the mentioned behavior of the four corner nodes. Using a 3D Inverted Bell curve, the border nodes have higher probability of detect events, reducing this differentiation and consequently, all nodes died in closed times. Finally, when a 3D Bell curve is used, the events have high probability of occur in the central area, decreasing to the borders. As a result, there is a significant variation in the frequency of events stimulating different nodes, and reinforcing the border effect. There is a wider spread on the nodes' energy depletion time, and an overall increase in network lifetime.

Results shows also that when the event spatial distribution respects a 3D Bell curve, the network lifetime is higher than for the other both distributions on average around 26 and 32 % compared to 3D Inverted Bell curve and Uniform, respectively, in the conducted experiments. Moreover, as expected, we observe larger average standard errors when using Uniform distribution (0.40), compared to 3D Bell (0.06) and Inverted 3D Bell (0.07). These errors are not depicted in Fig. 3 because of the y-axis scale. However, with Uniform and 3D Inverted Bell curves is possible to say that the network fails around the day 80. While using the 3D Bell curve, nodes start to die in half of this time, compromising the network efficacy after this and demanding a different design strategy. Thus, we have shown that the network behavior regarding service availability significantly differs depending on the event spatial distribution.

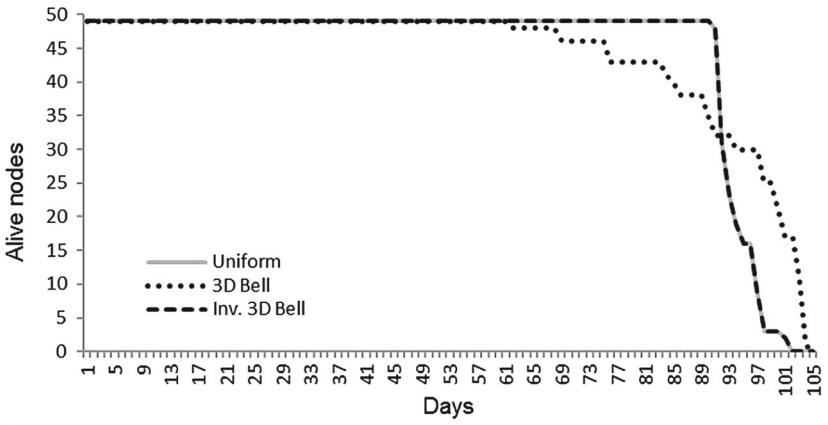
In addition, we have analyze the effects of these distributions on the Ant-based and PS load balancing algorithms. Figure 4b, c illustrates the results for alive nodes in the scenarios with the three different distributions achieved by PS and Ant-based algorithms, respectively. Firstly, we observe that for both algorithms, the curve of alive nodes using Uniform and Inverted 3D Bell are overlapped. The behavior of the network without any load balancing was already similar in these two distributions and it was intensified in the load balanced networks.



(a) Without load balancing



(b) Pheromone Signaling approach



(c) Ant-based approach

Fig. 4 Results for different spatial distributions

Table 2 Sensed events according different spacial distributions

	PS	Ant-based	No load balancing
Uniform	214850.0	270383.2	255283.2
3D bell	201225.6	230958.6	201353.0
Inv. 3D bell	214874.8	270381.2	255274.8

Comparing yet the number of alive nodes by days achieved by the non-balanced network, both load balancing algorithms improve these numbers, and keep nodes alive for longer times (around 10 days more for Uniform and Inverted 3D Bell distributions). Using the 3D Bell distribution, we observe that PS and Ant-based algorithm improve also the network efficiency, since these algorithms can retard the death of the first node in around 30 days compared to a non-balanced network.

Similar to the behavior already observed into non-balanced network, both algorithms achieved a smooth curve when using 3D Bell, compared to the others distributions. The smoothness of the decreasing curve is higher with non-balancing, medium for Ant-based and is lower in PS. As discussed before, 3D Bell intensifies the difference between the frequency of events stimulating different nodes. This results in a more significant difference between the times each node's battery is completely depleted, in all three scenarios.

However, to better analyze the efficiency of load balancing algorithms, it is important to observe also the total number of sensed events and not only the number of alive nodes. This information considers that an event is given as sensed if it is processed by a node and has its result received by the sink, which means that a multi-hop way to the sink should be provided by the network. Table 2 resumes the total number of sensed events achieved by the three approaches varying the probability distributions. These numbers again reinforce the similarities observed between results achieved for each approach using Uniform and Inverted 3D Bell distributions and the differentiation when a 3D Bell is used. In a comparison of Ant-based algorithm against to a non-balanced network, Ant-based achieves around 6 % of improvement using Uniform and Inverted 3D Bell distributions, and 15 % of improvement using a 3D Bell distribution. Moreover, these results highlight that a comparison among strategies for event-triggered WSN cannot be done based only in a single scenario (using a single probability distribution), except when a specialist guarantee that this distribution is appropriate for the application.

5.3.2 Temporal

To demonstrate the impact of the event temporal distribution, we conduct experiments using Uniform, Poisson, and Normal (2D Bell curve) distributions. Usually, the Uniform distributions are adopted by simulators when using simple random functions. We choose a Normal distribution, when events occur in the middle of the interval with higher probability, and Poisson which is applied into simulation of various phenomena with discrete properties.

In a first experiment without considering load balancing approaches, we vary the function interval between events from [1,10] to [1,10000] s to observe the impacts on WSN lifetime using lower and higher events frequency. For instance, using Normal distribution, in the first interval, events occur on average at each 5 s and in the last one, at each 1 h and 23 min. The values of the random variables produced by Poisson distribution is 1/4 of the time interval, instead of 1/2 in the others distributions. Consequently, as expected, the statistical analysis pointed out that there is no significant difference between the network average

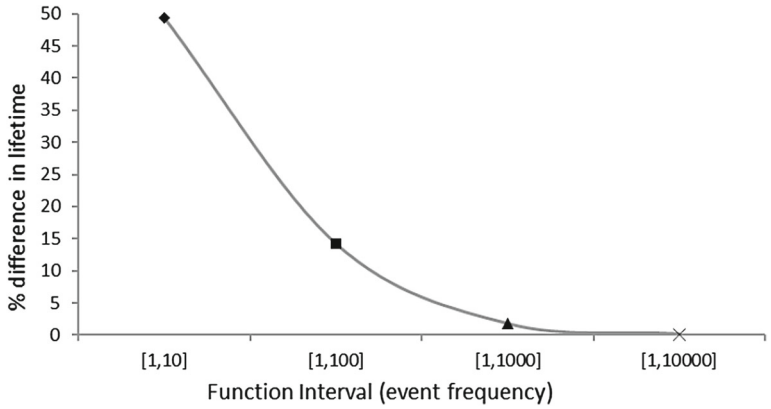


Fig. 5 Proportional difference in lifetime between Poisson and the other distributions for different function intervals

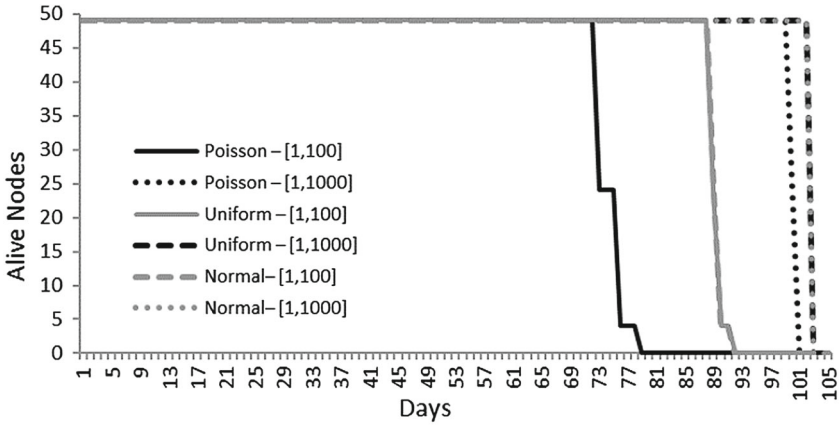
lifetime with Normal and Uniform distributions. However, when Poisson distribution is used, significant differences are found in all cases, with different magnitudes depending on the events frequency. For the higher frequency, the amount of generated events is higher and in consequence the lifetime is reduced on average 23.8 days (49 %) using Poisson distribution. For smaller frequencies, the Poisson distribution reduces on average the network lifetime in only 1.8 days (0.018 %) with the function interval of [1,1000] or yet 4.4 h (0.0018 %) when the interval is [1,10000]. As depicted in Fig. 5, despite the size of the interval increases linearly, the proportional difference in lifetime between Poisson and the other distributions decrease exponentially.

Figure 6a depicts in details the number of alive nodes by day for the three probability distributions varying the intervals between [1,100] and [1,1000] without load balancing strategies. When the interval [1,1000] is used the lifetime achieves almost its maximum (104 days) due to the low number of generated events. However, in the interval [1,100], the network break out around the day 88 because the increased number of events. One can observe that this break out is yet earlier, in the day 72, when the Poisson distribution is adopted. The results for the intervals [1,10000] and [1,10] are omitted in this plot. The first one because there is no significant difference among the results obtained with the different distributions. The second was omitted for the sake of comparison to the WSN with load balancing approaches, in which the simulation time becomes too long.

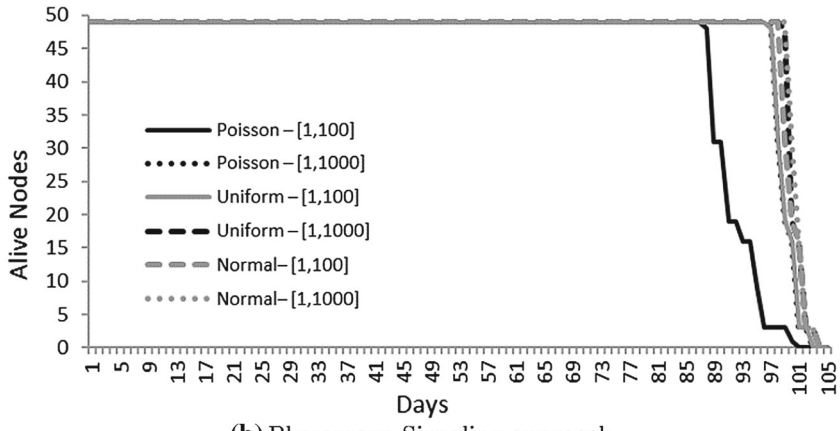
Figure 6b, c illustrates the number of alive nodes in the WSN using the PS and Ant-based algorithms for load balancing, respectively. These results show that both algorithms improve significantly the WSN lifetime using the interval [1,100] for all distributions. However, when the interval [1,1000] is adopted, the load balancing approaches reduce the network lifetime around 5 %. This result indicates that the overhead of the dynamic management execution is larger than the improvements that they could achieve in this scenario. For example, if the WSN is monitoring an environment, the load balancing algorithms should be applied only when the frequency of events emergence is necessarily larger than 1 at each 16 min.

5.3.3 Movement pattern

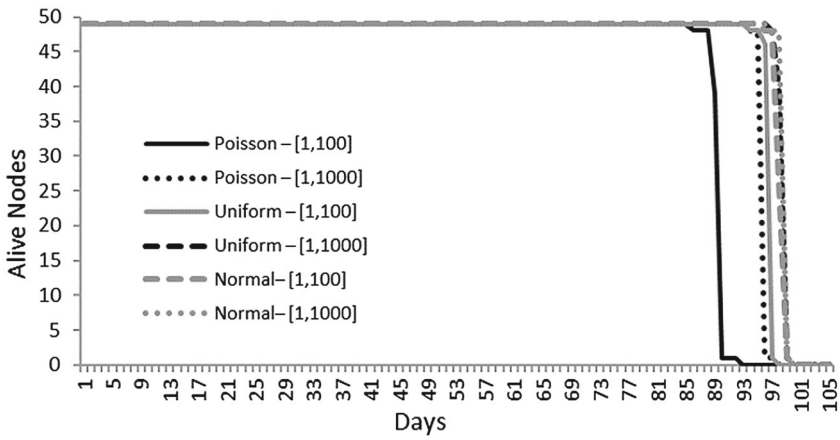
Finally, to observe the impact of the events movement pattern into the network lifetime and in the efficiency of load balancing algorithms, we conduct experiments to compare random



(a) Without load balancing

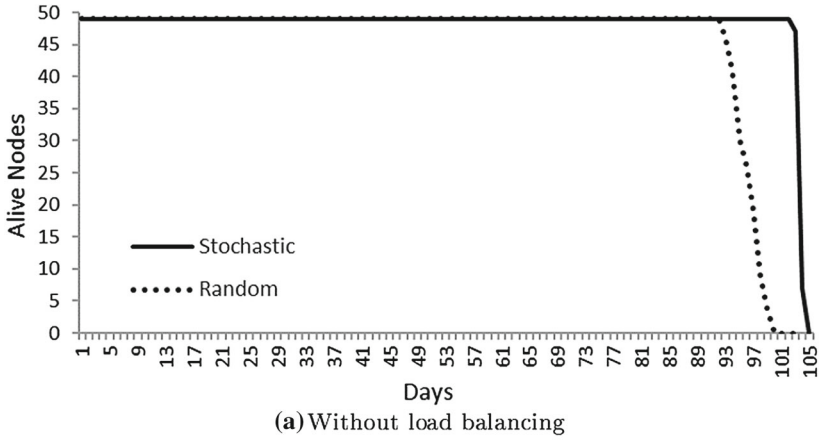


(b) Pheromone Signaling approach

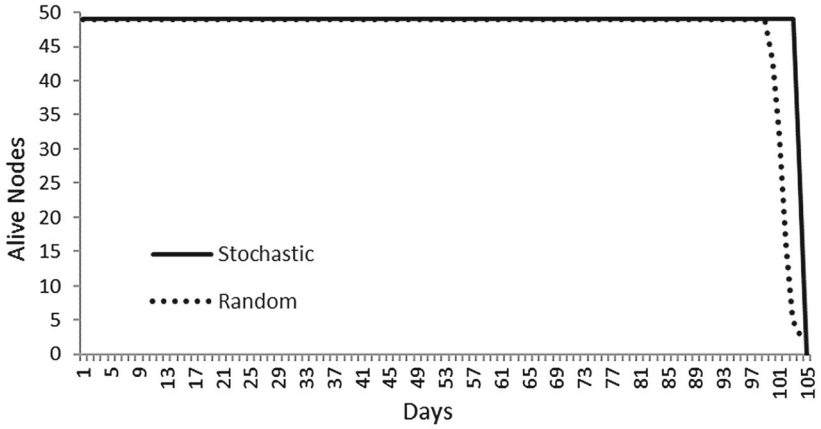


(c) Ant-based approach

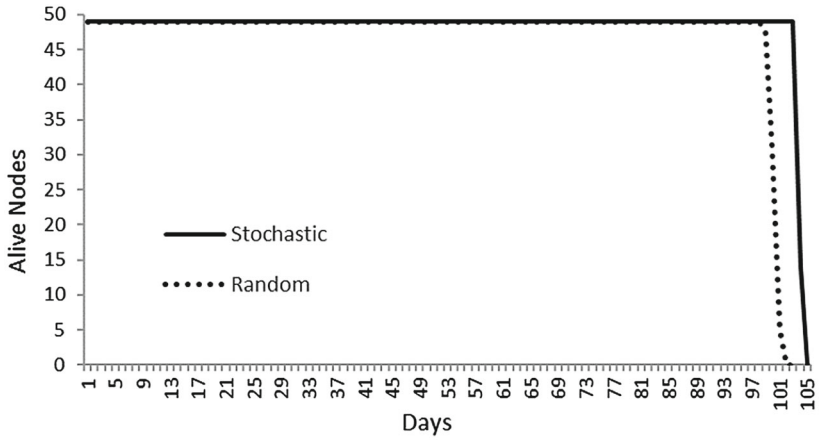
Fig. 6 Results for different temporal distributions



(a) Without load balancing



(b) Pheromone Signaling approach



(c) Ant-based approach

Fig. 7 Results for different movement patterns

and stochastic movement patterns using 100 mobile objects produced uniformly during the simulation time, which generate several events in the network. As mentioned before, our movement patterns use direction, time between direction changes and speed as parameters. In the stochastic model, such parameters are determined by a Normal function using intervals of 0 to 360, 0 to 10, and 0 to 20, respectively. In the random model, the implemented behavior mimics direction and speed changes in the same intervals but following a Uniform distribution.

Figure 7a illustrates achieved results for a non-balanced WSN. It points out that mobile objects moving according to stochastic models induce lower energy consumption by the network, enabling the network to be alive for more days compared to random movement models. This difference represents around 5 days of lifetime, and the first nodes begin to die 10 days before using the random model.

The impact of the adopted movement model into a dynamically balanced WSN can be seen in Fig. 7b, c. Results point out that load balancing approaches cannot improve the network lifetime when stochastic movement is adopted. As using the stochastic model the objects tends to keep the movement direction, objects leave the network area faster than when the random movement is used. It causes a reduction on the number of events that will be processed by the network, and thus, reducing also the opportunities for load balancing. However, when the random model is used, both algorithms improve the network lifetime by 5 %, delaying the energy depletion of the first node in 7 days. In this experiment, the number of sensed events does not aggregate value to the discussion conducted here because all nodes had their batteries depleted almost at the same time.

In [1], 1000 mobile objects were used in a non-balanced WSN and we show that, the adoption of a stochastic model enables the network to be alive for more days compared to random movement model. This difference represents around 30 days of lifetime, which means 35 %. Comparing these previous results to the one discussed here, one can observe that the stochastic model presents a similar behavior even increasing the number of objects and does not offer opportunities for load balancing.

As each mobile object produces an event in short intervals of time, the increasing in the number of mobile objects drives our event-discrete simulator to almost real-time synchronization. Furthermore, dynamic load balancing techniques produce an overhead of computation. The Ant-based algorithm, for instance, runs each time an event is produced in the environment. Thus, we have chosen do not run the load balancing algorithms with 1000 mobile objects due the required simulation time.

It pointed out that when the WSN objective is to monitoring wildlife, the usage of a random-based object mobility achieves pessimistic results, while a stochastic based model can achieve more realistic results, once the field observations usually are represented as stochastic models [16]. It is possible to conclude that load balancing implies no advantages in this kind of realistic situation.

6 Conclusions

We have proposed an approach to model event-triggered WSN applications aiming to evaluate the performance of WSN configurations. Events are the basis of the proposed approach, enabling the simulation of different natural phenomena through probabilistic models. Following this approach, end-users may capture a variety of application specific scenarios. Thus, our load modeling approach enables end-users (possibly without any programming skills)

to conduct experiments and evaluate WSN configurations as well as dynamic management strategies for load balancing, when running real-life scenarios of their event-triggered applications. We have demonstrated with experiments that the nature of the load has a significant impact on the WSN efficiency, mainly when load balancing strategies are considered and thus cannot be neglected by WSN simulators. As future work, we plan to extend our primitive hierarchy to support events triggered or created by other events.

References

1. Brisolara L, Ferreira P, Soares Indrusiak L (2015) Impact of temporal and spatial application modeling on event-triggered wireless sensor network evaluation. In: Proceedings of the V Brazilian Symposium on Computing Systems Engineering
2. Brooks C, Lee EA, Liu X, Zhao Y, Zheng H, Bhattacharyya SS, Brooks C, Cheong E, Goel M, Kienhuis B, Lee EA, Liu J, Liu X, Muliadi L, Neuendorffer S, Reekie J, Smyth N, Tsay J, Vogel B, Williams W, Xiong Y, Zhao Y, Zheng H (2005) Ptolemy ii: heterogeneous concurrent modeling and design in java. Technical report
3. Caliskanelli I, Harbin J, Indrusiak LS, Mitchell P, Polack F, Chesmore D (2013) Bioinspired load balancing in large-scale WSNs using pheromone signalling. Int J Distrib Sens Netw 9(7):172012. doi:10.1155/2013/172012
4. Delicato FC, Pires PF, Pirmez L, da Costa Carmo LFR (2003) A flexible middleware system for wireless sensor networks. In: Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware, Middleware '03. Springer, New York, pp 474–492
5. Doddapaneni K, Ever E, Gemikonakli O, Malavolta I, Mostarda L, Muccini H (2012) A model-driven engineering framework for architecting and analysing wireless sensor networks. In: Proceedings of the Third International Workshop on Software Engineering for Sensor Network Applications, SESENA '12. IEEE Press, Piscataway, pp 1–7
6. Dohare U, Lobiyal DK, Kumar S (2014) Energy balanced model for lifetime maximization in randomly distributed wireless sensor networks. Wirel Pers Commun 78(1):407–428
7. Ferreira P, Brisolara L, Soares Indrusiak L (2016) Eboracum project. http://sourceforge.net/projects/eboracum/
8. Ferreira P, Brisolara L, Soares Indrusiak L (2015) Decentralised load balancing in event-triggered wsns based on ant colony work division. In: Proceedings of the 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp 69–75
9. Grassi PR, Beretta I, Rana V, Atienza D, Sciuto D (2012) Knowledge-based design space exploration of wireless sensor networks. In: Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS '12. ACM, New York, pp 225–234
10. Haghighi M (2013) Dynamic data storage estimation for multiple concurrent applications using probability distribution modeling in wsns. J Adv Comput Netw 1(3):254–259
11. Imran M, Said A, Hasbullah H (2010) A survey of simulators, emulators and testbeds for wireless sensor networks. In: Proceedings of the 2010 International Symposium in Information Technology (ITSim), vol 2, pp 897–902
12. nc M. Iris: Wireless measurement system. http://www.memsic.com/userfiles/files/Datasheets/WSN/IRIS_Datasheet.pdf
13. Iova O, Theoleyre F, Noel T (2015) Using multiparent routing in rpl to increase the stability and the lifetime of the network. Ad Hoc Netw 29(C):45–62
14. Isik S, Donmez MY, Tunca C, Ersoy C (2013) Performance evaluation of wireless sensor networks in realistic wildfire simulation scenarios. In: Proceedings of the 16th ACM International Conference on Modeling, Analysis 'I&' Simulation of Wireless and Mobile Systems, MSWiM '13. ACM, New York, pp 109–118
15. Jin Y, Wei D, Gluhak A, Moessner K (2010) Latency and energy-consumption optimized task allocation in wireless sensor networks. In: Proceedings of the 2010 IEEE Conference on Wireless Communications and Networking Conference (WCNC), pp 1–6
16. Juang P, Oki H, Wang Y, Martonosi M, Peh LS, Rubenstein D (2002) Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In: Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS X. ACM, New York, pp 96–107

17. Kacimi R, Dhaou R, Beylot AL (2013) Load balancing techniques for lifetime maximizing in wireless sensor networks. Ad Hoc Netw 11(8):2172–2186
18. Lee E (2008) Cyber physical systems: design challenges. In: Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), pp 363–369
19. Levis P, Lee N, Welsh M, Culler D (2003) Tossim: Accurate and scalable simulation of entire tinyos applications. Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys '03. ACM, New York, pp 126–137
20. Menichelli F, Olivieri M (2010) Tiktak: a scalable simulator of wireless sensor networks including hardware/software interaction. Wirel Sens Netw 2(11):815–822
21. Mozumdar M, Song ZY, Lavagno L, Sangiovanni-Vincentelli AL (2014) A model-based approach for bridging virtual and physical sensor nodes in a hybrid simulation framework. Sensors 14(6):11,070
22. Navarro M, Liang Y (2016) Efficient and balanced routing in energy-constrained wireless sensor networks for data collection. In: Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks, EWSN '16. Junction Publishing, pp 101–113
23. Ochirsuren E, Indrusiak L, Glesner M (2008) An actor-oriented group mobility model for wireless ad hoc sensor networks. In: Proceedings of the 28th International Conference on Distributed Computing Systems Workshops, pp 174–179
24. Park S, Savvides A, Srivastava MB (2000) Sensorsim: a simulation framework for sensor networks. In: Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWIM '00. ACM, New York, pp 104–111
25. Pathak A, Prasanna VK (2010) Energy-efficient task mapping for data-driven sensor network macroprogramming. IEEE Trans Comput 59(7):955–968
26. Potdar V, Sharif A, Chang E (2009) Wireless sensor networks: a survey. In: Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops, pp 636–641
27. Rodrigues T, Dantas P, Delicato F, Pires P, Pirmez L, Batista T, Miceli C, Zomaya A (2011) Model-driven development of wireless sensor network applications. In: Proceedings of the 2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing (EUC), pp 11–18
28. Rosello V, Portilla J, Krasteva Y, Riesgo T (2009) Wireless sensor network modular node modeling and simulation with visualsense. In: Proceedings of the 35th Annual Conference of IEEE Industrial Electronics, pp 2685–2689
29. Ruiz LB, Siqueira IG, Oliveira LBe, Wong HC, Nogueira JMS, Loureiro AAF, (2004) Fault management in event-driven wireless sensor networks. In: Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM '04. ACM, New York, pp 149–156
30. Salhi I, Ghamri-Doudane Y, Lohier S, Roussel G (2010) Network coding for event-centric wireless sensor networks. In: Proceedings of the 2010 IEEE International Conference on Communications (ICC), pp 1–6
31. Santi P (2012) Mobility models for next generation wireless networks: ad-hoc, vehicular and mesh networks. Wiley, Hoboken
32. Senthilkumar J, Chandrasekaran M, Suresh Y, Arumugam S, Mohanraj V (2012) Advertisement timeout driven bee's mating approach to maintain fair energy level in sensor networks. Appl Soft Comput 12(7):1884–1890
33. Sobeih A, Chen WP, Hou J, Kung LC, Li N, Lim H, ying Tyan H, Zhang H (2005) J-sim: a simulation environment for wireless sensor networks. In: Proceedings of the 38th Annual Simulation Symposium, pp 175–187
34. Soong TT (2004) Fundamentals of probability and statistics for engineers. Wiley-Backwall, New York
35. Sungur C, Spiess P, Oertel N, Kopp O (2013) Extending bpmn for wireless sensor networks. In: Proceedings of the 2013 IEEE 15th Conference on Business Informatics (CBI), pp 109–116
36. Zeng Z, Liu A, Li D, Long J (2008) A highly efficient dag task scheduling algorithm for wireless sensor networks. In: The 9th International Conference for Young Computer Scientists, pp 570–575