A Learning-based Algorithm for Optimal MAC Parameters Setting in IEEE 802.15.4 Wireless Sensor Networks

Simone Brienza[1], Domenico De Guglielmo[1], Cesare Alippi[2], Giuseppe Anastasi[1], Manuel Roveri[2]

[1]Dept. of Information Engineering
University of Pisa, Italy
{firstname.lastname}@iet.unipi.it

[2] Dip.di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Italy
{lastname}@elet.polimi.it

ABSTRACT

Recent studies have shown that the IEEE 802.15.4 MAC protocol may suffer from severe limitations, in terms of reliability and energy efficiency, if CSMA/CA parameter settings are not appropriate. On the other hand, choosing the optimal setting that guarantees the application requirements with minimum energy consumption, may not be a trivial task in a real environment, where the operating conditions change over time. In this paper we propose a LEarning-based Adaptive Parameter tuning (LEAP) algorithm that, in addition to adapting the CSMA/CA parameter settings to the time-variant operating conditions, also exploits the past history to figure out the most appropriate settings for the current conditions. Simulation results show that, in stationary conditions, the performance of the proposed algorithm is very close to an ideal (but unfeasible) algorithm. It is shown that LEAP is able to select in dynamic scenarios the optimal settings faster than related algorithms.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network protocols.

General Terms

Algorithms, Performance, Reliability.

Keywords

IEEE 802.15.4, CSMA/CA, Energy Efficiency, Reliability, MAC Parameters Tuning.

1. INTRODUCTION

Wireless sensor networks (WSNs) are increasingly perceived as an effective technology for developing distributed sensing systems in many application domains, ranging from environmental monitoring to logistics, from healthcare to industrial applications, from location/tracking to automatic building management. This trend is also pushed by two communication standards available for WSNs, namely the IEEE 802.15.4 standard [1] and ZigBee specifications [2], defining the low and high layers of the protocol stack, respectively.

Typically, *energy efficiency* is considered the major requirement in the design of WSN-based sensing systems. However, in many application scenarios, additional requirements need to be considered, such as *reliability* and *timeliness* [3]. In this respect, several studies have highlighted that IEEE 802.15.4-based sensor networks suffer from severe limitations in terms of *communication reliability* (i.e., *packet delivery probability*) and timeliness (i.e., *packet latency*), which make them unsuitable in many application scenarios [4][5][6][7]. The reason for such limitations is the CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) algorithm used by the 802.15.4 MAC protocol for channel access. As any other random access scheme, the 802.15.4 CSMA/CA algorithm is not able to solve contentions when the number of contending sensor nodes is large, thus resulting in significant collision and packet dropping rates. However, in the 802.15.4 MAC protocols this problem is even more severe than in other CSMA/CA-based MAC protocols (e.g., S-MAC [8]) due to the default settings suggested by the standard. In a previous paper [7] we have shown, both by relying to simulations and experimental measurements, that the CSMA/CA parameters suggested by the standard are not appropriate, even when the number of sensor nodes is low. However, the packet delivery probability can increase by using larger CSMA/CA parameters, at the cost of an increased latency and energy consumption.

Ideally, the CSMA/CA parameter setting should be chosen to guarantee the reliability (and latency) level required by the application, while minimizing the energy consumption at sensor nodes. However, identification of such optimal parameter setting is not a trivial task in real sensor networks, as reliability and latency strongly depends on time-varying factors such as the number of sensor nodes, the offered load and the packet error rate. In addition, these factors cannot be controlled or predicted. Hence, a number of strategies have been proposed, to derive the optimal CSMA/CA parameters in 802.15.4 sensor networks. The proposed approaches range from *offline computation* [9], to *model-based adaptation* [10] and *measurement-based adaptation* [11]. A simulation-based comparative analysis of the previous strategies has been carried out in [12].

Model-based strategies rely on an analytical model of the sensor network. They use the analytical model to derive the optimal parameter setting, depending on the operating conditions. This can be done either by solving the exact analytical model [9], or using a simplified version of the model to dynamically adapt to time-variant operating conditions [10]. In the former case the model must be solved at the sink node as it requires a significant amount of computation resources; in the latter case the simplified analytical model can be solved directly at sensor nodes. In both cases, the effectiveness of the algorithm in providing the optimal setting depends on the accuracy of the underlying model. In addition, the analytical model requires knowing the number of sensor nodes and other input parameters that are generally unknown and/or difficult to predict.

An alternate approach consists in using a measurement-based strategy as in [11], where the *ADaptive Access Parameters Tuning*

The work of D. De Guglielmo is supported by the Tuscany Region under the Pegaso project.

PE-WASUN'13, November 04 - 08 2013, Barcelona, Spain

http://dx.doi.org/10.1145/2507248.2507251

(ADAPT) algorithm is proposed. ADAPT is a heuristic algorithm that dynamically adapts the CSMA/CA parameters to the current operating conditions, based on local measurements of the packet delivery probability. Specifically, CSMA/CA parameters are increased/decreased depending on the packet delivery probability experienced by the sensor node. ADAPT does not require any input parameter (beyond the minimum packet delivery probability required by the applications) and can be used in any real scenario [11]. However, it still has a couple of limitations, as emphasized in [12]. Specifically, due to its nature, it tends to oscillate between adjacent CSMA/CA parameter sets and never stabilizes on a given setting, thus consuming more energy than necessary. In addition, it is memory-less and does not take advantage of a recurrent state loading procedure. As a consequence, if the same operating conditions repeat over time, the algorithm is not able to immediately set the optimal parameter values, but takes time to iteratively converge to them.

In this paper we propose a *LEarning-based Adaptive Parameter tuning* (LEAP) algorithm, that is adaptive and relies on local measurements, like ADAPT. However, it also learn the past history to select the optimal parameter settings and avoid unnecessary energy wastes. Like ADAPT, neither makes any assumption on the channel conditions, nor requires input parameters. We show, by simulation, that LEAP outperforms all mentioned algorithms and behaves very closely to an ideal, but unfeasible, algorithm for optimal CSMA/CA parameter setting.

The rest of the paper is organized as follows. Section II describes the 802.15.4 standard. Section III presents a formal formulation of the problem that we intend to face. Section IV presents the LEAP algorithm. Section V describes the simulation setup, while the simulation results are discussed in Section VI. Finally, conclusions are drawn in Section VII.

2. 802.15.4 MAC PROTOCOL

IEEE 802.15.4 [1] is a standard for low-rate, low-power, and low-cost Personal Area Networks (PANs). The standard supports both *star* (single-hop), and *peer-to-peer* (multi-hop) network topologies, and defines two different methods for regulating the channel access, namely a *beacon enabled* and a *non-beacon enabled* mode. In this paper, we focus on the beacon-enabled mode only, which provides a power management mechanism, based on a duty cycle approach. Moreover, for the sake of simplicity and without loss of generality, we refer in the sequel to a star topology, consisting of one coordinator and a number of sensor nodes. However, the beacon enabled mode can also be used in a peer-to-peer topology.

In beacon enabled mode the coordinator node periodically transmits special messages called *beacons*, that define a superframe structure and are used by sensor nodes for synchronization. The time interval between two consecutive beacons is called *Beacon Interval (BI)*, and is defined through the *Beacon Order* (*BO*) parameter ($BI = 15.36 \cdot 2^{BO}$ ms, $0 \leq BO \leq 14$). Each superframe consists of an *Active Period* and an *Inactive Period*. In the Active Period nodes communicate with the coordinator, while during the Inactive Period they enter a low-power sleep mode to save energy. The size of the Active Period is denoted as *Superframe Duration* (*SD*) and is defined through the *Superframe Order* (*SO*) parameter ($SD = 15.36 \cdot 2^{SO}$ ms, with $0 \leq SO \leq BO \leq 14$). The Active Period is further divided into a *Contention Access Period* (*CAP*) and a *Collision Free Period* (*CFP*). During the CAP a slotted CSMA/CA algorithm is used for channel access, while in the CFP communication occurs in a TDMA (Time Division Multiple Access) style by using a number of *Guaranteed Time Slots* (GTSs), assigned to individual nodes.

2.1 CSMA/CA Algorithm

In the beacon-enabled mode a slotted CSMA/CA scheme is used in the *CAP*, i.e., all operations are aligned to backoff slots (whose duration is 320 μs). Upon receiving a data frame to be transmitted, the CSMA/CA algorithm performs the following steps.

1. A set of state variables is initialized, i.e. the *contention window size* (*CW* = 2), the *number of backoff stages* for the ongoing transmission (*NB* = 0), and the *backoff exponent* (*BE* = *macMinBE*).
2. A random backoff time, uniformly distributed in the range [0, 2^{BE} - 1] · 320 μs , is generated to initialize the *backoff timer*. Then, the sensor node waits until this timer expires.
3. A *Clear Channel Assessment* (CCA) is performed to check the state of the wireless medium.
   a. If the medium is *busy*, the state variables are updated as follows: *NB* = *NB* + 1, *BE* = min(*BE* + 1, *macMaxBE*) and *CW* = 2. If the number of backoff stages exceeds the maximum admissible value (i.e., *NB* > *macMaxCSMABackoffs*), the frame is dropped. Otherwise, the algorithm returns to Step 2.
   b. If the medium is *free*, then *CW* = *CW* - 1. If *CW* = 0 the frame is transmitted; otherwise the algorithm returns to Step 3 and performs a second CCA.

The 802.15.4 CSMA/CA algorithm includes an optional retransmission mechanism for reliability purposes. If this mechanism is enabled, the destination node must acknowledge each individual data frame upon its reception. If the acknowledgement is not received by the source node, within a pre-defined timeout, the frame is immediately re-transmitted. The maximum number of retransmissions per frame is specified by the *macMaxFrameRetries* parameter.

From the previous description it emerges that the CSMA/CA behavior is regulated by four parameters that are summarized in Table I, along with the range of values allowed by the standard. The impact of each single parameter on performance of the sensor network, has been analyzed in several previous papers [5][6][7][9][11]. Specifically, the authors of [11] have shown, through analysis and simulations, that in a star topology the packet delivery probability provided by the 802.15.4 CSMA/CA algorithm increases *monotonically* with the values of *macMinBE*, *macMaxCSMABackoffs*, and *macMaxFrameRetries*.

Table I. CSMA/CA Parameters and allowed values [1].

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

However, its increase becomes negligible after certain values of the above-mentioned parameters. They also show that increasing *macMinBE* is more energy efficient than increasing *macMaxCSMABackoffs* – in the sense that it increases the packet delivery probability with a lower power consumption – and increasing *macMaxCSMABackoffs* is more energy efficient than increasing *macMaxFrameRetries*. These results have been exploited in the design of the LEAP algorithm, presented in Section 4.

3. PROBLEM FORMULATION

We consider a star network formed by a PAN coordinator, acting

as the sink node, and a number of sensor nodes that acquire information from the external environment and periodically report acquired data to the sink node (i.e., at each *Beacon Interval*). We assume that the application has stringent requirements in terms of reliability and energy efficiency. To quantify these requirements we introduce the following indexes.

- *Packet delivery ratio* (R_D), defined as the ratio between the number of packets correctly received by the sink from a sensor node, and the total number of packets generated by that sensor node. It measures the *long-term reliability* experienced by a sensor node and is requested to be higher than a minimum value, R_D^{min}.
- *Miss ratio* (R_M), defined as the fraction of times the packet delivery ratio – calculated by a sensor node over the current Beacon Interval – drops *below* R_D^{min}. It measures the inability to achieve *short-term reliability* and should not exceed a given threshold R_M^{max}.
- *Average energy consumption per packet* (E_P), defined as the total energy consumed by a sensor node divided by the total number of packets generated by that node. It is a measure of *energy efficiency*.

Note that all considered indexes refer to a single sensor node, and not the entire sensor network. Let $R_D(\boldsymbol{par})$, $R_M(\boldsymbol{par})$, 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

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

The optimization problem expressed in (1) can be solved analytically, by deriving an analytical model of the sensor network and calculating the CSMA/CA parameter values that satisfy (1). This is basically the approach taken in [9] where, however, the authors consider packet delivery ratio and latency (instead of miss ratio). The main limitation of this approach is that its validity depends on the accuracy of the underlying analytical model. Since modeling typically requires making some simplifying assumption, an analytical approach may not always provide the optimal parameter values in a real scenario. In this paper we follow a different way to solve the problem expressed in (1). Specifically, we use a heuristic approach, based on a learning algorithm, to identify the optimal parameter setting adaptively.

4. LEAP ALGORITHM DESCRIPTION

In this section we describe the LEAP algorithm. We start with a high level description. Then, we will detail the different phases of the algorithm. Finally, we will describe some optimizations aimed at improving energy efficiency.

4.1 Basic Ideas

LEAP dynamically selects the optimal parameter setting, depending on the current operating conditions, in order to guarantee the reliability level required by the application with the minimum energy consumption. To this end, it also exploits the knowledge learned in the past history, whenever available.

At every Beacon Interval, each sensor node derives – through local measurements – the operating conditions of the network and stores the measured data in some local data structures. Initially, the sensor node has no knowledge about the past history and, hence, it enters an *Exploration* phase during which it uses an adaptive approach to set CSMA/CA parameters. Specifically, it starts with an initial set of parameter values and, then, increases or decreases the value of one parameter at a time, following an approach similar to that used in ADAPT [11]. Parameter values are increased when the reliability experienced by the sensor node, in terms of R_D and R_M, does not satisfy the application requirements, and decreased otherwise. After a certain number of steps, the adaptive algorithm stabilizes, in the sense that the sensor node oscillates between two (or few) adjacent parameter sets. At that point in time, the sensor node builds a *Learning Table* (see below for details), by using the knowledge learned during the *Exploration* phase, and transits to the *Exploitation* phase.

During the *Exploitation* phase the sensor node still performs local measurements to assess the experienced reliability and the current operating conditions, and stores the measured data in local data structures. In addition, it tries to exploit the knowledge learned during the previous phase to derive the optimal parameter setting. Specifically, at each *Beacon Interval*, the sensor node checks the learning table to derive the optimal parameter setting, given the current operating conditions. The following 3 outcomes can occur.

(a) The parameter setting suggested by the *Learning Table* coincides with the current one. No action is taken.
(b) The new setting suggested by the *Learning Table* differs from the current one. The sensor node assumes that the operating conditions have changed. Thus, it starts a new *Exploration* phase, by using the parameters suggested by the *Learning Table* as the initial settings.
(c) There is no entry in the *Learning Table* corresponding to the measured operating conditions. The sensor node assumes that the operating conditions have changed. As above, it starts a new *Exploration* phase.

In the next subsections we will describe in detail the actions carried out by the LEAP algorithm during the *Exploration* and *Exploitation* phases, and the data structure it uses.

4.2 Status Assessment and Data Structures

We denote by $\boldsymbol{s}(t) = [p_{busy}(t),\ \boldsymbol{par}(t)]$ the state of the sensor network, as perceived by a generic sensor node, at a given Beacon Interval t. Specifically, $\boldsymbol{par}$ is the set of CSMA/CA parameter values, while p_{busy} denotes the probability to find the channel busy during a channel access. The latter is a measure of congestion experienced by the sensor node and depends on a number of external factors, such as number of sensor nodes, offered load, etc. It also depends on the CSMA/CA parameter values used by the sensor node, i.e., the value of $\boldsymbol{par}$ currently used by the sensor node. The state $\boldsymbol{s}(t)$ can be easily derived by the sensor node as the parameter settings (i.e., $\boldsymbol{par}$) is known, and p_{busy} is measured locally as $p_{busy} = p_{busy}^1 + (1 - p_{busy}^1) \cdot p_{busy}^2$, where p_{busy}^1 (p_{busy}^2) is the probability to find the channel busy during the first (second) CCA operation. In practice, p_{busy}^1 and p_{busy}^2 are estimated by calculating the fraction of CCA operations that resulted in a busy channel in the last w Beacon Intervals. In addition to p_{busy}, at each Beacon Interval t the sensor node also estimates the delivery ratio $R_D(t)$ and the miss ratio $R_M(t)$. Along with the network state $\boldsymbol{s}(t)$, $R_D(t)$ and $R_M(t)$ contribute to determine the operating conditions experienced by the sensor node at Beacon Interval t. At each sensor node LEAP uses the following data structures to store information about the operating conditions experienced in the past.

- *Data cluster.* A table with an entry for each CSMA/CA parameter set used since the beginning of the last Exploration phase. The cluster is cleared whenever a new Exploration phase starts. Each entry in the cluster has the format $\langle \boldsymbol{par}, R_D, R_M, count \rangle$,

where *count* indicates the number of times the corresponding parameter set has been used so far.

- *State List.* This list contains all the states experienced by the sensor node since the beginning of the current Exploration phase. Every time a new state is encountered an element is added to the list. The State List is used to build the Learning Table, at the end of the Exploration phase.
- *Learning Table.* This table is created at the end of the first Exploration phase, and updated after each Exploration phase, on the basis of the State List. The Learning Table includes an entry for each operating condition experienced during the past history, and the corresponding optimal parameter settings, according to the learned knowledge. Each entry has the following format

$$\langle \boldsymbol{par}, elem_1, elem_2, \dots elem_i, \dots \rangle$$

where $elem_i = \langle [p^i_{busy-min}, p^i_{busy-max}],\ \boldsymbol{new\_set} \rangle$, for any i. Table II shows an example of Learning Table. Let us assume that $\boldsymbol{par4}$ is the current parameter set and $\overline{p_{busy}}$ is the p_{busy} value measured in the current *Beacon Interval*. Based on the past history, the *Learning Table* suggests to use, as the new set, $\boldsymbol{par8}$ or $\boldsymbol{par7}$, depending on whether $\overline{p_{busy}}$ is in the range [0.25, 0.37], or in the range [0.77, 0.84], respectively. We will see later how to manage cases for which there is no entry in the *Learning Table.*

Table II. Example of Learning Table.

Current Set	Elem1		Elem2		Elem3	
	p_{busy}	New Set	p_{busy}	New Set	p_{busy}	New Set
par1	[0.30, 0.33]	par2	[0.64, 0.70]	par3	[0.80, 0.88]	par6
par2	[0.43, 0.51]	par3				
par3	[0.70, 0.75]	par5				
par4	[0.25, 0.37]	par8	[0.77, 0.84]	par7		

Table III. 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	$MaxBE_{MAX}$	$MinBE_{MIN+2}$	$MaxCSMABackoffs_{MIN}$	$MaxFrameRetries_{MIN}$
...	…	…	…	…
...	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MIN}$	$MaxFrameRectries_{MIN}$
...	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MIN+1}$	$MaxFrameRetries_{MIN}$
...	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MIN+2}$	$MaxFrameRetries_{MIN}$
...	…	…	…	…
...	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MAX}$	$MaxFrameRetries_{MIN}$
...	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MAX}$	$MaxFrameRetries_{MIN+1}$
...	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MAX}$	$MaxFrameRetries_{MIN+2}$
...	…	…	…	…
I_{MAX}	$MaxBE_{MAX}$	$MinBE_{MAX}$	$MaxCSMABackoffs_{MAX}$	$MaxFrameRetries_{MAX}$

4.3 Exploration Phase and Adaptive Tuning

Initially, the algorithm has no information about the past history. Hence, it starts with an *Exploration* phase, during which it uses a simple *Adaptive Tuning* – inspired from the ADAPT algorithm proposed in [11] – to dynamically adjust CSMA/CA parameters. Adaptive Tuning increases or decreases one parameter at a time, depending on the experienced reliability. Specifically, at each Beacon Interval, the sensor node measures the delivery ratio R_D and the miss ratio R_M. If at least one of them is below the required threshold, the value of one parameter is increased, giving higher priority in the modifications of MACMINBE over 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 reached their maximum value, MACMAXFRAMERETRIES is progressively increased. If both R_D and R_M satisfy the application requirements, then the set of parameter values is tentatively reduced. The strategy for decreasing parameter is just the opposite. First, MACMAXFRAMERETRIES is reduced until it reaches its minimum value. Then, the same procedure is applied at first to MACMAXCSMABACKOFFS and MACMINBE, in the order. Without loss in generality, we can assume that CSMA/CA parameter sets are ordered as shown in Table III. Under this assumption each parameter set can be identified by the corresponding set-index, and the adaptive tuning algorithm always moves from one set to another adjacent set. During the Exploration phase the sensor node also accumulates information in its data structure to keep track of the experienced conditions.

At each Beacon Interval the sensor node adds an entry in the *Data Cluster* to store the delivery ratio $\widetilde{R_D}$, and the miss ratio $\widetilde{R_M}$ measured in the present Beacon interval, with the current parameter set (identified by the corresponding set index). The *count* field is set to 1. If an entry already exists for that parameter set, then the fields containing statistics about R_D and R_M are updated as follows

$$R_D = \frac{\widetilde{R_D} + R_D \cdot count}{count + 1} \qquad R_M = \frac{\widetilde{R_M} + R_M \cdot count}{count + 1}$$

and the *count* field is increased, to track the number of times the set has been used so far. In this way, estimates of R_D and R_M become more and more accurate. In addition to update the Data Cluster, during the Exploration phase the State List is also built. At each Beacon Interval, after estimating p_{busy}, a new state is added to the list (if not present). We recall here that the state, at Beacon Interval t, is given by $\boldsymbol{s}(t) = [p_{busy}(t),\ \boldsymbol{par}(t)]$. Due to its behavior, after a short transient time, the adaptive tuning algorithm tends to oscillate between two (or few) adjacent parameter sets. We assume that the Exploration phase ends when the value of *count*, for one of the sets in the Data Cluster, reaches a predefined threshold $count^{min}$. Afterwards, the corresponding parameter set is assumed to be the most appropriate for the current operating conditions, i.e., the "optimal" set according to the Adaptive Tuning algorithm. Throughout, we will refer to this set as $\boldsymbol{par}_{opt}$. Finally, information collected so far in the State List is stored in the Learning Table. Specifically, the following steps are performed. For each parameter set used in the Exploration phase, the corresponding p_{busy} values are extracted from the State List. Let $P_j = \{p_1^j, p_2^j, p_3^j, \dots, p_K^j\}$ be the sequence of values measured when using set j (in Table III). Then, the average value μ_j and standard deviation σ_j for values in P_j are calculated, i.e.,

$$\mu_j = \frac{1}{K}\sum_{i=1}^{K} p_i^j \qquad \sigma_j = \sqrt{\frac{\sum_{i=1}^{K}\left(p_i^j - \mu_j\right)^2}{(k-1)}}$$

Finally, a new element is added in the Learning Table, in the entry corresponding to set-index *j* (if there is no entry for this set, a new entry is created). The new added element is

$$\langle [\mu_j - 2\sigma_j,\ \mu_j + 2\sigma_j], \boldsymbol{par}_{opt} \rangle$$

This concludes the Exploration phase. Then, the sensor nodes enter the Exploitation phase.

4.4 Exploitation Phase

During this phase the sensor node uses the Learning Table to decide the most appropriate parameter setting, depending on the operating conditions. At each Beacon Interval the sensor node performs the following steps.

1. *Operating Conditions Assessment.* The sensor node measures the channel-busy probability p_{busy} and derives the new state. In addition, it estimates the delivery ratio $\widetilde{R_D}$ and the miss ratio $\widetilde{R_M}$.

2. *Data Cluster Update*. The sensor node updates statistics in the Data Cluster, on the basis of the $\widetilde{R_D}$ and $\widetilde{R_M}$ estimates obtained in the previous step.
3. *Parameter Setting*. The sensor node checks for an entry in the Learning Table corresponding to the current parameter set. The following three outcomes can occur.
   a. The *new set* suggested by the Learning Table is equal to the *current set* or it is *adjacent* to it (i.e. $new\ set = current\ set \pm 1$). This means that the operating conditions have not changed significantly. Hence, no action is taken by the sensor node.
   b. The *new set* suggested by the Learning Table is different from the *current set* and it is not an *adjacent set* (i.e. $new\ set \neq current\ set \pm 1$). This means that the operating conditions have changed, but the new conditions have been already experienced. The sensor node changes the parameter settings according to the Learning Table, deletes the current Data Cluster, and starts a new Exploration phase.
   c. No element in the selected entry of the Learning Table matches the measured value of p_{busy}. This means that the network conditions have changed, but the new operating conditions have never been experienced in the past. The sensor node deletes the current Data Cluster and starts a new Exploration phase.

4.5 Controlled Tuning Algorithm

In this section we describe some specific actions performed by LEAP to optimize its behavior and further improve its energy efficiency. The algorithms used during the Exploration or Exploitation phases aim at providing the reliability level required by the application with the minimum energy consumption. However, since CSMA/CA parameters assume discrete values, typically the delivery ratio (miss ratio) experienced by a sensor node is significantly above (below) R_D^{min} (R_M^{max}), thus consuming more than necessary. On the other hand, using a lower set would not guarantee at least one of the two reliability indexes. To optimize energy consumption, while still meeting reliability constraints, LEAP uses a *Controlled Tuning* algorithm that finely adjusts the parameter settings on the sensor node, by switching between two adjacent sets in a controlled way. The goal is to have a reliability level just above the application threshold, so as to minimize energy consumption. The *Controlled Tuning* algorithm is used both during the Exploration phase and the Exploitation phase. It is executed after the new parameter set j' has been derived, either through the Adaptive Tuning algorithm (Exploration Phase) or the Learning Table (Exploitation Phase), and returns the adjusted parameter set *j* that will be actually used in the next Beacon Interval. The *Controlled Tuning* algorithm is described in Algorithm 1. Let $R_D(i)$ and $R_M(i)$ denote the delivery ratio and miss ratio statistics contained in the Data Cluster for set *i*, (for $i = j'-1,\ j',\ j'+1$), provided that an entry for set i is available in the Data Cluster. The Controlled Tuning algorithm checks whether the reliability constraints are satisfied (line 4). If *both* reliability indexes are satisfied using set j' and statistics are available in the Data Cluster for set $j'-1$, then the algorithm performs a fine tuning between sets j' and $j'-1$ (lines 6-10). Specifically, it computes the distance of $R_D(j')$ and $R_M(j')$ from the corresponding threshold (i.e., R_D^{min} and R_M^{max}, respectively) and considers the index closer to the threshold. Then, the new parameter set *j*, 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).

Instead, if *at least one* of the two reliability indexes does not meet the application requirements, then the algorithm performs a fine tuning between sets j' and $j'+1$ (lines 12-17). If no entry is available for set $j'+1$ in the Data Cluster the algorithms returns $j'+1$ as the set to use in the next Beacon Interval (line 12). Otherwise, as above, the distance between $R_D(j')$ and $R_M(j')$ and the corresponding thresholds is calculated. However, now 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      R_D(i) = Cluster[i].R_D;   R_M(i) = Cluster[i].R_M;
3   end for
4   if ((R_D(j') ≥ R_D^min) and (R_M(j') ≤ R_M^max))
5      if (Cluster[j' − 1] = NULL) return j' − 1;
6      else
7         p_D = (R_D(j') − R_D^min)/(R_D(j') − R_D(j' − 1));
8         p_M = (R_M^max − R_M(j'))/(R_M(j' − 1) − R_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 = (R_D^min − R_D(j'))/(R_D(j' + 1) − R_D(j'));
15        p_M = (R_M(j') − R_M^max)/(R_M(j') − R_M(j' + 1));
16        p = max(p_D, p_M);
17        return j' (j' + 1) with probability: (1 − p)  p
```

5. SIMULATION SETUP

To evaluate the performance of LEAP we used the ns2 simulation tool [13] that includes the 802.15.4 module. We considered a star network scenario where sensor nodes are placed in a circle centered at the sink node (also acting as PAN coordinator), 10 m far from it. The transmission range was set to 15 m, while the carrier sensing range was set to 30 m (according to the model in [14]).

In our analysis, in addition to the reliability and energy efficiency indexes already introduced in Section 3, we also considered the two following performance indexes:

- *Average latency*, defined as the average time since the packet transmission starts at the source node to when the packet is correctly received by the sink. It characterizes *timeliness*.
- *Transient time*, defined as the time instant, after a change in the operating conditions, when the delivery probability – calculated over the current Beacon Interval – reaches the steady-state value for the new operating conditions (with a tolerance of 3%). It measures the *ability to adapt* to changing conditions.

The energy consumed by a sensor node was calculated according to the model in [15] , which is based on the Chipcon CC2420 radio transceiver [16]. This model assumes four radio states, namely *transmit*, *receive*, *idle* and *sleep*. In addition, it also accounts for the energy spent during the transition from a state to another. In our analysis, during the backoff waits, a sensor node is supposed to enter the idle state.

In our simulations we assumed $R_M^{max} = 0.15$ and $R_D^{min} = 0.8$, i.e., the application requires a packet delivery ratio $R_D \geq 80\%$ and a miss ratio $R_M \leq 15\%$, for any sensor node. We need to point out that these thresholds are somewhat arbitrary as they strongly depend on the specific application. However, we performed other experiments with different thresholds and the obtained results are in line with those presented below.

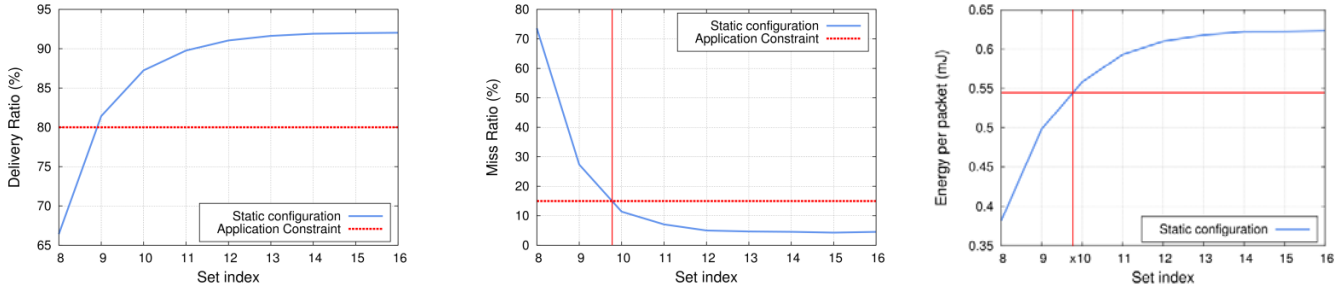

Figure 1. Delivery Ratio (left), Miss Ratio (center) and Energy (right) associated to some sets of MAC parameters

5.1 Algorithms for Comparison

We compared the performance of LEAP with that of the following three algorithms characterized by different approaches in deriving the optimal CSMA/CA setting.

- *Model-based offline computation* [9]. This algorithm derives the optimal parameter setting offline, by solving an analytical model of the sensor network, based on a Discrete Time Markov Chain (DTMC). Due to the high computational costs, the algorithm is executed on the sink node, and optimal parameter settings are then communicated to sensor nodes.
- *Model-based online adaptation* [10]. This is an adaptive algorithm still based on a DTMC model of the WSN. The considered analytical model is a simplified version of the one used by the previous algorithm and, thus, it can be executed at the sensor nodes. Specifically, sensor nodes derive some congestion indexes through online measurements (instead of deriving them analytically from the model) and adapt the parameter values to time-varying operating conditions.
- *ADaptive Access Parameter Tuning (ADAPT)* [11]. ADAPT is a heuristic *measurement-based adaptive* algorithm, similar to the Adaptive Tuning algorithm used by LEAP during the Exploration phase. Specifically, ADAPT dynamically increases and decreases the value of CSMA/CA parameters, one at a time, in such a way that the measured delivery ratio remains confined within a region defined by two thresholds, i.e. R_D^{low} and R_D^{high} above the minimum value required by the application, i.e., R_D^{min} (in our simulation experiments we used 0.86 and 0.90 for R_D^{low} and R_D^{high} respectively).

For comparison, we also considered an *Ideal* (but unfeasible) algorithm that assumes a complete knowledge of the network conditions. To derive the optimal settings with the Ideal algorithm, in a given scenario, we performed simulation experiments with all possible CSMA/CA parameter values. Then, we selected the optimal parameter set as the one that satisfies (1), i.e., guarantees $R_D \geq R_D^{min}$ and $R_M \leq R_M^{max}$ with minimum energy consumption. Since CSMA/CA parameters only take discrete values, to compute the optimal parameter set, and the corresponding reliability, performance and energy indexes, the Ideal algorithm uses an approach similar to the Controlled Tuning algorithm. For instance, Figure 1 shows the delivery ratio, miss ratio and energy consumption for different CSMA parameter sets. We can see that, in order to satisfy both reliability constraints, we need to use the parameter set with index 10. However, this provides a miss ratio lower than 15% and a delivery ratio significantly higher than 80%. Ideally, we can assume that curves in Figure 1 are continuous and, thus, we can derive the ideal CSMA/CA parameter set that exactly matches the most stringent reliability constraint (i.e., miss ratio in our case). Figure 1 shows that this ideal set would have a (virtual) index equal to 9.8. Correspondingly, we can calculate the value of delivery ratio, miss ratio and energy per packet.

5.2 Parameter Values and Methodology

Table IV summarizes the operating parameters used in our simulation experiments. We also assumed that each sensor node generates 10 data packets at every Beacon Interval. All these packets are together sent to the MAC layer – in order to be transmitted – at the beginning of the Contention Access Period.

Since the model-based algorithms considered for comparison assume ideal channel conditions, in our analysis we considered an error-free communication channel. However, LEAP is able to work also in the presence of corrupted and/or missed packets. In our experiments, for each simulated scenario, we performed 10 independent experiments, where each experiment consists of 1000 Beacon Periods. For each experiment we discarded the initial transient interval (10% of the overall duration) during which nodes associate to the coordinator node and start generating packets. The results shown in Section VI are averaged over all the different experiments. We also derived confidence intervals by using the independent replication method. They are typically so small that they cannot be appreciated in the figures below.

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)	11, 8
$MinBE^{min}$, ^{max}	1, 7
$MaxCSMABackoffs^{min}$, ^{max}	1, 10
$MaxFrameRetries^{min}$, ^{max}	0, 9
Power Consumption in RX, TX, Idle, Sleep mode	35.46 mW, 31.32 mW, 0.77 mW, 0.036 µW
Window size (w)	2
$count^{min}$	10

6. SIMULATION RESULTS

Our analysis is divided in two parts. In the first one, we analyze the performance of the considered algorithms in stationary conditions, i.e., we assume that the number of sensor nodes is fixed for the whole duration of the experiment. In the latter case, we consider network topologies where the number of sensor nodes changes over time. The presented results refer to a specific sensor node. However, we observed similar results for all the nodes in the network.

6.1 Analysis in stationary conditions

Figure 2 shows the delivery ratio and miss ratio experienced by the considered node for an increasing number of sensor nodes. We

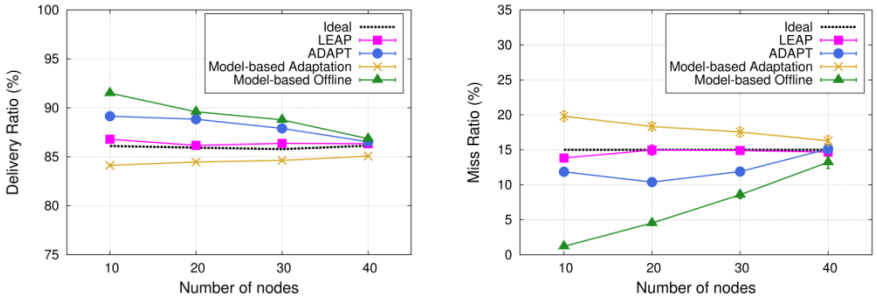

Figure 2. Packet delivery ratio (left) and miss ratio (right) for an increasing number of sensor nodes.

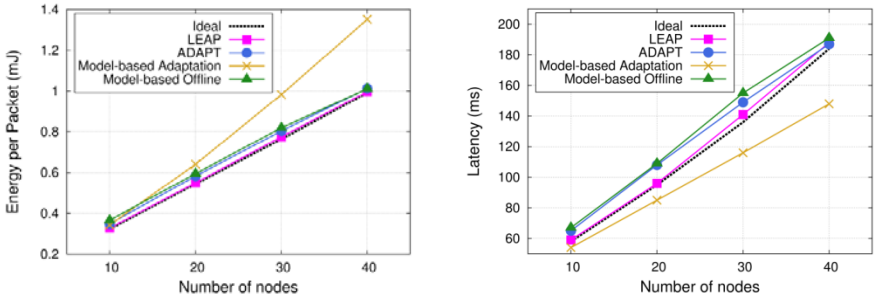

Figure 3. Average per packet energy consumption (left), and average latency (right) for an increasing number of sensor nodes.

can see that all the four algorithms are able to meet the application requirements in terms of delivery ratio. The delivery ratio provided by the model-based adaptation algorithm is very close to the minimum threshold required by the application (i.e., 80%).
However, this produces frequent drops below the threshold, which are reflected in a high miss ratio, higher than the maximum value tolerated by the application. The model-based offline algorithm provides the highest delivery ratio and the lowest miss ratio. However, this results in an energy consumption higher than necessary (see below). The LEAP performance is very close to that of the Ideal algorithm. In terms of average energy consumption per packet (Figure 3-left) the Ideal algorithm is the best solution, as expected. However, LEAP is very close to it (the two curves basically overlap). Due to its Controlled Tuning algorithm, LEAP is able to minimize the energy consumption. Also ADAPT and the model-based offline algorithm exhibits good performance in terms of energy per packet, while the energy consumption of the model-based adaptation algorithm is significantly higher. This is mainly because the latter algorithm tends to increase the communication reliability by increasing the number of allowed retransmissions (*macMaxFrameRetries)*, while other algorithms achieve the same result by increasing the number of backoff trials (*macMaxCSMA-Backoffs*), which is more energy efficient, as highlighted in [11]. Finally, Figure 3-right shows the average latency experienced by packets for the different algorithms. Now, the model-based adaptive algorithm has the best performance. The reason is the same as above, i.e., this algorithm tends to increase the number of retransmissions, while the others give priority to the number of backoff trials. When a packet is re-transmitted the contention window size is re-initialized to its minimum value. Instead, when a new backoff trial is started, the contention window size is doubled (unless it has reached its maximum value). Anyhow, the low latency introduced by this algorithm is paid in terms of higher energy consumption, as discussed above. Again, the performance of LEAP is very close to that of the Ideal algorithm, while the model-based offline algorithm and ADAPT introduce a slightly higher latency.

6.2 Analysis in dynamic conditions

Let us now turn our attention to dynamic scenarios where operating conditions vary over time. For the sake of space, we limit our analysis to scenarios where only the number of sensor nodes changes dynamically, i.e., some sensor nodes activate and deactivate at certain time instants. Specifically, in our experiments we assume that there are 10 sensor nodes always active, while 50 more sensor nodes activate and deactivate periodically, at every 100 Beacon Intervals. Our goal is to investigate how the different algorithms react to such changes. We will limit our analysis to adaptive algorithms, as the model-based offline algorithm and the Ideal algorithm are not well suited for dynamic scenarios.
Figure 4 shows the transient time introduced by the considered algorithms to adapt to the new operating conditions. We analyzed separately the transient originated by an increase and a decrease in the number of sensor nodes (left and right side, respectively, in Figure 4). As a general comment, the transient times experienced by LEAP, in subsequent activation/deactivations of sensor nodes, tend to become shorter and shorter, while they remain approximately constant for the other two algorithms. This is due to the learning mechanism used by LEAP. When the sensor network passes through the same operating conditions already experienced in the past, after a certain number of times, LEAP is able to find out the optimal settings very quickly, as it can exploit the information available in the Learning Table. In the case of decrease in the number of sensor nodes, LEAP is significantly faster than the other algorithms. The model-based adaptive algorithm converges in no more than 5 Beacon Intervals as the optimal setting is derived by using the samples measured in the previous m Beacon Intervals (and we used m = 4 in our experiments). However, this algorithm requires to know in advance the number of (active) sensor nodes in the network. This is, generally, difficult to predict. It may become a serious issue in dynamic scenarios, like the ones considered in our analysis, where the number of sensor nodes changes with time. In our experiments, we ran the algorithm with the maximum number of sensor nodes (i.e., 60). This means that,

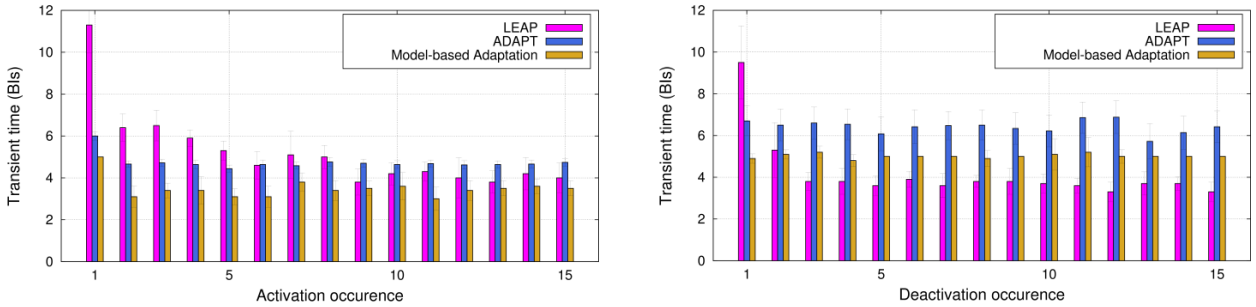

Figure 4. Transient time when the number of active sensor nodes increases (left) and decreases (right).

when there are only 10 nodes, the parameter settings are not optimal and sensor nodes consumes more energy than necessary. On the other hand, running the algorithm with the minimum number of sensor nodes (i.e., 10) has negative drawbacks as well. When the number of active nodes is higher, the algorithm may not satisfy the reliability constraints required by the applications.

LEAP and ADAPT do not suffer from such limitations, as they do not require any input parameter (beyond the reliability requirements). 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 guarantee the same reliability.

Finally, let us focus on the average energy per packet consumed by the three algorithms in the considered scenario. Thanks to its learning mechanism, LEAP introduces the minimum energy consumption (1.37 mJ) with respect to ADAPT (1.51 mJ) and the model-based approach (2.23 mJ).

7. CONCLUSIONS

In this paper we have proposed a new, learning-based algorithm for deriving the optimal CSMA/CA parameter setting in IEEE 802.15.4 sensor networks, that guarantees the reliability constraints required by the application with the minimum energy consumption. The proposed algorithm adapts the CSMA/CA parameters, on the basis of the reliability experienced by the sensor node (measured locally). However, unlike previous similar adaptive algorithms, it also exploits a learning mechanism to speed up the transient time when the new operating conditions have been already experienced in the past and, thus, information about the optimal settings is known. We have analyzed our algorithm in both stationary and dynamic scenarios. Our simulation results have shown that LEAP outperforms all previous similar algorithms and behaves very closely to an ideal, but unfeasible, algorithm for optimal CSMA/CA parameter setting.

8. REFERENCES

[1] 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 (LR-WPANs), IEEE Computer Society, 2006.

[2] ZigBee Alliance, The ZigBee Specification version 1.0 (Q4/2007).

[3] R. Zurawski, "Networked Embedded Systems: An Overview" Chapter 1 in Networked Embedded Systems (R. Zurawski, Editor), pp. 1.11-1.16, CRC Press, 2009.

[4] K. Yedavalli and B. Krishnamachari, "Enhancement of the IEEE 802.15.4 MAC protocol for scalable data collection in dense sensor networks", in *Proc. of the 6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops (WiOPT 2008)*, April 2008, pp. 152–161.

[5] C. K. Singh, A. Kumar, and P. M. Ameer, "Performance evaluation of an IEEE 802.15.4 sensor network with a star topology", *Wireless Networks*, vol. 14, no. 4, 2008.

[6] S. Pollin, M. Ergen, S. Ergen, B. Bougard, L. Der Perre, I. Moerman, A. Bahai, P. Varaiya, and F. Catthoor, "Performance analysis of slotted carrier sense IEEE 802.15.4 medium access layer", *IEEE Trans. on Wireless Communications*, vol. 7, no. 9, pp. 3359–3371, Sep. 2008.

[7] G. Anastasi, M. Conti, and M. Di Francesco, "A comprehensive analysis of the MAC unreliability problem in IEEE 802.15.4 wireless sensor networks", *IEEE Transactions on Industrial Informatics*, vol. 7, no. 1, 2011.

[8] W. Ye, J. Heidemann, and D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 3, pp. 493–506, Jun. 2004.

[9] P. Park, P. Di Marco, P. Soldati, C. Fischione, and K. Johansson, "A generalized Markov chain model for effective analysis of slotted IEEE 802.15.4," in *Proc. of the 6th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS '09)*, October 2009, pp. 130–139.

[10] P. Park, P. Di Marco, C. Fischione, K. Johansson, "Modeling and Optimization of the IEEE 802.15.4 Protocol for Reliable and Timely Communications", *IEEE Transactions on Parallel and Distributed Systems*, 2012.

[11] M. Di Francesco, G. Anastasi, M. Conti, S. Das, V. Neri, Reliability and Energy Efficiency in IEEE 802.15.4/ZigBee Sensor Networks: A Cross-layer Approach", *IEEE Journal on Selected Areas in Communications*, Vol. 29, N. 8, 2011.

[12] S. Brienza, D. De Guglielmo, G. Anastasi, M. Conti, V. Neri, "Strategies for Optimal MAC Parameter Setting in IEEE 802.15.4 Wireless Sensor Networks: a Performance Comparison", Proc. *IEEE Symposium on Computers and Communications (ISCC 2013)*, Split, Croatia, July 2013.

[13] Network Simulator Ns2, http://www.isu.edu/nsnam/ns.

[14] G. Anastasi, E. Borgia, M. Conti, E. Gregori, and A. Passarella, "Understanding the Real Behavior of 802.11 and Mote Ad hoc Networks", *Pervasive and Mobile Computing*, Vol. 1, N. 2, 2005.

[15] B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan, W. Dehaene, "Energy Efficiency of the IEEE 802.15.4 Standard in Dense Wireless Microsensor Networks: Modeling and Improvement Perspectives", *Proc. Conference on Design, Automation and Test in Europe (DATE)*, Volume 1, 2005.

[16] CC2420 Website, http://www.ti.com/product/cc242