Just-in-Time Adaptive Algorithm for Optimal Parameter Setting in 802.15.4 WSNs

SIMONE BRIENZA, University of Pisa
MANUEL ROVERI, Politecnico di Milano
DOMENICO DE GUGLIELMO, University of Pisa
GIUSEPPE ANASTASI, University of Pisa

Recent studies have shown that the IEEE 802.15.4 MAC protocol suffers from severe limitations, in terms of reliability and energy efficiency, when the CSMA/CA parameter setting is not appropriate. However, selecting the optimal setting that guarantees the application reliability requirements, with minimum energy consumption, is not a trivial task in wireless sensor networks, especially when the operating conditions change over time. In this paper we propose a *Just-in-Time LEarning-based Adaptive Parameter tuning* (JIT-LEAP) algorithm that adapts the CSMA/CA parameter setting to the time-varying operating conditions by also exploiting the past history to find the most appropriate setting for the current conditions. Following the approach of *active* adaptive algorithms, the adaptation mechanism of JIT-LEAP is triggered by a change detection test only when needed (i.e., in response to a change in the operating conditions). Simulation results show that the proposed algorithm outperforms other similar algorithms, both in stationary and dynamic scenarios.

• **Networks→Network Protocols.**

General Terms: Design, Algorithms, Performance, Reliability

Additional Key Words and Phrases: Wireless sensor networks, IEEE 802.15.4, CSMA/CA, active adaptive algorithms, change detection tests.

1. INTRODUCTION

Wireless sensor networks (WSNs) are composed of a (large) number of tiny sensor nodes deployed over a certain geographical area and interconnected through wireless links. Each node is a low-power device that senses physical information from the surrounding environment, performs a local processing of acquired data and transmits them to a coordinator node referred to as *sink*. Given the relatively low cost, simple installation and ease of deployment, WSNs are increasingly perceived as an effective technology for developing distributed sensing systems in a large number of application domains, ranging from environmental monitoring to logistics, from healthcare to industrial applications, from building automation to smart cities. This positive trend is also pushed by a number of available communication standards for WSNs [IEEE 802.15.4 2006; ZigBee 2007; HART 2012; ISA-100.11a 2009]. Among them, IEEE 802.15.4 [2006] is certainly the most popular one for commercially-available off-the-shelf sensor platforms. A proper choice of the communication protocol is of fundamental importance in WSNs, since sensor nodes are energy constrained and communication is typically the most energy-consuming activity [Anastasi et al. 2009].

Authors' addresses: S. Brienza, D. De Guglielmo, and G. Anastasi, Department of Information Engineering, University of Pisa, Italy, e-mails: simone.brienza@for.unipi.it, d.deguglielmo@iet.unipi.it, giuseppe.anastasi@unipi.it; M. Roveri, Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy, e-mail: manuel.roveri@ polimi.it.

In this perspective, IEEE 802.15.4 is a standard specifically designed for low-power, low-rate, low-cost Personal Area Networks (PANs) that defines the *physical* (PHY) and *Medium Access Control* (MAC) layers of the protocol stack. It supports both *star* and *peer-to-peer* topologies, and provides two different operation modes, namely *Beacon Enabled (BE)* and *Non-Beacon Enabled (NBE)* mode. In this paper, we focus on the BE mode, since it is the most popular one, and provides a power-saving mechanism, based on duty-cycling. In BE mode nodes periodically wait for the reception of a special control message, called *Beacon*, from the coordinator node. Then, they transmit their data packets using a slotted CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) algorithm to access the shared wireless medium.

As mentioned above, energy efficiency is typically the most critical aspect to consider in the design of WSN-based systems. However, in many application domains, additional requirements such as reliability and timeliness must be taken into account as well [Zurawski 2009]. In this respect, several studies [Yedavalli and Krishnamachari 2008; Singh et al. 2008; Pollin et al. 2008; Anastasi et al. 2011] have highlighted that 802.15.4 WSNs suffer from severe limitations in terms of reliability (i.e., low packet delivery probability) and timeliness (i.e., high packet latency). These limitations are mainly due to the CSMA/CA algorithm used by the 802.15.4 MAC protocol. As well known, the packet delivery probability of CSMA/CA protocols degrades sharply when the number of nodes increases. In the 802.15.4 MAC this degradation is even stronger than in other similar MAC protocols due to the default CSMA/CA parameter values suggested by the standard. Anastasi et al. in [2011] have shown that these default values are not appropriate, even when the number of sensor nodes is low. The delivery probability can be increased by using higher CSMA/CA parameter values. However, this comes at the cost of a higher latency and energy consumption. Hence, an appropriate parameter setting should be found, depending on the application requirements.

Ideally, the CSMA/CA parameter setting should be chosen in such a way to guarantee the reliability (and timeliness) requirements of the application with minimum energy consumption at sensor nodes. However, in real WSNs the identification of such optimal setting is not a trivial task, as reliability and timeliness strongly depend on a number of time-varying factors – such as number of sensor nodes, offered load and packet error rate (PER) – that can neither be controlled nor predicted. Several solutions have been proposed to identify the optimal CSMA/CA setting in 802.15.4 WSNs. They can be broadly classified as *model-based* strategies [Park et al. 2009; Park et al. 2013], and *measurement-based* strategies [DiFrancesco et al. 2011; Brienza et al. 2013a]. A detailed analysis of the related literature is presented in Section 2, where we also emphasize the limitations of the existing solutions.

To overcome these limitations, in this paper we propose a *Just-in-Time LEarning-based Adaptive Parameter tuning* (JIT-LEAP) algorithm. JIT-LEAP follows a measurement-based approach, hence it does not make any assumption on the channel conditions, and does not require any a-priori information about the WSN (e.g., number of nodes). This makes it suitable for real-life scenarios where operating conditions may change over time. JIT-LEAP allows nodes to derive the optimal setting *autonomously*, i.e., by relying only on local measurements. Furthermore, it avoids unnecessary energy wastes and, by learning from the past history, is able to speed up the selection of the optimal setting. Following an *active* approach for adaptive algorithms, JIT-LEAP relies on a *change detection test* to monitor the

network conditions and trigger the adaptation mechanism only when necessary [Boracchi and Roveri 2014]. Besides, a theoretically grounded statistical technique is considered to characterize the operating conditions of the network once a change has been detected (this is fundamental to identify previously encountered network conditions).

In summary, this paper makes the following contributions. We propose a just-in-time adaptive algorithm for CSMA/CA parameter setting in 802.15.4 WSNs that is practical and suitable for real-life scenarios. To the best of our knowledge, JIT-LEAP is the first solution combining a statistical change detection test and a learning mechanism to promptly detect changes in the operating conditions and speed up the adaptation. We show, by simulation, that JIT-LEAP outperforms all the previous solutions meant to identify the optimal CSMA/CA setting in 802.15.4 WSNs.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes the 802.15.4 standard. Section 4 formulates the problem addressed in this paper in a formal way. Section 5 presents the JIT-LEAP algorithm. Section 6 describes the simulation setup, while Section 7 presents the simulation results. Conclusions are drawn in Section 8.

2. RELATED WORK

IEEE 802.15.4 WSNs (in Beacon Enabled mode) have been extensively studied in the past and many proposals have been presented to improve their performance and/or introduce additional features not provided by the standard. A thorough review of the main proposed solutions is available in [Khanafer et al. 2014], where a taxonomy is also provided, based on eight different categories, namely *Priority-based*, *QoS-based*, *Hidden Node Resolution-based*, *IEEE 802.11-based*, *Duty Cycle-based*, *Backoff-based*, *Parameter Tuning-based* and *Cross Layer-based*.

Priority-based and *QoS-based* solutions aim at adding functionalities and flexibility to the IEEE 802.15.4 MAC protocol in order to provide a better support to time-sensitive applications. This can be achieved by allowing priorities among sensor nodes [Kim and Kang 2010] or enhancing the GTS mechanism (see Section 3) [Na et al. 2008]. On a similar basis, *Hidden Node Resolution-based* approaches [Koubaa et al. 2009, Sheu et al. 2009] enhance the 802.15.4 MAC protocol to make it more aware of the existence of hidden nodes. This reduces the number of collisions and allows a better utilization of communication and energy resources. All the solutions belonging to the previous classes typically introduce modifications in the standard MAC protocol.

IEEE 802.11-based approaches [Lee et al. 2009] apply strategies conceived for improving the performance of 802.11 Wireless LANs to 802.15.4 WSNs. The main drawback of these solutions is that they have not been designed considering energy efficiency as a primary concern. Hence, they are not suitable for most of WSNs.

Duty-cycle based approaches [Kwon and Chae 2006; Lee et al. 2007; Neugebauer et al. 2005] dynamically adapt the duty-cycle of sensor nodes to traffic conditions. Increasing the duty-cycle gives sensor nodes more chances to transmit and, hence, reduces the contention for channel access. However, such solutions are ineffective when a large number of sensor nodes is transmitting simultaneously (e.g., periodic or event-driven traffic). Furthermore, the energy consumption increases with the duty-cycle.

Backoff-based approaches [Lee et al. 2009; Khan et al. 2010; Khanafer et al. 2011] propose modifications to the backoff algorithm of 802.15.4 CSMA/CA. Basically, they adapt the backoff window size depending on network congestion and wireless

medium conditions. The proposed modifications can actually increase the throughput and reduce the delay. However, they are not standard-compliant and, hence, they cannot be adopted by sensor-platforms implementing the MAC-layer functionalities in hardware or not allowing changes to the MAC protocol.

Finally, *Parameter Tuning-based* [Zhao et al. 2010; Rao and Marandin 2006; Park et al. 2009; Park et al. 2013] and *Cross Layer-based* [DiFrancesco et al. 2011, Brienza et al. 2013a] solutions improve the performance/reliability of 802.15.4 WSNs, by appropriately tuning the CSMA/CA parameters. The difference between the two categories is that, in *Cross Layer-based* solutions, CSMA/CA parameters (i.e., MAC-layer parameters) are tuned by exploiting also information provided by other layers in the protocol stack (e.g., application, or network layer). Since both classes rely on the same basic idea (i.e., parameter tuning), for simplicity, hereafter we will refer to them, generically, as solutions based on parameter tuning. Such solutions do not modify the MAC protocol and, hence, they can be implemented on any sensor platform. JIT-LEAP belongs to this category.

The idea of tuning of CSMA/CA parameters is motivated by a number of studies [Yedavalli and Krishnamachari 2008; Singh et al. 2008; Pollin et al. 2008; Anastasi et al. 2011] showing that the 802.15.4 MAC has severe limitations in terms of reliability and timeliness, mainly due to an improper setting of its CSMA/CA algorithm. Specifically, Anastasi et al. in [2011] have shown that unreliability in 802.15.4 WSNs is exacerbated by the default CSMA/CA parameter values suggested by the standard, that are inappropriate, even when the number of sensor nodes is low. Using more appropriate (i.e., higher) parameter values improves reliability, at the cost of increased latency and energy consumption.

Solutions based on parameter tuning can be further distinguished, depending on the number of CSMA/CA parameters they consider and the methodology they use for their tuning. Some solutions [Zhao et al. 2010; Rao and Marandin 2006] focus on a single CSMA/CA parameter (e.g. *macMinBE*). However, adjusting a single parameter may not be sufficient to meet the reliability requirements of the application [Anastasi et al. 2011]. For this reason, other solutions [Park et al. 2009; Park et al. 2013; DiFrancesco et al. 2011, Brienza et al. 2013a] consider the whole set of CSMA/CA parameters. Also JIT-LEAP falls in the latter category and, hence, hereafter we will focus on such solutions. As regards the methodology used for parameter tuning, according to the taxonomy introduced in [Brienza et al. 2013b], the proposed solutions can be classified as *model-based offline computation* [Park et al. 2009], *model-based adaptation* [Park et al. 2013] and *measurements-based adaptation* [DiFrancesco et al. 2011, Brienza et al. 2013a].

Model-based strategies rely on an analytical model of the WSN to derive the optimal parameter setting, under the current operating conditions. This is done either by solving the analytical model offline [Park et al. 2009], or by using it to dynamically adapt to time-varying operating conditions [Park et al. 2013]. Model-based approaches have a number of limitations. First, the effectiveness in providing the optimal setting depends on the accuracy of the analytical model. Typically, simplifying assumptions are introduced to make the model tractable. Furthermore, the model usually requires some input parameters, which may not be available in a real environment. For instance, the model used in [Park et al. 2009] and [Park et al. 2013] assumes ideal channel conditions and requires knowing in advance the number of network nodes. Differently, JIT-LEAP considers a real communication channel where packet errors/losses can occur, and does not require any input parameter. Hence, it is suitable for real-life scenarios.

Measurement-based approaches [DiFrancesco et al. 2011, Brienza et al. 2013a] do not require any network model, instead they rely on measurements acquired by sensor nodes. For instance, ADAPT [DiFrancesco et al. 2011] is a heuristic algorithm that allows sensor nodes to autonomously adjust the CSMA/CA parameters, according to local measurements of the packet delivery probability. Specifically, parameter values are increased or decreased depending on the delivery probability experienced by the node. However, ADAPT tends to oscillate between two or more parameter sets and never stabilizes, thus consuming more energy than necessary. Furthermore, it is memory-less. This means that, if the same operating conditions repeat over time, the algorithm is not able to recall the previously calculated optimal parameter setting and re-executes the adaptation procedure.

Like ADAPT, JIT-LEAP follows a measurement-based approach, though the analyzed quantities are different. In addition, it also exploits the knowledge acquired through a learning mechanism to select the optimal parameter setting, based on the past history. JIT-LEAP belongs to the class of *active* adaptive algorithms [Alippi 2014] since it relies on a trigger mechanism to activate the adaptation phase only when needed. Differently from *passive* algorithms (e.g., [Park et al. 2013]) – where the adaptation mechanism is always on – active algorithms are generally prompter in adapting to new conditions, hence providing better performance and lower energy consumption. Formally, also ADAPT is an active algorithm as the adaptation mechanism is activated only when the locally-estimated packet delivery probability is below/over a predefined threshold. In practice, ADAPT tends to change the parameter setting (almost) at each step, thus behaving similarly to passive algorithms.

The JIT-LEAP algorithm presented in this paper extends a previous *LEarning-based Adaptive Parameter tuning* (LEAP) algorithm presented in [Brienza et al. 2013a]. While LEAP assumes ideal channel conditions (i.e., error/loss free channel), JIT-LEAP overcomes this limitation by explicitly taking into account packet losses and transmission errors. This makes it suitable for real-life scenarios. In addition, unlike LEAP, JIT-LEAP relies on a theoretically-grounded mechanism to detect changes in the operating conditions, based on a statistical Change Detection Test (CDT). This allows to significantly reduce the number of false positive detections and identifying even small variations in the operating conditions. Hence, the parameter setting provided by JIT-LEAP is more stable and accurate, allowing to strictly adhere to the application requirements.

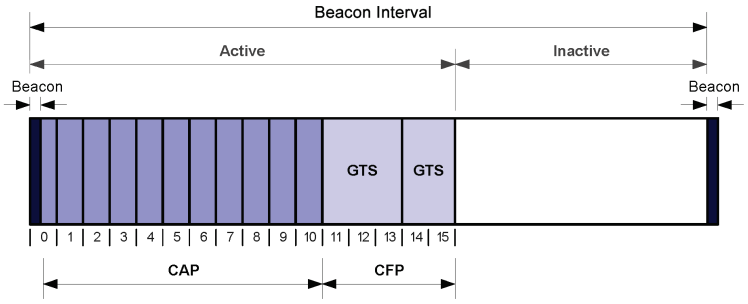

Fig. 1. Superframe structure

3. 802.15.4 MAC PROTOCOL

The 802.15.4 MAC in Beacon Enabled mode provides a power management mechanism, based on a duty-cycle, and relies on a *superframe* structure bounded by

Beacons, i.e., special messages transmitted periodically by coordinator nodes (Figure 1). The time interval between two consecutive Beacons is called *Beacon Interval (BI)* and each superframe consists of an *Active Period* and an *Inactive Period*. During the Active Period sensor nodes can communicate with their coordinator. Instead, in the Inactive Period, they enter a low-power state to save energy. The Active Period is further divided into a *Contention Access Period (CAP)* and a *Collision Free Period (CFP)*. During the CAP, nodes use a slotted CSMA/CA algorithm for channel access, while during the CFP they use *Guaranteed Time Slots (GTSs)* in a *TDMA (Time Division Multiple Access)* style.

3.1 CSMA/CA Algorithm

In the *slotted* CSMA/CA algorithm, used during CAP, time is divided into slots of equal duration (*backoff slots*) and all the operations are aligned with them. Upon receiving a data packet to transmit, each node executes a *backoff stage*, i.e., it waits for a *random* number of backoff slots (*backoff time*) and, then, performs two consecutive *Clear Channel Assessments* (CCAs) to check the channel state. If the channel is found idle in both CCAs, the node transmits its packet. Otherwise, it must perform a new backoff stage. After the transmission of a packet, the sensor node waits for the acknowledgment from the recipient. If the acknowledgement is not received within a predefined timeout, a retransmission is triggered. The transmission of a packet may result either in a success (if an acknowledgment is eventually received) or in a packet drop. A packet is dropped when either the maximum number

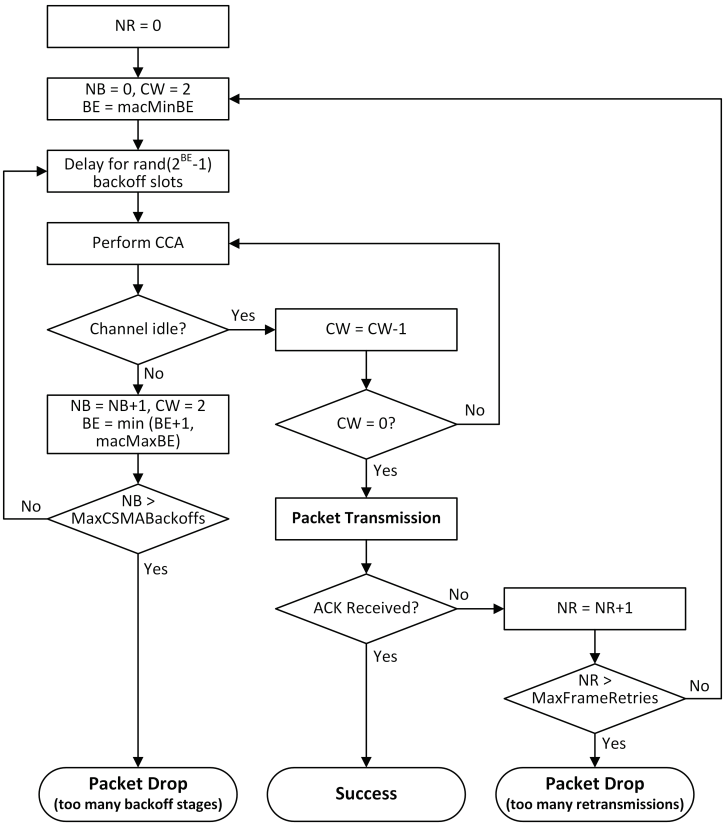

Fig. 2. 802.15.4 slotted CSMA/CA Algorithm

of consecutive backoff stages, or the maximum number of retransmissions is exceeded.

Figure 2 presents the slotted CSMA-CA algorithm by detailing the sequence of actions performed by a sensor node to transmit a packet. Each node maintains a number of state variables, namely *contention window size* (*CW*), *number of backoff stages* (*NB*), *backoff exponent* (*BE*) and *number of retransmissions* (*NR*). *CW* specifies the number of CCAs still to perform in the current backoff stage. It is initialized to 2 and, hence, the sensor node has to perform two consecutive CCAs before starting the packet transmission. *BE* defines the maximum (random) backoff delay a node will wait, at each backoff stage, before checking the channel state. It is initialized to *macMinBE* and incremented every time the channel is found busy during the CCAs, i.e., before starting a new backoff stage (however *BE* cannot exceed *macMaxBE).* Basically, the number of backoff slots to wait in a backoff stage is randomly chosen in the interval $[0;\ 2^{BE} - 1]$. *NB* indicates the number of backoff stages performed for the current transmission attempt. If *NB* exceeds the maximum allowed value *macMaxCSMABackoffs*, the packet is dropped. Finally, *NR* indicates the number of retransmissions for the current packet and is incremented every time the acknowledgement is not received. If *NR* exceeds the *macMaxFrameRetries* parameter, the packet is dropped.

From the previous description it emerges that the CSMA/CA behavior is regulated by four parameters, listed in Table I together with the range of values allowed by the 802.15.4 standard.

Table I. CSMA/CA Parameters and values [IEEE 802.15.4 2006]

PARAMETER	VALUES	DESCRIPTION
macMaxFrameRetries	Range: 0-7 Default: 3	Maximum number of retransmissions
macMaxCSMABackoffs	Range: 0-5 Default: 4	Maximum number of backoff stages-1
macMaxBE	Range: 3-8 Default: 5	Maximum backoff window exponent
macMinBE	Range: 0-7 Default: 3	Minimum backoff window exponent

In [DiFrancesco et al. 2011] it is shown that, in a star topology, the packet delivery probability provided by CSMA/CA increases monotonically with the values of *macMinBE, macMaxCSMABackoffs,* and *macMaxFrameRetries*. However, its increase becomes negligible after certain values of the aforementioned parameters. It is also shown that increasing *macMinBE* is more energy efficient than increasing *macMaxCSMABackoffs* (i.e., it improves the packet delivery probability with a lower energy consumption), whereas increasing *macMaxCSMABackoffs* is more energy efficient than increasing *macMaxFrameRetries*. These conclusions are at the basis of the design of the JIT-LEAP algorithm (as described in Section 5).

4. PROBLEM FORMULATION

In the following we refer to a WSN with a star topology, including a sink node (acting as the coordinator) and a number of sensor nodes. We consider a periodic reporting application where data gathered by a sensor node are reported to the sink at each Beacon Interval. We assume that data/acknowledgement packets transmitted by nodes may be corrupted or lost. Despite that, the application requires a certain reliability level (expressed as percentage of data packets correctly delivered to the sink), that must be guaranteed with minimum energy consumption. To formulate the

problem in a more formal way, we define the following indexes:

— *Packet delivery ratio* (D): the ratio between the number of data packets correctly delivered to the sink by a sensor node, and the total number of packets generated by that node. It measures the *long-term reliability* experienced by a sensor node, and is requested to be higher than a minimum value D^{min}.

— *Miss ratio* (M): the fraction of times the packet delivery ratio – calculated by a sensor node over the current Beacon Interval – drops below D^{min}. It measures the inability to achieve *short-term reliability* and should not exceed a pre-defined threshold M^{max}.

— *Average energy consumption per packet* (E_P): the total energy consumed by a sensor node divided by the total number of packets generated by that node. It measures the *energy efficiency*.

Let $D(\boldsymbol{par})$, $M(\boldsymbol{par})$, and $E_P(\boldsymbol{par})$ denote the delivery ratio, miss ratio, and average energy consumption, respectively, experienced by a sensor node when using a set of CSMA/CA parameter values denoted by $\boldsymbol{par}$. Hence, the problem of optimal parameter setting can be formulated as

$$\begin{cases} minimize\ E_P(\boldsymbol{par}) \\ D(\boldsymbol{par}) \geq D^{min} \\ M(\boldsymbol{par}) \leq M^{max} \end{cases} \quad (1)$$

A possible approach for solving this problem is through the derivation of an analytical model of the WSN and the computation of the optimal CSMA/CA parameter setting that satisfy (1). This is very close to the approach used in [Park et al. 2009], where the authors consider delivery ratio and latency (instead of miss ratio). As emphasized in Section 2, the use of a model-based approach has some limitations that make it unsuitable for real-life scenarios. In this paper we use a heuristic solution, following a measurement-based approach that leverages a change detection test and a learning algorithm to identify the optimal setting adaptively.

5. JIT-LEAP ALGORITHM DESCRIPTION

In this section we describe the proposed JIT-LEAP algorithm. We start with a high-level presentation (Section 5.1). Then, we detail the different phases of the algorithm (Sections 5.2-5.5). Finally, we describe some optimization mechanisms for improving its energy efficiency (Section 5.6).

5.1 Basic Ideas

The goal of JIT-LEAP is to select the set of CSMA/CA parameter values (depending on the current operating conditions) that satisfy the reliability requirements of the application with minimum energy consumption at sensor nodes. To this end, it also exploits the knowledge learned in the past history (if any). Figure 3 details the actions performed by JIT-LEAP during each Beacon Interval. First of all, each node characterizes the current network conditions by measuring some quantities related to network congestion and channel unreliability (see Section 5.2). In addition, each node derives the estimates of delivery ratio and miss ratio experienced in the current Beacon Interval and inserts them into a specific data structure called *Experienced-Performance Table* (see Sections 5.2 and 5.3). The latter table is used to store the performance experienced, with each set of parameters, since the last change in the

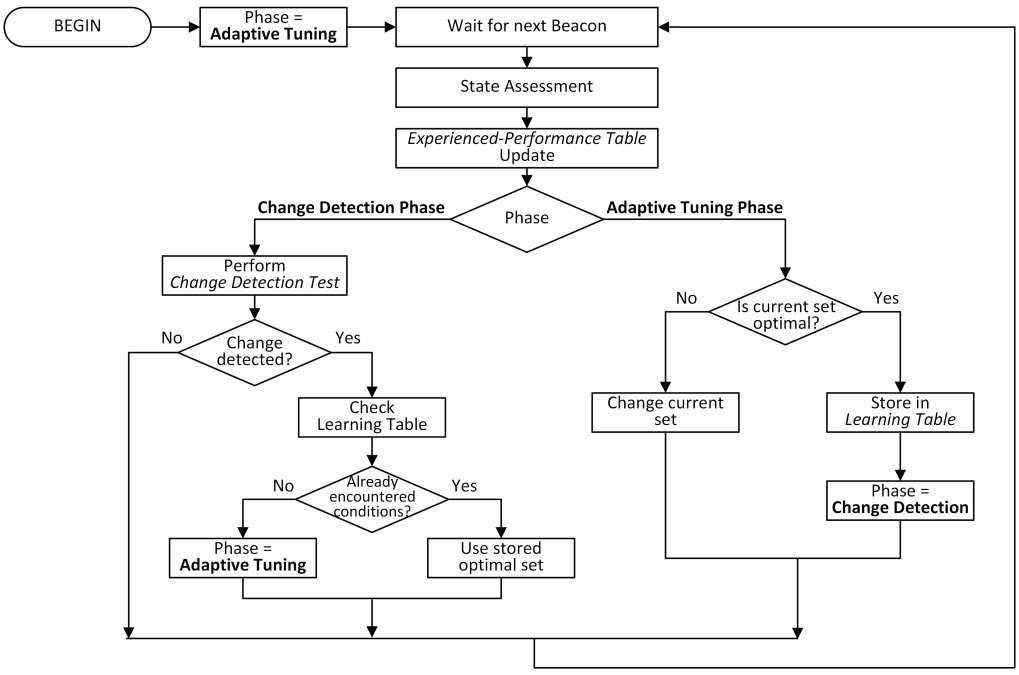

Fig.3. Actions performed by JIT-LEAP during each Beacon Interval

network conditions. Then, the algorithm behaves in different ways depending on its current operating phase, namely Adaptive Tuning or Change Detection Phase.

1) *Adaptive Tuning Phase* (right side of Figure 3). When no information about the current operating conditions is available (e.g., at startup), the sensor node executes an Adaptive Tuning algorithm similar to ADAPT [DiFrancesco et al. 2011]. CSMA/CA parameter values are increased when the reliability experienced by the sensor node (in terms of D and M) does not satisfy the application requirements, and decreased otherwise. After a number of steps, the Adaptive Tuning algorithm starts oscillating between two parameter sets. This means that the most approriate setting, for the current conditions, has been reached. Hence, this information is inserted into an appropriate data structure called *Learning Table* (see below for details) which indicates, for each parameter set, the optimal set to be used if a certain network condition is encountered. Then, the algorithm moves to the *Change Detection Phase*.
2) *Change Detection Phase* (left side of Figure 3). This phase aims at detecting possible changes in the operating conditions, so as to trigger the adaptation to the new conditions. To this end, a *Change Detection Test (CDT)*, i.e., an on-line statistical test, is performed to detect possible variations in the operating conditions measuring network congestion and channel unreliability. If no change is detected, the current CSMA/CA parameter values used by the sensor node are not updated and no further actions are performed. Conversely, if a change is detected, JIT-LEAP first identifies the new operating conditions and, then, determines the new optimal setting. If similar conditions have been already experienced in the past, the learning mechanism allows to immediately re-activate the optimal setting previously used. Specifically, upon detecting a change, the algorithm checks if there is an entry in the Learning Table corresponding to

the current set and the new network conditions. The following two outcomes can occur.

— The Learning Table contains an entry matching the new operating conditions with the corresponding optimal setting. Therefore, the node sets up the optimal parameter values suggested by the table (just in one step, or *leap*). Afterwards, the Change Detection phase restarts.

— There is no entry in the Learning Table for the new operating conditions. Therefore, a new Adaptive Tuning phase starts.

In the next subsections we will detail the data structures used by JIT-LEAP and the actions carried out during the Adaptive Tuning and Change Detection phases.

5.2 State Assessment and Data Structures

We use $\boldsymbol{s}(t)=[p_b(t), p_f(t), \boldsymbol{par}(t)]$ to store the state of the sensor network, as perceived by a generic sensor node, at a given Beacon Interval t. Specifically, $\boldsymbol{par}(t)$ is the used set of CSMA/CA parameter values, $p_b(t)$ denotes the probability to find the channel busy during a channel access, and $p_f(t)$ gives the probability that a packet transmission fails (i.e., the sensor node does not receive the related acknowledgment). In particular, p_b is a measure of the congestion experienced by the sensor node and depends on some factors such as number of sensor nodes and offered load. It also depends on the CSMA/CA parameter setting used by the sensor node, i.e., $\boldsymbol{par}$. On the contrary, p_f measures the communication unreliability and mainly depends on the wireless medium unreliability (i.e., the Packet Error Rate, PER). However, it also depends on the network congestion since a transmission can fail also due to a collision. Despite p_b and p_f depend on many different factors, for simplicity, we will use the terms $p_b(t)$ and $p_f(t)$ to indicate their values at Beacon Interval t.

The state vector $\boldsymbol{s}(t)$ is derived by the sensor node as follows. The CSMA/CA parameter values (i.e., $\boldsymbol{par}(t)$) are known. $p_b(t)$ can be measured locally as $p_b(t) = p_b^1 + (1 - p_b^1) \times p_b^2$, where p_b^1 (p_b^2) is the probability to find the channel busy during the first (second) CCA operation. In practice, p_b^1 and p_b^2 are estimated by calculating the fraction of CCA operations resulted in a busy channel in the current Beacon Interval. Similarly, $p_f(t)$ is computed as the fraction of transmissions for which the acknowledgment was missed during the current Beacon Interval.

At each sensor node JIT-LEAP uses the following data structures to store information.

— *Experienced-Performance Table*. A table with an entry for each CSMA/CA parameter setting used since the last change in the operating conditions (or network startup). The table is cleared whenever a new change is detected. Each entry has the following format: $\langle \boldsymbol{par}, D, M, F, count \rangle$, where D (M) represents the delivery ratio (miss ratio) experienced by the sensor node with the $\boldsymbol{par}$ parameter set. F denotes the *Transmission Failure Ratio*, defined as the ratio between the number of transmissions for which the acknowledgment was missed, and the total number of transmissions performed by the sensor node using the $\boldsymbol{par}$ parameter set. Finally, *count* indicates the number of times the corresponding parameter set has been used so far.

— *Training Buffer*. A data structure containing the last W experienced states. It is needed for training the CDT, both at network startup and after a change detection.

— *State Sample.* Whenever the CDT detects a change in the operating conditions, it estimates the Beacon Interval τ when the change more likely occurred. Then, it calculates the mean value and standard deviation of p_b and p_f over the Beacon Intervals between τ and the instant when the change has been detected. These values characterize the new operating conditions. Thus, they are inserted into a proper data structure, called *State Sample*, that will be used to build the *Learning Table* at the end of the next Adaptive Tuning phase.

— *Learning Table.* This data structure is created at the end of the first Adaptive Tuning phase, and updated after each Adaptive Tuning phase, on the basis of the State Sample. The Learning Table contains information about each operating condition experienced during the past history, and the corresponding optimal setting, according to the previously acquired knowledge. Each entry in the table has the following format: $\langle \boldsymbol{par}, elem_1, elem_2, \ldots, elem_i, \ldots \rangle$, where $elem_i = \langle [p^i_{b\_min}, p^i_{b\_max}], [p^i_{f\_min}, p^i_{f\_max}], \boldsymbol{new\_set} \rangle$ for any i. Basically, each operating condition is represented by an element *elem*, where the two intervals $[p_{b\_min}, p_{b\_max}]$ and $[p_{f\_min}, p_{f\_max}]$ indicate the range of p_b and p_f characterizing that specific operating condition. Therefore, the table tells that, whenever the estimated values of p_b and p_f fall within the above-mentioned intervals, and the parameter set $\boldsymbol{par}$ is used, the new optimal setting must be $\boldsymbol{new\_set}$. Examples on how to access and use the Learning Table will be given below.

5.3 Experienced-Performance Table Update

As anticipated in Section 5.1, CSMA/CA parameter values are increased or decreased, depending on the current estimates of delivery ratio and miss ratio (i.e, D and M). For this reason, for each used set, the algorithm stores, inside the Experienced-Performance Table, the value of D and M experienced with that set. At each Beacon Interval, the entry corresponding to the current set is updated, as follows:

$$D = \frac{\overline{D} + D \cdot count}{count + 1} \quad M = \frac{\overline{M} + M \cdot count}{count + 1}$$

$\overline{D}$ and $\overline{M}$ represent the delivery ratio and miss ratio measured in the current Beacon Interval, while *count* tracks the number of times the corresponding set has been used so far (it is increased after each update). This way, the estimates of D and M get more and more accurate over time.

If the channel is ideal (i.e., PER = 0), $\overline{D}$ can be obtained as $\overline{D} = P_{ACK}/P_{gen}$, where P_{gen} is the number of packets generated by the sensor node and P_{ACK} is the number of received acknowledgements. Furthermore, if $\overline{D} > D^{min}$, then $\overline{M} = 0$; otherwise $\overline{M} = 1$. If the channel is not ideal (i.e., PER > 0), the previous formula for $\overline{D}$ underestimates the delivery ratio since a packet may have been delivered correctly to the sink even if the corresponding acknowledgment was missed by the sensor node. To correctly estimate the delivery ratio, when PER > 0, we need to take into account also the packets dropped due to exceeded number of retransmissions, but correctly received by the sink. Let P_{MFR} denote the total number of packets dropped due to exceeded number of retransmissions and α be the probability that a dropped packet is received correctly by the sink. The delivery ratio can be estimated as $\overline{D} = \frac{P_{ACK} + P_{MFR} \cdot \alpha}{P_{gen}}$. The value of P_{MFR} is provided by the MAC layer, while α can be derived using the following claim.

CLAIM 1. Assuming that *(i)* packet transmission errors are independent from each other, and *(ii)* the PER is the same for both data packets and acknowledgements, then

$$\alpha = 1 - \left(\frac{F - PER}{(1 - PER)F}\right)^{macMAXFrameRetries+1}$$

where F denotes the transmission failure ratio, i.e., the probability that a packet transmission fails for any reason.

PROOF. See Appendix A.

The previous claim allows to derive α, once PER and F are known. PER is estimated by the sensor node by computing the ratio between the number of missed Beacons and the number of expected Beacons. Instead, the transmission failure ratio F is estimated by taking an approach similar to that used for estimating D. As mentioned above, for each used parameter set, the corresponding entry in the Experienced-Performance Table also includes a field F. The latter is updated, at each Beacon Interval, as $F = \frac{\bar{F}+F \cdot count}{count+1}$, where $\bar{F}$ is the ratio between the number of missed acknowledgements and the total number of transmissions (including retransmissions) performed by the sensor node in the current Beacon Interval.

5.4 Adaptive Tuning Phase

5.4.1 CSMA/CA parameters change

As shown in Figure 3, initially JIT-LEAP starts with a simple Adaptive Tuning algorithm to dynamically adjust CSMA/CA parameters, as it has no information about the past history. This algorithm increases or decreases one parameter at a time, depending on the experienced reliability, as follows. At each Beacon Interval, the sensor node updates the estimates of delivery ratio D and miss ratio M with measurements taken in the current Beacon Interval. If at least one of these estimates does not satisfy the application requirements, the value of a CSMA/CA parameter is increased, by considering first *macMinBE* and, then, *macMaxCSMABackoffs* (*macMaxBE* is kept to a fixed value, i.e., $MaxBE^{max}$). The retransmission mechanism is initially disabled. Only when both *macMinBE* and *macMaxCSMABackoffs* have

Table II. Ordered CSMA Parameter Sets

INDEX	MAXBE	MINBE	MAXCSMABACKOFFS	MAXFRAMERETRIES
1	$MaxBE_{MAX}$	$MinBE_{MIN}$	$MaxCSMABackoffs_{MIN}$	$MaxFrameRetries_{MIN}$
2	$MaxBE_{MAX}$	$MinBE_{MIN}+1$	$MaxCSMABackoffs_{MIN}$	$MaxFrameRetries_{MIN}$
3	$MaxBE_{MAX}$	$MinBE_{MIN}+2$	$MaxCSMABackoffs_{MIN}$	$MaxFrameRetries_{MIN}$
…	…	…	…	…
…	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MIN}$	$MaxFrameRetries_{MIN}$
…	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MIN}+1$	$MaxFrameRetries_{MIN}$
…	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MIN}+2$	$MaxFrameRetries_{MIN}$
…	…	…	…	…
…	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MAX}$	$MaxFrameRetries_{MIN}$
…	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MAX}$	$MaxFrameRetries_{MIN}+1$
…	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MAX}$	$MaxFrameRetries_{MIN}+2$
…	…	…	…	…
I_{MAX}	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MAX}$	$MaxFrameRetries_{MAX}$

reached their maximum value, *macMaxFrameRetries* is also progressively increased. Conversely, if both D and M satisfy the application requirements, the set of parameter values is tentatively reduced. The strategy for decreasing parameter is just the opposite. First, *macMaxFrameRetries* is progressively reduced until it reaches its minimum value. Then, the same procedure is applied to *macMaxCSMABackoffs* and, afterwards, to *macMinBE*. Without loss in generality, we can assume that CSMA/CA parameter sets are ordered as shown in Table II. Hence, each parameter setting can be identified by the corresponding index in the table and the Adaptive Tuning algorithm always moves from a set to an adjacent one.

5.4.2 Training Buffer and Learning Table update

The Training Buffer and the Learning Table are also updated during the Adaptive Tuning phase. At each Beacon Interval t, after estimating $p_b(t)$ and $p_f(t)$, the new state $\boldsymbol{s}(t) = [p_b(t),\ p_f(t),\ \boldsymbol{par}(t)]$ is added to the Training Buffer. Since the Training Buffer has a limited size, when it is full, the new state overwrites the oldest one, following a FIFO approach. We emphasize that, due to its behavior, after a (short) transient time the Adaptive Tuning algorithm tends to oscillate between two adjacent parameter sets. We assume that the Adaptive Tuning phase ends when all the states stored inside the Training Buffer refer to only two parameter sets. Then, the Training Buffer is ready to be used for training the CDT as described below. The most frequent setting within the Training Buffer is assumed to be the most appropriate set for the current operating conditions, i.e., the "optimal" set according to the Adaptive Tuning algorithm. Throughout, we will refer to this set as $\boldsymbol{par}_{opt}$.

Now, a new element can be added in the Learning Table, pointing to the optimal set $\boldsymbol{par}_{opt}$. Let us denote by $\boldsymbol{par}_{prev}$ the parameter set used before the current Adaptive Tuning phase started, i.e., before the operating conditions changed. Also, let us indicate as μ_b (μ_f) and $\sigma_b(\sigma_f)$ the mean value and standard deviation of p_b (p_f) inserted by the CDT into the State Sample. As explained in Section 5.2, these values have been calculated after the change occurred, but before the Adaptive Tuning phase started. Hence, they characterize the new operating conditions. Therefore, a new entry corresponding to set-index $\boldsymbol{par}_{prev}$ can be inserted into the Learning Table (if there is no entry for this set, a new entry is created). The added element is

$$\langle [\mu_b - \sigma_b,\ \mu_b + \sigma_b], [\mu_f - \sigma_f,\ \mu_f + \sigma_f], \boldsymbol{par}_{opt} \rangle$$

In the future, if these operating conditions are encountered again, while the set $\boldsymbol{par}_{prev}$ is used, the algorithm immediately knows that the set $\boldsymbol{par}_{opt}$ has to be used. The update of the Learning Table concludes the Adaptive Tuning phase. Then, the sensor node enters the Change Detection phase.

5.5 Change Detection Phase

5.5.1 Change Detection Test

JIT-LEAP detects possible changes in the operating conditions by inspecting variations in p_b and p_f. To achieve this goal, among the wide range of CDTs available in the literature [Basseville et al. 1993; Tartakovsky et al. 2006; Ross et al. 2011; Alippi and Roveri 2008; Kawahara and Sugiyama 2012], we focus on the family of CDTs based on the *Intersection-of-Confidence-Interval (ICI)* rule [Alippi et al. 2011], which revealed to be particularly effective in several application scenarios [Alippi et

al. 2013; Boracchi et al. 2014]. In addition, ICI-based CDTs are theoretically grounded and exhibit a reduced computational complexity, which makes them particularly suitable for WSNs. Finally, this family of CDTs follows the "non-parametric" approach, i.e., they do not require any a-priori information about the measured state variables or changes that might affect the network. This makes ICI-based CDTs particularly suitable for time-varying and a-priori unknown environments (such as WSNs). Among the ICI-based CDTs, we focus on the element-wise CDT [Boracchi and Roveri 2014]. This CDT is able to operate in an element-wise manner, thanks to a Gaussian transform of measured variables, providing very prompt detections to changes. The considered Gaussian transform is the Manly transform, i.e.,

$$\bar{p}_i(t) = \begin{cases} \dfrac{e^{\lambda p_i(t)} - 1}{\lambda}; & \lambda \neq 0 \\ p_i(t); & \lambda = 0 \end{cases}$$

where $p_i(t)$ can be either $p_b(t)$ or $p_f(t)$and $\lambda \in \mathbb{R}$ is the transform parameter. The Manly transform is applied both to $p_b(t)$ and $p_f(t)$ to generate the approximately Gaussian variables $\bar{p}_b(t)$ and $\bar{p}_f(t)$. As mentioned above, this CDT requires an initial training sequence to configure the test parameters and the parameter λ of the Manly transform. Specifically, in our scenario the CDT is configured on the Training Buffer. Details about the configuration phase of the CDT can be found in [Boracchi and Roveri 2014]. During the operational life, the CDT is then applied to $\bar{p}_b(t)$ and $\bar{p}_f(t)$ to detect possible variations in their expected values, basing on what has been learned during the training phase.

When a change is detected, the Learning Table is looked up to determine the optimal set for the new operating conditions. A key requirement for obtaining correct values from the Learning Table is the ability to correctly characterize the operating conditions after the change, in terms of the new values of p_b and p_f. To achieve this goal, once a change is detected, it is necessary to determine when it occurred. Therefore, a Change-Point Method (CPM) is applied to the Training Buffer containing the last W acquired data to identify the time instant at which the operating conditions changed. CPMs are statistical hypothesis tests [Hawkins et al. 2003] that are able to assess whether a change point exists in a given sequence of data and to locate it within the sequence. Specifically, let $\hat{T}$ be the Beacon Interval when the change was detected (either in p_b or p_f), and let $\mathcal{X}$ be the sequence of the corresponding variable up to $\hat{T}$, i.e.,

$$\mathcal{X} = \left\{\bar{p}_D\left(\left(\hat{T} - W + 1\right)\right), \dots, \bar{p}_D\left(\hat{T}\right)\right\}$$

where $\bar{p}_D(t)$ is either p_b or p_f. The CPM acts as follows. For each Beacon Interval t, such that $\hat{T} - W + 1 \leq t \leq \hat{T}$, the sequence $\mathcal{X}$ is split into two parts

$$A_t = \{\bar{p}_D\left(\hat{T} - W + 1\right), \dots, \bar{p}_D(t)\}$$
$$B_t = \{\bar{p}_D(t+1), \dots, \bar{p}_D\left(\hat{T}\right)\}$$

and a test statistics $\mathcal{T}_t$=(A_t, B_t) is computed for all the Beacon Intervals t with $\hat{T} - W + 1 \leq t \leq \hat{T}$. Let $\mathcal{T}_M$ be its maximum value, i.e.,

$$\mathcal{T}_M = \max_{t=\hat{T}-W+1,\dots,\hat{T}} \mathcal{T}_t.$$

When $\mathcal{T}_M$ is larger than a predefined threshold $H_{\varepsilon,\hat{T}}$ (that depends on the test statistic, $\hat{T}$ and a given confidence level ε) there is enough statistical confidence that a change point exists in $\mathcal{X}$. Let τ be the time instant of this change point, i.e.,

$$\tau = \underset{t=\hat{T}-W+1,\ldots,\hat{T}}{\operatorname{argmax}} \mathcal{T}_t.$$

Among the test statistics present in the literature (e.g., [Mann and Whitney 1947], [Bartlett and Kendall 1946], [Mood 1954], [Lepage 1974]), we focused on the Mann-Whitney [Mann and Whitney 1947] since we are interested in detecting change-points affecting the expected value of $\mathcal{X}$. We emphasize that τ represents an estimate of the Beacon Interval at which a change affected the operating conditions. Hence, the new operating conditions can be computed as follows:

$$\mu = \frac{1}{\hat{T}-\tau}\sum_{i=\tau}^{\hat{T}} \bar{p}_D(i)$$

$$\sigma = \frac{1}{\hat{T}-\tau-1}\sum_{i=\tau}^{\hat{T}} (\bar{p}_D(i) - P_D)^2$$

where μ and σ are respectively the sample mean and sample variance of the state variable $\bar{p}_D(t)$ after the change occurred. The values of μ and σ, for both p_b and p_f (μ_b, σ_b, μ_f and σ_f), represent the new operating conditions and are stored inside the State Sample.

5.5.2 Learning Table access

Once a change has been detected, the Learning Table is checked to verify whether the new conditions have been already experienced in the past. To this aim, the current set index, along with the values of μ_b and μ_f contained in the State Sample, are used.

Table III. Example of Learning Table

CURRENT SET	ELEM1			ELEM2		
	p_b	p_f	new set	p_b	p_f	new set
par1	[0.30, 0.33]	[0.33, 0.52]	par2	[0.64, 0.70]	[0.29, 0.51]	par3
par2	[0.43, 0.51]	[0, 0.24]	par3			
par3	[0.70, 0.75]	[0.16, 0.22]	par5			
par4	[0.64, 0.76]	[0.20, 0.43]	par8	[0.25, 0.35]	[0, 0.15]	par1

Table III shows an example of Learning Table. Let us assume that ***par4*** is the current parameter set and that $\hat{\mu}_b$ and $\hat{\mu}_f$ are the values of μ_b and μ_f stored in the State Sample. Based on the past history, the Learning Table suggests to use ***par8*** as the new set, if $\hat{\mu}_b$ is in the range [0.64, 0.76] and $\hat{\mu}_f$ is in the range [0.20, 0.43], or ***par1,*** if $\hat{\mu}_b$ is in the range [0.25, 0.35] and $\hat{\mu}_f$ in the range [0, 0.15].

When the Learning Table does not contain any entry for the values in the State Sample, JIT-LEAP infers that the current operating conditions have never been experienced in the past, and a new Adaptive Tuning phase is started to identify the optimal MAC parameter set. When the optimal set is determined, at the end of the Adaptive Tuning phase, a new entry is added to the Learning Table by using the data contained in the State Sample, as previously explained.

5.6 Controlled Tuning Algorithm

The Adaptive Tuning phase aims at identifying the parameter setting that satisfies the reliability requirements of the application with minimum energy consumption. However, since CSMA/CA parameters assume discrete values, typically the delivery ratio (miss ratio) experienced with the obtained parameter set is significantly above (below) D^{min} (M^{max}), thus consuming more energy than necessary. On the other hand, using a lower set might not satisfy the application requirements.

To reduce the energy consumption as much as possible, while still satisfying the reliability constraints, JIT-LEAP uses a *Controlled Tuning* algorithm (during both the Adaptive Tuning phase and the Change Detection phase) that finely adjusts the parameter setting on the sensor node, by switching between two adjacent sets in a controlled way. The idea is to have a reliability level just above the required value, so as to minimize the energy consumption. The Controlled Tuning algorithm is detailed in Appendix B.

6. SIMULATION SETUP

To evaluate the performance of JIT-LEAP we relied on the *ns2* simulation tool [Ns-2]. We used simulation in order to make the analysis as general as possible. However, to validate our simulation results we also implemented JIT-LEAP in a real sensor platform and carried out some experiments in a real testbed. The comparison between simulation and experimental results is shown in Appendix C (due to space limitations).

We considered a star network topology, where sensor nodes are placed in a circle centered at the sink node (PAN coordinator), 10m far from it. The transmission range was set to 15m, while the carrier sensing range was set to 30m (according to [Anastasi et al. 2005]). In our analysis, in addition to the reliability and energy efficiency indexes already introduced in Section 4 (i.e., *packet delivery ratio*, *miss ratio* and *average energy consumption per packet*), we also considered the following two performance indexes.

— *Average latency*, defined as the average time from the beginning of the packet transmission at the source node to when the packet is correctly received by the sink. This index measures the *timeliness* in delivering packets.

— *Transient time*, defined as the time instant, after a change in the operating conditions, when the packet delivery probability – calculated over the current Beacon Interval – reaches the steady-state value for the new operating conditions (with a tolerance of ±3%). This index measures the *ability to adapt* to changing conditions.

The energy consumed by a sensor node is calculated according to the model in [Bougard et al. 2008], which is based on the Chipcon CC2420 radio transceiver [Texas Instruments 2012], commonly used in sensor nodes.

6.1 Algorithms for Comparison

We compared the performance of JIT-LEAP with that of the following algorithms.

— *Model-based offline computation* [Park et al. 2009]. This algorithm derives the optimal setting offline, by solving an analytical model of the WSN. The algorithm is executed on the sink node and parameter values are then communicated to sensor nodes.

— *Model-based online adaptation* [Park et al. 2013]. This is an adaptive algorithm based on an analytical model of the WSN, which is a simplified version of that

used in the previous algorithm and, hence, can be executed at sensor nodes. Sensor nodes measures some congestion indexes locally and use them as input values for the model, so as to adapt the CSMA/CA parameter values to time-varying operating conditions.

— *ADAPT* [DiFrancesco et al. 2011]. ADAPT is a measurement-based heuristic algorithm that dynamically increases/decreases the CSMA/CA parameter values, one at a time, in such a way that the measured delivery ratio remains confined within a region defined by two thresholds D^{low} and D^{high} and above the minimum value D^{min} required by the application. ADAPT also includes a control mechanism to achieve the required reliability also in case of unreliable channel. Each sensor node measures the experienced packet error/loss rate and enables retransmissions if the measured value exceeds a pre-defined threshold D^{loss}.

— *LEAP* [Brienza et al. 2013a]. This is the preliminary versions of JIT-LEAP. It assumes ideal channel conditions and does not rely on a statistical CDT to detect changes.

6.2 Parameter Values and Methodology

Table IV summarizes the operating parameters used in our simulations. Since all the other algorithms (except LEAP) do not consider miss ratio when deriving the optimal setting, the operating parameters for these algorithms have been chosen in such a way to guarantee both D^{min} and M^{max} required by the application. This allows a fair comparison of the considered algorithms in terms of both energy consumption and latency.

Table IV. Operating Parameters

PARAMETER	VALUE
Bit Rate	250 Kbps
Data Frame (Payload) Size	109 (100) bytes
ACK Frame Size	11 bytes
Beacon Order (BO), Superframe Order (SO)	13, 8
$MinBE^{min}$, $MinBE^{max}$	1, 7
$MaxBE^{max}$	10
$MaxCSMABackoffs^{min}$, $MaxCSMABackoff^{max}$	1, 10
$MaxFrameRetries^{min}$, $MaxFrameRetries^{max}$	0, 3
Power Consumption[1] in RX, TX, Idle, Sleep mode	56.4 mW, 52.2 mW, 1.28 mW, 0.06 mW
Training Buffer size (W)	15

In our simulations we considered both ideal and noisy channels. In the latter case, we used the Gilbert-Elliot (GE) model [Gilbert 1960; Elliot 1963] to simulate packet errors/losses as it provides a good approximation of fading in real environments [Willig et al. 2002]. The channel is represented by a continuous-time Markov Chain, consisting of two states, namely *bad* and *good*. In the bad state *no packet* can be successfully delivered, whereas in the good state *all packets* are correctly received. Sojourn times in the two states follow an exponential distribution and their average value determines the average PER experienced during the entire simulation. To derive the model parameters we took an approach similar to [De Pellegrini et al.,

[1] Power consumptions have been derived from the CC2420 datasheet [Texas Instruments 2012], considering a voltage equal to 3 V and a transmission power of 0 dBm.

2006] and [Anastasi et al. 2011] and used values inspired from the real measurements in [Willig et al. 2002]. It turns out that, when the PER is equal to 10% the average sojourn time in the bad and good state is 5.7 and 46.2 ms, respectively. Larger values of the PER are obtained by changing the average sojourn time in the bad state accordingly, while leaving all the other parameters unchanged.

We also assumed that each sensor node generates 10 data packets at every Beacon Interval. All these packets are simultaneously passed down to the MAC layer at the beginning of each Beacon Interval.

For each simulated scenario, we performed 10 independent replications, each consisting of 1000 Beacon Intervals. For each replication we discarded the initial transient interval (10% of the overall duration) during which nodes associate to the PAN coordinator and start generating packets. The results presented below are averaged over all the replications. We also derived confidence intervals using the independent replications method and 95% confidence level. In some cases, the confidence intervals are too small to be observed in the figures.

7. SIMULATION RESULTS

Our analysis is divided into two parts. In the first part, we compare the considered algorithms in *stationary* conditions, i.e., we assume that the operating conditions do not change over time. In the second part we consider *dynamic* scenarios with time-varying operating conditions. Since the two model-based algorithms (and LEAP as well) assume ideal channel conditions, in our analysis – both in stationary and dynamic scenarios – we initially assume that packet errors/losses never occur (i.e., PER = 0). Then, we restrict our analysis to ADAPT and JIT-LEAP only, and investigate the impact of PER on their performance.

In our simulations we assumed that the application requires a packet delivery ratio $D \geq 80\%$ and a miss ratio $M \leq 20\%$, for any sensor node (i.e., $D^{min} = 0.80$ and $D^{max} = 0.20$). Obviously, these thresholds are somehow arbitrary, as they strongly depend on the specific application. However, we performed additional simulations with different thresholds (omitted for space limitations), and we achieved results in line with those presented below.

7.1 Analysis in stationary conditions

As mentioned above, we start our analysis in stationary conditions assuming an ideal channel. Figure 4 shows the delivery ratio and miss ratio of a generic sensor node, for an increasing size of the WSN. All the algorithms satisfy the reliability requirements, both in terms of D and M, with the only exception of LEAP, which exhibits a miss ratio slightly above M^{max}. The offline algorithm provides the highest delivery ratio and the lowest miss ratio. However, it also exhibits the largest latency and energy consumption, as shown in Figure 5. ADAPT provides a delivery ratio between the two thresholds, D^{low} and D^{high}, that have been determined to satisfy also the miss ratio constraint. The model-based adaptive algorithm and JIT-LEAP have similar performance in terms of delivery ratio and miss ratio. However, due to the Controlled Tuning algorithm, JIT-LEAP provides a delivery ratio (miss ratio) very close to D^{min} (M^{max}). This allows JIT-LEAP to reduce the energy consumption, as shown in Figure 5-left. We emphasize that JIT-LEAP is the best solution in terms of energy consumption. In terms of latency (Figure 5-right), the model-based adaptive algorithm has the best performance. This is because the latter algorithm tends to improve the delivery ratio by increasing the number of retransmissions (*macMaxFrameRetries*), whereas the other algorithms achieve the same result by

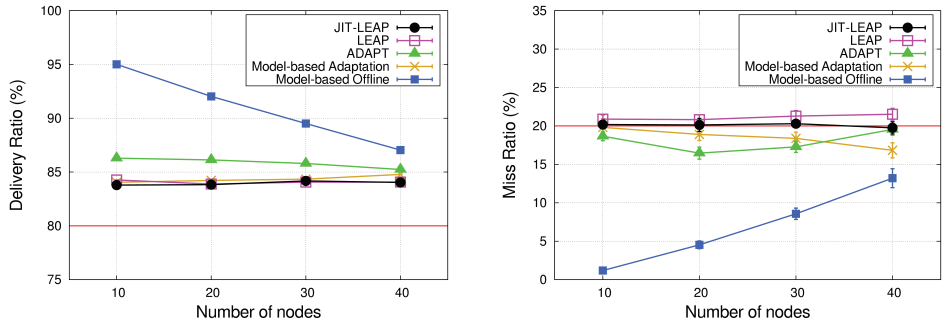

Fig. 4. Delivery ratio (left) and miss ratio (right) vs. number of sensor nodes

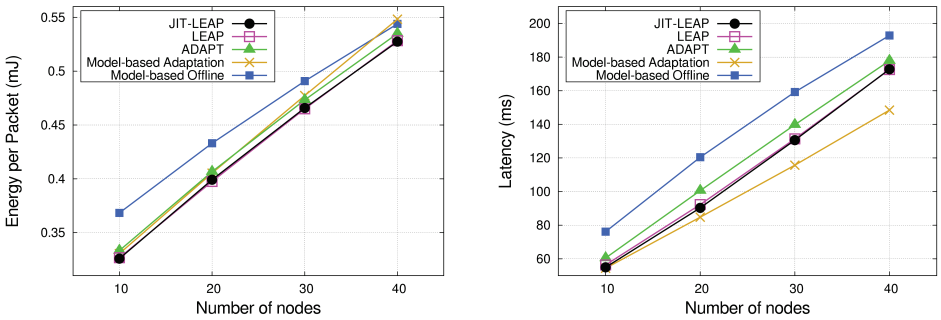

Fig. 5. Average per-packet energy consumption (left), and average latency (right) vs. number of sensor nodes

increasing the number of backoff stages (*macMaxCSMABackoffs*). When a packet is retransmitted, the Backoff Exponent (BE) is re-initialized to its minimum value. Instead, when a new backoff stage is started, the BE is doubled (unless it has reached its maximum value). However, the lower latency introduced by the model-based adaptive algorithm is paid in terms of higher energy consumption. This is because increasing the number of retransmissions is more energy consuming than increasing the number of backoff stages [DiFrancesco et al. 2011]. Finally, JIT-LEAP performs significantly better than both ADAPT and the model-based offline algorithm, also in terms of latency.

Figure 4 and Figure 5 show that LEAP and JIT-LEAP exhibit a similar trend for all the performance indexes. However, LEAP exhibits a higher miss ratio and slightly exceeds the maximum value M^{max}. This is because the mechanism used by LEAP to detect changes in the operating conditions results in more false positives than the CDT used by JIT-LEAP. In our simulations we observed a percentage of wrong detections (vs. the total number of detections) less than 1% for JIT-LEAP and close to 7% for LEAP. This means that, even in stationary conditions, LEAP can erroneously detect changes, thus triggering unnecessary variations in the CSMA/CA parameter setting. Given the higher stability of JIT-LEAP and its accuracy in determining the best set of MAC parameter values, it is more suitable than LEAP for all those critical applications that require minimum guaranteed reliability levels.

In the second set of simulations in stationary conditions, we consider a non-ideal channel. Specifically, we consider different values for the average PER experienced over the simulation run. In this set of simulations the number of sensor nodes is constant and equal to 30. As anticipated, this analysis does not consider both model-based algorithms and LEAP, since they assume ideal channel conditions. Therefore,

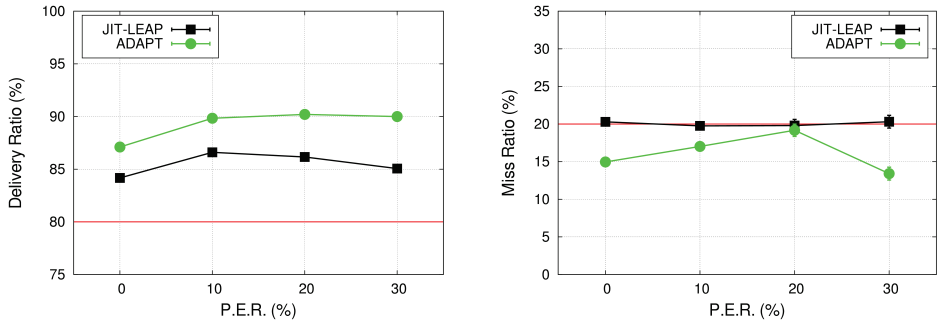

Fig. 6. Delivery ratio (left) and miss ratio (right) vs. Packet Error Rate

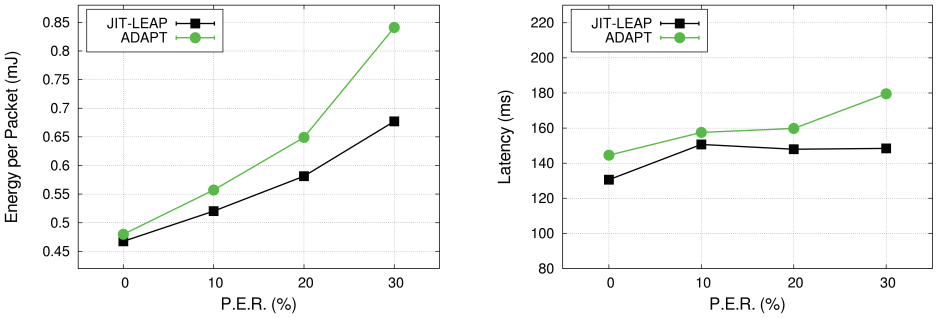

Fig. 7. Average per-packet energy consumption (left), and average latency (right) vs. Packet Error Rate

the analysis is restricted to JIT-LEAP and ADAPT. The capacity of working under lossy channel conditions makes these two solutions much more convenient than the others, since they can be effectively used in real-life scenarios where packet errors/losses occur and PER changes over time. Indeed, Figure 6 shows that both JIT-LEAP and ADAPT satisfy the reliability requirements (in terms of D and M). However, JIT-LEAP outperforms ADAPT in terms of energy consumption and latency. The difference is mainly due to the way the two algorithms estimate the delivery ratio experienced by a sensor node. Specifically, JIT-LEAP relies on a more effective mechanism (as explained in Section 5.3), since – when an acknowledgment is missed – it distinguishes between packet loss/corruption and acknowledgment loss/corruption. Conversely, ADAPT does not consider the effect of lost/corrupted acknowledgements and, thus, it underestimates the delivery ratio experienced by the sensor node. Hence, it tends to use CSMA/CA parameter values higher than necessary, which result in higher energy consumption (up to 20%) and latency (see Figure 7).

7.2 Analysis in dynamic conditions

We now turn our attention to dynamic scenarios, where operating conditions vary over time. We limit our analysis to adaptive algorithms, since the model-based offline algorithm is not suited for such scenarios.

In the first set of simulations, we consider a scenario where the number of active sensor nodes changes over time. We assume that 10 sensor nodes are always active, while 50 more nodes activate and deactivate simultaneously and periodically, every 300 Beacon Intervals. Our goal is to investigate how the different algorithms react to such changes. The results presented below refer to nodes that are always active.

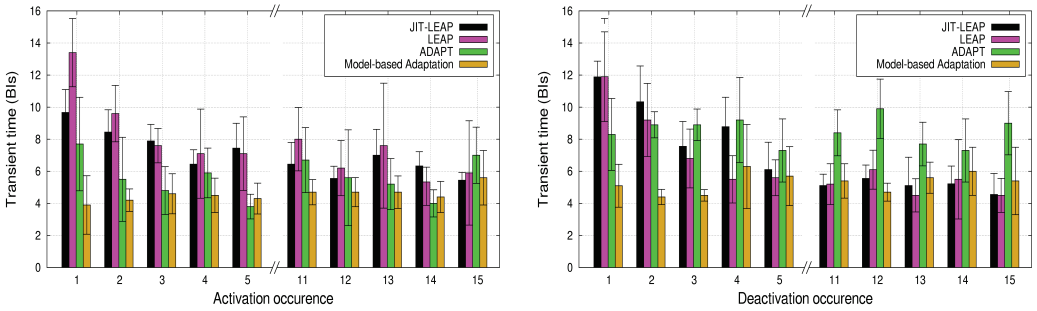

Fig. 8. Transient time when the number of active nodes increases (left) and decreases (right)

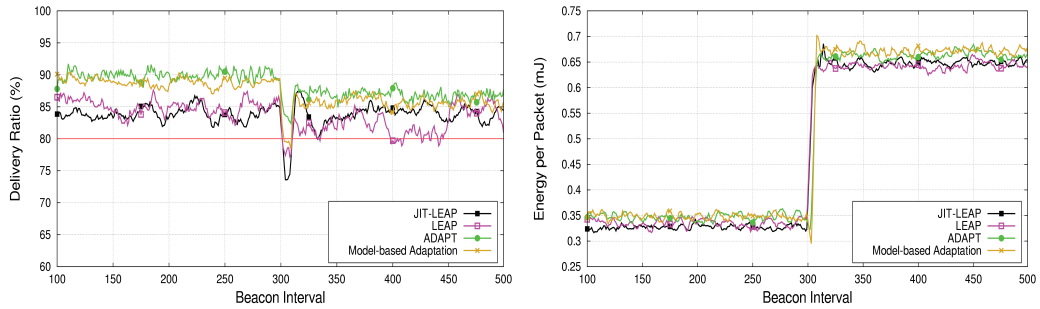

Fig. 9. Delivery ratio (left) and energy per packet (right) when the number of active nodes increases

Figure 8 shows the transient time taken by the various algorithms to adapt to the new operating conditions. We analyzed separately the transient originated by an increase and a decrease in the number of active nodes (left and right side in Figure 8, respectively). As a general remark, transient times experienced by JIT-LEAP and LEAP tend to become shorter and shorter as time elapses, while they remain approximately constant for the other algorithms. This is due to the learning mechanism used by JIT-LEAP and LEAP. When the WSN passes through similar operating conditions experienced in the past, they are able to find out the optimal setting by exploiting the information available in the Learning Table.

Let us now focus on deactivation events (Figure 8-right). After a number of steps, JIT-LEAP and LEAP become significantly faster than the other two algorithms. In ADAPT the transient time depends on the number of active sensor nodes, as the algorithm converges to the optimal setting step by step, and a larger network generally requires higher CSMA/CA parameter values to achieve the same reliability. The model-based adaptive algorithm converges in about 5 Beacon Intervals as the optimal setting is derived by using samples measured in the previous m Beacon Intervals (and we used $m=4$ in our simulations). However, both ADAPT and the model-based adaptive algorithm exhibits some drawbacks. The latter assumes to know in advance the number of (active) sensor nodes in the WSN. This is generally difficult to predict and may become a serious issue in dynamic scenarios. In our simulations, we ran the algorithm using the maximum number of sensor nodes (i.e., 60). This means that, when there are only 10 nodes, the provided setting is not optimal and sensor nodes consume more energy than necessary. Conversely, running the algorithm with the minimum number of nodes (i.e., 10) has negative drawbacks as well. When the number of active nodes is larger, the algorithm may not satisfy the reliability constraints required by the applications. Similarly, ADAPT requires the

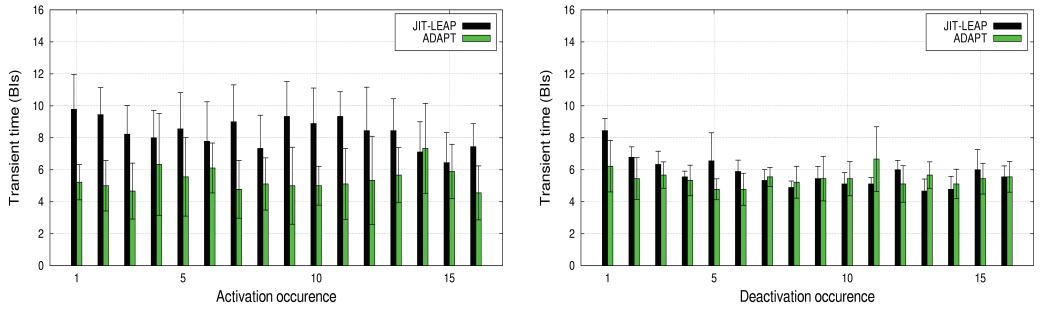

Fig. 10. Transient time when the Packet Error Rate increases (left) and decreases (right)

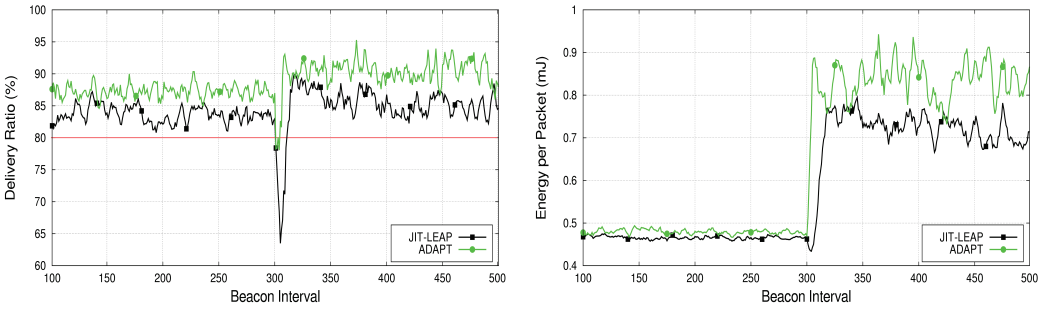

Fig. 11. Delivery ratio (left) and energy per packet (right) when the Packet Error Rate increases

definition of the two thresholds D^{low} and D^{high} that strictly depend both on the reliability requirements (M^{max} and D^{min}) and the network congestion. When the number of sensor nodes increases, the two thresholds should take larger values in order to guarantee the same levels of D and M. As above, in our simulations, we referred to the worst case (60 active nodes) to define the threshold values, so as to satisfy the reliability constraints both with 10 and 60 nodes. JIT-LEAP and LEAP do not suffer from such limitations, as they do not require any input parameter. This leads to significant benefits, especially in terms of energy consumption. Figure 9 compares the delivery ratio (left) and energy consumption (right) of the different algorithms, before and after an increase in the number of sensor nodes has occurred. JIT-LEAP and LEAP are characterized by a delivery ratio that is the closest to the application requirement (80%), and provide the lowest energy consumption. This is because they use a more appropriate (i.e., lower) parameter setting than the other algorithms. For the same reason, when the number of sensor nodes changes abruptly, the parameter setting used by JIT-LEAP and LEAP is the least appropriate to the new conditions. Hence, they experience the biggest drop in the delivery ratio, as shown in Figure 9-left.

As a final remark, we need to point out that JIT-LEAP and LEAP exhibit similar performance under ideal channel conditions, both in terms of transient time and energy consumption. However, once again, the introduction of the CDT makes JIT-LEAP much more stable in the choice of parameter setting and, thus, more reliable. As shown in Figure 9-left, in the interval 400-450, LEAP moves to a non-appropriate setting, thus temporarily violating the reliability requirements, due to some false positives in the detection of changes. Conversely, the problem does not occur with JIT-LEAP.

In the second set of simulations we consider a scenario where the average PER

changes over time during the simulation run (conversely, the number of sensor nodes is constant and equal to 30). Specifically, we assume that PER changes periodically (every 300 Beacon Intervals) and abruptly, from 0% to 30% and vice versa. Similarly to the analysis in stationary conditions with non-ideal channel, this part of the analysis is limited to ADAPT and JIT-LEAP. As shown in Figure 10, ADAPT exhibits shorter transient times than JIT-LEAP. This difference can be explained as follows. In ADAPT retransmissions are generally disabled ($macMaxFrameRetries = 0$) and are enabled only when a packet error/loss rate larger than D^{loss} [2] is experienced. When this occurs *macMaxFrameRetries* is set to the maximum value allowed by ADAPT (i.e., 3). Obviously, such an approach makes ADAPT very reactive. However, since retransmissions are very energy consuming [DiFrancesco et al. 2011], this approach also introduces a larger energy consumption (see Figure 11-right). On the contrary, JIT-LEAP derives the exact value of *macMaxFrameRetries* in order to satisfy the reliability constraints required by the application. In addition, as explained above, ADAPT tends to underestimate the delivery ratio experienced by the sensor node, thus consuming more energy than necessary. For all these reasons JIT-LEAP matches the application requirement (D^{min}) more closely than ADAPT and outperforms ADAPT in terms of energy efficiency, as shown in Figure 11.

7.3 Resource Usage

We conclude our analysis looking at the computational resource usage required by the considered algorithms. As a preliminary remark, we observe that the model-based offline algorithm does not require any computational/memory resource, since it is run offline. Thus, we will focus on the remaining algorithms.

In terms of computational cost, ADAPT is the lightest one, as it only requires few simple operations to update the estimates. For the model-based adaptive algorithm, the authors suggest two implementations. In the first one (used in our simulations) the optimal setting is obtained by solving the analytical model at the sensor node, thus resulting in a significant computational load. In the second implementation, the optimal parameter values are computed offline and stored on the sensor node in a look-up table. Obviously, the latter approach requires no computational cost but introduces a high memory occupancy. Finally, JIT-LEAP is particularly suitable to be executed on sensor nodes. Like ADAPT and LEAP, it has a lightweight Adaptive Tuning phase. The CDT is a light task as well, as it requires few simple calculations over the state variables. Only the CDT training and CPM are a bit more computationally intensive operations, however they are performed only when a change is detected. Hence, they introduce a slight additional computational load just in a small fraction of Beacon Intervals.

Table V - Memory occupancy

JIT-LEAP	LEAP	ADAPT	MODEL-BASED ADAPTATION		OFFLINE COMPUTATION
			Online computation	Lookup table	
≤ 1 KByte	≤ 1 KByte	≈ 10 Bytes	≈ 100 Bytes	1 - 10 KBytes	0 Bytes

Let us now analyze the memory footprint. Table V shows the memory occupancy of the considered algorithms. Among the adaptive algorithms, ADAPT exhibits the smallest footprint, since it needs to store only some statistics and estimates. For the model-based adaptive algorithm, both versions require the node to store the

[2] In our simulations we considered $D^{loss} = 1 - (D^{low} + D^{high})/2$.

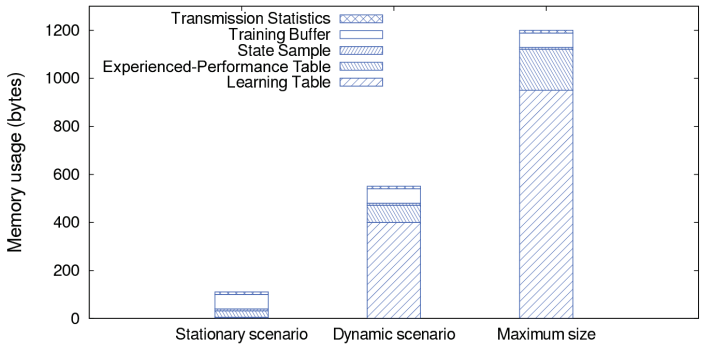

Fig. 12. Memory occupancy in JIT-LEAP

measured congestion indexes (which takes about 100 bytes). When the computation of the optimal set is carried out offline, the memory occupancy is much higher, due to the lookup table. The memory occupancy of JIT-LEAP and LEAP is similar and strongly depends on the considered scenario, as the algorithms store data about each used parameter set and experienced operating condition. More specifically, in JIT-LEAP, as shown in Figure 12, the Learning Table is the more consuming data structure and its size increases if the operating conditions change frequently. Let M denote the maximum number of elements for each entry, and N the maximum number of entries (i.e. the number of possible parameter sets). Hence, the size S of the Learning Table is $S = N \cdot M \cdot E$, where E denotes the size of each element. In JIT-LEAP N is constant and equal to 19, while $E = 5$ bytes. Assuming $M = 10$, it yields $S = 950$ bytes. Figure 12 shows the memory space required by each data structure. In our simulations, both in stationary and dynamic scenarios, the observed footprint of JIT-LEAP was well below 1 Kilobyte.

8. CONCLUSIONS

In this paper we have proposed a new Just-in-Time learning-based algorithm, called JIT-LEAP, for deriving the optimal CSMA/CA parameter setting in IEEE 802.15.4 sensor networks. The proposed algorithm adapts the CSMA/CA parameters so as to satisfy the reliability constraints required by the application, with minimum energy consumption, on the basis of the reliability experienced by the sensor nodes. Unlike many similar adaptive algorithms, it exploits a learning mechanism to speed up the transient time when the network operating conditions have been already experienced in the past. JIT-LEAP extends a previous learning-based algorithm (LEAP) by explicitly considering packet errors/losses and introducing a statistical test for detecting changes in the operating conditions. We have analyzed our algorithm both in stationary and dynamic scenarios. Our results show that JIT-LEAP behaves better than the other algorithms in the literature, since it allows to satisfy the reliability requirements of applications while providing the minimum energy consumption, both for ideal and lossy channel.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

This work has been partially supported by the University of Pisa, in the framework of the PRA 2015 program. Special thanks to Cesare Alippi for his help and advice.

REFERENCES

Cesare Alippi. 2014. *Intelligence for Embedded Systems*. Springer.

Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. 2011. A hierarchical, nonparametric, sequential change-detection test. In *Proceedings of the 2011 IEEE International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2889–2896.

Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. 2013. Just-in-time classifiers for recurrent concepts. *IEEE Transactions on Neural Networks and Learning Systems* 24, 4 (2013), 620–634.

Cesare Alippi and Manuel Roveri. 2008. Just-in-time adaptive classifiers – Part I: Detecting non-stationary changes. *IEEE Transactions on Neural Networks* 19, 7 (2008), 1145–1153.

Giuseppe Anastasi, Eleonora Borgia, Marco Conti, Enrico Gregori, and Andrea Passarella. 2005. Understanding the real behavior of Mote and 802.11 ad hoc networks: an experimental approach. *Pervasive and Mobile Computing* 1, 2 (2005), 237–256.

Giuseppe Anastasi, Marco Conti, and Mario DiFrancesco. 2011. A comprehensive analysis of the MAC unreliability problem in IEEE 802.15.4 wireless sensor networks. *IEEE Transactions on Industrial Informatics* 7, 1 (2011), 52–65.

Giuseppe Anastasi, Marco Conti, Mario DiFrancesco, Andrea Passarella. 2009. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks* 7, 3 (May 2009), 537–568.

M. S. Bartlett and D. G. Kendall. 1946. The statistical analysis of variance – heterogeneity and the logarithmic transformation. *Supplement to the Journal of the Royal Statistical Society* 8, 1, 128–138.

Michèle Basseville, Igor Nikiforov, and others. 1993. *Detection of abrupt changes: theory and application*. Vol. 104. Prentice Hall Englewood Cliffs.

Giacomo Boracchi, Michalis Michaelides, and Manuel Roveri. 2014. A Cognitive Monitoring System for Contaminant Detection in Intelligent Buildings. In *Proceedings of the 2014 IEEE International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Giacomo Boracchi and Manuel Roveri. 2014. A reconfigurable and element-wise ICI-based change-detection test for streaming data. In *Proceedings of the 2014 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA 2014)*. IEEE.

Bruno Bougard, Francky Catthoor, Denis C Daly, Anantha Chandrakasan, and Wim Dehaene. 2008. Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: Modeling and improvement perspectives. In *Design, Automation, and Test in Europe*. Springer, 221–234.

Simone Brienza, Domenico DeGuglielmo, Cesare Alippi, Giuseppe Anastasi, and Manuel Roveri. 2013a. A Learning-based Algorithm for Optimal Mac Parameter Setting in IEEE 802.15.4 Wireless Sensor Networks. In *Proceedings of the 10th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks (PE-WASUN'13)*. ACM, New York, NY, USA, 73–80.

Simone Brienza, Domenico DeGuglielmo, Giuseppe Anastasi, Marco Conti, and Vincenzo Neri. 2013b. Strategies for optimal MAC parameter setting in IEEE 802.15.4 wireless sensor networks: A performance comparison. In *Proceedings of the 2013 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 000898–000903.

Francesco De Pellegrini, Daniele Miorandi, Stefano Vitturi, and Andrea Zanella. 2006. On the Use of Wireless Networks at Low Level of Factory Automation. *IEEE Transactions on Industrial Informatics* 2, 2 (May 2006), 129.143.

Mario DiFrancesco, Giuseppe Anastasi, Marco Conti, Sajal K Das, and Vincenzo Neri. 2011. Reliability and Energy-Efficiency in IEEE 802.15.4/ZigBee Sensor Networks: An Adaptive and Cross-Layer Approach. *Selected Areas in Communications, IEEE Journal on* 29, 8 (2011), 1508–1524.

Edwin O. Elliot. 1963. Estimates of Error Rates for Codes on Burst-Noise Channels. Bell system technical journal 42.5 (1963), 1977-1997.

Edgar N. Gilbert. 1960. Capacity of a Burst-Noise Channel. Bell system technical journal 39.5 (1960), 1253-1265.

HART Communication Foundation Std. 2007. *HART Field Communication Protocol Specification*.

Jan-Hinrich Hauer. 2009. *TKN15.4: An IEEE 802.15.4 MAC implementation for TinyOS*. TKN Technical Report TKN-08-003. Telecommunication Networks Group, Technical University Berlin.

Douglas Hawkins, Qiu Peihua, and Wook Kang Chang. 2003. The changepoint model for statistical process control. *Journal of quality technology* 35, 4 (2003), 355–366.

IEEE Computer Society. 2006. *IEEE Standard for Information technology, Part 15.4; Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LRWPANs)*.

ISA, International Society of Automation. 2009. *Standard ISA-100.11a, Wireless Systems for Industrial Automation: Process Control and Related Applications*.

Yoshinobu Kawahara and Masashi Sugiyama. 2012. Sequential change-point detection based on direct density-ratio estimation. *Statistical Analysis and Data Mining* 5, 2 (2012), 114–127.

Bilal Muhammad Khan, Falah H Ali, and Elias Stipidis. 2010. Improved backoff algorithm for IEEE 802.15.4 wireless sensor networks. In *Proceedings of the 2010 IFIP Wireless Days (WD)*. IFIP/IEEE, 1–5.

Mounib Khanafer, Mouhcine Guennoun, and Hussein Mouftah. 2011. An Efficient Adaptive Backoff Algorithm for Wireless Sensor Networks. In *Proceedings of the 2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)*. IEEE, 1–6.

Mounib Khanafer, Mouhcine Guennoun, and Hussein T. Mouftah. 2014. A Survey of Beacon-Enabled IEEE 802.15.4 MAC Protocols in Wireless Sensor Networks. *Communications Surveys Tutorials, IEEE* 16, 2 (Second 2014), 856–876. 1553-877X

Meejoung Kim and Chul-Hee Kang. 2010. Priority-Based Service-Differentiation Scheme for IEEE 802.15.4 Sensor Networks in Nonsaturation Environments. *Vehicular Technology, IEEE Transactions on* 59, 7 (Sept 2010), 3524–3535. 0018-9545

Anis Koubaa, Ricardo Severino, Màrio Alves, and Eduardo Tovar. 2009. Improving Quality-of-Service in Wireless Sensor Networks by Mitigating Hidden-Node Collisions. *Industrial Informatics, IEEE Transactions on* 5, 3 (Aug 2009), 299–313. 1551-3203 http://dx.doi.org/10.1109/TII.2009.2026643

Younggoo Kwon and Yohan Chae. 2006. Traffic Adaptive IEEE 802.15.4 MAC for Wireless Sensor Networks. In *Embedded and Ubiquitous Computing*, Edwin Sha, Sung-Kook Han, Cheng-Zhong Xu, Moon-Hae Kim, Laurence T. Yang, and Bin Xiao (Eds.). Lecture Notes in Computer Science, Vol. 4096. Springer Berlin Heidelberg, 864-873.

Jongwook Lee, Jae Yeol Ha, Joseph Jeon, Dong Sung Kim, and Wook Hyun Kwon. 2007. ECAP: A bursty traffic adaptation algorithm for IEEE 802.15.4 Beacon-Enabled networks. In *Proceedings of the 65th IEEE Vehicular Technology Conference (VTC2007-Spring)*. IEEE, 203–207. DOI:http://dx.doi.org/10.1109/VETECS.2007.54

Seung-Youn Lee, Youn-Soon Shin, Jong-Suk Ahn, and Kang-Woo Lee. 2009. Performance analysis of a non-overlapping binary exponential backoff algorithm over IEEE 802.15.4. In *Proceedings of the 4th International Conference on Ubiquitous Information Technologies & Applications (ICUT'09)*. IEEE, 1–5.

Yves Lepage. 1974. A combination of Wilcoxon's and Ansari-Bradley's statistics. *Biometrika* 58, 1 (Apr. 1974), 213–217.

Philip Levis, David Gay, Vlado Handziski, Jan-Hinrich Hauer, Ben Greenstein, Martin Turon, Jonathan Hui, Kevin Klues, Cory Sharp, Robert Szewczyk, Joe Polastre, Philip Buonadonna, Lama Nachman, Gilman Tolle, David Culler, and Adam Wolisz. 2005. *T2: A second generation OS for embedded sensor networks*. TKN Technical Report TKN-05-007. Telecommunication Networks Group, Technical University Berlin.

Henry B. Mann and Donald R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics* 18 (1947), 50–60.

A. M. Mood. 1954. On the asymptotic efficiency of certain nonparametric two-sample tests. *The Annals of Mathematical Statistics* 25, 3 (Sep. 1954), 514–522.

Chewoo Na, Yaling Yang, and Amitabh Mishra. 2008. An optimal GTS scheduling algorithm for time-sensitive transactions in IEEE 802.15. 4 networks. *Computer Networks* 52, 13 (2008), 2543–2557.

Mario Neugebauer, Jörn Plönnigs, and Klaus Kabitzsch. 2005. A new beacon order adaptation algorithm for IEEE 802.15.4 networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks*. IEEE, 302–311.

Ns-2, Network Simulator. http://www.isi.edu/nsnam/ns.

Pangun Park, Piergiuseppe DiMarco, Carlo Fischione, and Karl Henrik Johansson. 2013. Modeling and optimization of the IEEE 802.15.4 protocol for reliable and timely communications. *IEEE Transactions on Parallel and Distributed Systems* 24, 3 (2013), 550–564.

Pangun Park, Piergiuseppe DiMarco, Pablo Soldati, Carlo Fischione, and Karl Henrik Johansson. 2009. A generalized Markov chain model for effective analysis of slotted IEEE 802.15.4. In *Proceedings of the 6th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'09)*. IEEE, 130–139.

Sofie Pollin, Mustafa Ergen, Sinem Ergen, Bruno Bougard, Liesbet Van der Perre, Ingrid Moerman, Ahmad Bahai, Pravin Varaiya, and Francky Catthoor. 2008. Performance analysis of slotted carrier sense IEEE 802.15.4 medium access layer. *IEEE Transactions on Wireless Communications* 7, 9 (2008), 3359–3371.

VaddinaPrakash Rao and Dimitri Marandin. 2006. Adaptive Backoff Exponent Algorithm for Zigbee (IEEE 802.15.4). In *Next Generation Teletraffic and Wired/Wireless Advanced Networking*, Yevgeni Koucheryavy, Jarmo Harju, and VillyB. Iversen (Eds.). Lecture Notes in Computer Science, Vol. 4003. Springer Berlin Heidelberg, 501–516. 978-3-540-34429-2

Gordon Ross, Dimitris Tasoulis, and Niall Adams. 2011. Nonparametric monitoring of data streams for changes in location and scale. *Technometrics* 53, 4 (2011), 379–389.

Shiann-Tsong Sheu, Yun-Yen Shih, and Wei-Tsong Lee. 2009. CSMA/CF Protocol for IEEE 802.15.4 WPANs. *IEEE Transactions on Vehicular Technology* 58, 3 (2009), 1501-1516. DOI:http://dx.doi.org/10.1109/TVT.2008.928634

Chandramani Kishore Singh, Anurag Kumar, and P. M. Ameer. 2008. Performance Evaluation of an IEEE 802.15.4 Sensor Network with a Star Topology. *Wireless Networks* 14, 4 (Aug. 2008), 543–568.

Alexander Tartakovsky, Boris Rozovskii, Rudolf Blažek, and Hongjoong Kim. 2006. Detection of intrusions in information systems by sequential change-point methods. *Statistical Methodology* 3, 3 (2006), 252–293.

Texas Instruments. 2012. CC2420 2.4GHz IEEE 802.15.4/ZigBee ready RF Transceiver. Retrieved from http://focus.ti.com/lit/ds/symlink/cc2420.pdf.

Tmote Sky Platform, MoteIV Corporation. http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf

Andreas Willig, Martin Kubisch, Christian Hoene, and Adam Wolisz. 2002. Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer. *IEEE Transactions on Industrial Electronics* 49, 6 (2002), 1265–1282. DOI:http://dx.doi.org/10.1109/TIE.2002.804974

Kiran Yedavalli and Bhaskar Krishnamachari. 2008. Enhancement of the IEEE 802.15.4 MAC protocol for scalable data collection in dense sensor networks. In *Proceedings of the 6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops (WiOPT 2008)*. 152–161.

Xiaodong Zhao, Wan Zhang, Wensheng Niu, Yadi Zhang, and Liqiang Zhao. 2010. Power and Bandwidth Efficiency of IEEE 802.15.4 Wireless Sensor Networks. In *Ubiquitous Intelligence and Computing*, Zhiwen Yu, Ramiro Liscano, Guanling Chen, Daqing Zhang, and Xingshe Zhou (Eds.). Lecture Notes in Computer Science, Vol. 6406. Springer Berlin Heidelberg, 243–251. 978-3-642-16354-8

ZigBee Alliance. 2007. *The ZigBee Specification version 1.0.*

Richard Zurawski. 2009. *Networked embedded systems.* CRC press Boca Raton, FL, USA.

Online Appendix to: Just-in-Time Adaptive Algorithm for Optimal Parameter Setting in 802.15.4 WSNs

SIMONE BRIENZA, University of Pisa
MANUEL ROVERI, Politecnico di Milano
DOMENICO DE GUGLIELMO, University of Pisa
GIUSEPPE ANASTASI, University of Pisa

A. PROOF OF CLAIM 1

CLAIM 1. Assuming that *(i)* packet transmission errors are independent from each other, and *(ii)* the PER is the same for both data packets and acknowledgements, then

$$\alpha = 1 - \left(\frac{F - PER}{(1 - PER)F}\right)^{macMAXFrameRetries+1}$$

where F denotes the transmission failure ratio, i.e., the probability that a data packet transmission fails for any reason.

PROOF. Under non-ideal channel conditions, a packet transmission failure is experienced by the sensor node if one of the following disjoint events occur: (**i**) the data packet experiences a collision; (**ii**) no collision occurs but the transmitted data packet is corrupted by the channel; (**iii**) the data packet is received correctly by the sink but the corresponding acknowledgment is corrupted by the channel. Let p_{coll}, p_{txfail}, and $p_{ACKfail}$ denote the probability of event (**i**), (**ii**) and (**iii**), respectively. Hence, the following equations immediately follow.

$$\begin{cases} F = p_{coll} + p_{txfail} + p_{ACKfail} \\ p_{txfail} = (1 - p_{coll}) \cdot PER \\ p_{ACKfail} = (1 - p_{coll}) \cdot (1 - PER) \cdot PER \end{cases}$$

Solving the previous system yields $p_{ACKfail} = \frac{(1-F)\cdot PER}{(1-PER)}$. If a data packet transmission is not acknowledged, the probability that the packet has been received correctly by the sink is equal to $\beta = \frac{p_{ACKfail}}{R_F} = \frac{(1-F)\cdot PER}{(1-PER)\cdot F}$. Therefore, if a packet is dropped after *macMaxFrameRetries* retransmissions, the probability that it has been correctly received by the sink is equal to

$$\alpha = 1 - (1 - \beta)^{macMAXFrameRetries+1}.$$ □

B. CONTROLLED TUNING ALGORITHM

In this Appendix we detail the Controlled Tuning algorithm introduced in Section 5. Let j be the identified parameter setting and let $D(i)$ and $M(i)$ be the delivery ratio and miss ratio statistics contained in the Experienced-Performance Table for set

DOI:http://dx.doi.org/XXXXXXX

i, (for $i = j-1,\ j,\ j+1$), provided that an entry for set i is available in the Experienced-Performance Table. The Controlled Tuning algorithm checks whether the reliability constraints are satisfied (line 4). If both reliability indexes are satisfied by using setting j and statistics are available in the Experienced-Performance Table for setting $j-1$, then the algorithm performs a fine tuning between sets j and $j-1$ (lines 6-10). Specifically, it computes the distance of $D(j)$ and $M(j)$ from the corresponding threshold (i.e., D^{min} and M^{max}, respectively) and considers the index closer to the threshold. Then, the new parameter set $\boldsymbol{par}_{next}$, to be used in the next Beacon Interval, is changed from j to $j-1$ with a probability proportional to the distance of the considered index from its threshold (line 10).

On the contrary, when at least one of the two reliability indexes does not meet the application requirements, the algorithm performs a similar fine tuning between sets j and $j+1$ (lines 12-17). If no entry is available for set $j+1$ in the Experienced-Performance Table the algorithms returns $j+1$ as the set to be used in the next Beacon Interval (line 12). Otherwise, as described above, the distance between $D(j)$ and $M(j)$ and the corresponding thresholds is calculated but, in this case, the index with the larger distance is considered and the set to be used in the next Beacon Interval is changed from j to $j+1$ with a probability proportional to the latter distance (line 17).

ALGORITHM 1. *Controlled Tuning* algorithm

```
1  for i = j − 1 to j + 1
2    D(i) = Cluster[i].D;    M(i) = Cluster[i].M;
3  end for
4  if ((D(j) ≥ D^min) and (M(j) ≤ M^max))
5    if (Cluster [j − 1] = NULL) return j − 1;
6    else
7      p_D = (D(j) − D^min)/(D(j) − D(j − 1));
8      p_M = (M^max − M(j))/(M(j − 1) − M(j));
9      p = min(p_D, p_M);
10     return j (j − 1) with probability: (1 − p) p
11 else
12   if (Cluster [j + 1] = NULL) return j + 1;
13   else
14     p_D = (D^min − D(j))/(D(j + 1) − D(j));
15     p_M = (M(j) − M^max)/(M(j) − M(j + 1));
16     p = max(p_D, p_M);
17     return j (j + 1) with probability: (1 − p) p
```

C. EXPERIMENTAL RESULTS

To validate the simulation results shown in Section 7 we also implemented JIT-LEAP on Tmote Sky motes [Tmote Sky], using the TinyOS 2.1 operating system [Levis et al. 2005], and performed some experiments in a real testbed. Tmote Sky sensor nodes use the Chipcon CC2420 radio transceiver [Texas Instruments 2012] that is compliant to the 802.15.4 physical layer and supports a 250 Kb/s bit rate over the unlicensed 2.4 GHz ISM band. As an implementation of the 802.15.4 MAC protocol, we used TKN15.4 [Hauer 2009] that is provided with the TinyOS software. In our experimental analysis, we referred to the same star network scenario and

common parameter setting considered in the simulation analysis (see Section 7). We ran our experiments in a working place with several sources of interfering signals, including many WiFi networks. Regarding the Packet Error Rate, we measured the average value experienced by sensor nodes during an experiment and used the same value in the corresponding simulation. Figure 13 compares simulation and experimental results, in terms of delivery ratio and miss ratio. The comparison show that simulation and experimental results match very closely.

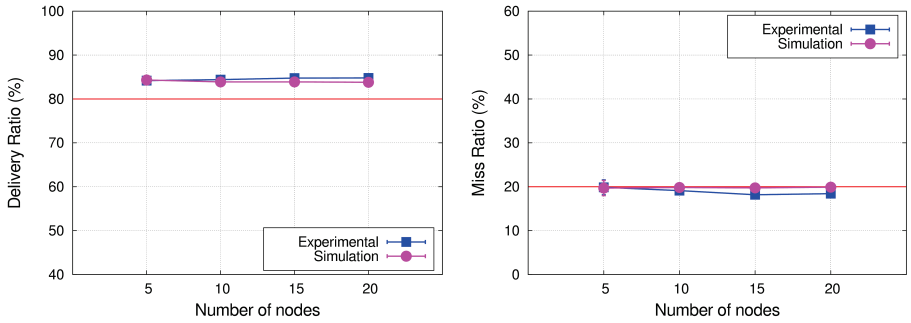

Fig. 13. Comparison between simulation and experimental results, in terms of delivery ratio (left) and miss ratio (right)