Smarter Radios for Energy efficiency in Wireless Sensor Networks

Maria Isabel Vergara Gallego

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministérial : 7 août 2006

Présentée par

María Isabel VERGARA GALLEGO

Thèse dirigée par **Frédéric Pétrot**
et codirigée par **Franck Rousseau**

préparée au sein de **Laboratoire d'Informatique de Grenoble (LIG), Laboratoire de Techniques de l'Informatique et de la Microélectronique pour l'Architecture de systèmes intégrés (TIMA)**
et de **École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (EDMSTII)**

Smarter Radios for Energy Efficiency in Wireless Sensor Networks

Thèse soutenue publiquement le **3 Octobre 2013**,
devant le jury composé de :

M. Tanguy Risset
Professeur, INSA de Lyon, Rapporteur
M. Christian Bonnet
Professeur, Eurecom, Rapporteur
M. Olivier Sentieys
Professeur, IRISA, Examinateur
M. Noel De Palma
Professeur, Université Joseph Fourier, Examinateur
M. Frédéric Pétrot
Grenoble INP, Directeur de thèse
M. Franck Rousseau
Grenoble INP, Co-Directeur de thèse

Acknowledgments

First of all, I would like to thank my supervisors Professor Frédéric Pétrot and Dr. Franck Rousseau for their guidance. I want to express my appreciation and admiration; their support and contributions have been invaluable during my research.

I also offer my gratitude to the members of the jury M. Tanguy Risse, M. Christian Bonnet, M. Olivier Sentieys and M. Noel De Palma, for accepting to evaluate my thesis and for all the suggestions that helped me to improve my work and to think about possible future directions of my research.

I also would like to express my gratitude to all members of the Drakkar team for the contributions and the discussions, and for offering me their friendship. I really enjoyed the company of the team. Besides, I want to thank the administrative and technical staff of LIG and Joseph Fourier University for their help and kindness.

To all my friends: Ana, German, Leidy, Alessio, Maru, Nazim, Carina, Reinaldo, Bogdan, Maciej, Sofia, Gabriele, Michal, Malisha Elodie, Jun-Young, Osama, Julien, Cecilia, Luiz, Pierre and Ana, I want to thank you all for offering me your company, a friendly hand and unforgettable moments.

I specially thank my parents, Isidro and Alicia, and my brother, Hernan, for their support and patience during my studies.

Finally I want to thank Carlos for giving me the strength to finish this work and for being at my side, along this year.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Instruction Processor
BER	Bit Error Rate
BO	Beacon Order
BI	Beacon Interval
CCA	Clear Channel Assessment
CDMA	Code Division Multiple Access
CRC	Cyclic Redundancy Check
CSMA/CA	Channel Sensing Medium Access/Collision Avoidance
FDMA	Frequency Division Multiple Access
FDR	Frame Delivery Rate
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
IIR	Infinite Impulse Response
ISA	Instruction Specific Architecture
LIFS	Long Inter Frame Space
LFSR	Linear Feedback Shift Register
LPL	Low Power Listening
LPP	Low Power Probe
LQI	Link Quality Indicator
MAC	Medium Access Control
MPDU	MAC Protocol Data Unit
PER	Packet Error Rate
PRNG	Pseudo-Random Number Generator
RSSI	Received Signal Strength Indicator
SD	Superframe Duration
SDR	Software Defined Radio
SI	Superframe Order
SIFS	Short Inter Frame Space
SoI	Sleep on Idle
TDMA	Time Division Multiple Access
WoI	Wake on Idle
WoR	Wake on Radio
WSN	Wireless Sensor Network

Des radios plus intelligentes pour améliorer l'efficacité énergétique dans les réseaux de capteurs

Résumé:

Les contraintes présentes dans les réseaux de capteurs impliquent l'introduction de techniques d'optimisation à différents niveaux de conception : du matériel au logiciel et dans la pile de communication. En effet, le déploiement des réseaux de capteurs, à faible consommation énergétique, exige une conception conjointe du matériel et du logiciel adaptée à l'application visée. Étant donné la nature évènementielle et multitâche des applications dans les réseaux de capteurs, nous pourrions penser à rajouter différentes unités de traitement qui coopèrent pour gérer les évènements et les tâches de manière optimale. Ainsi, la complexité des tâches accomplies par le processeur principal peut être réduite, ce qui contribue à atteindre l'efficacité énergétique.

Dans cette thèse nous étudions un ensemble de protocoles qui facilitent l'implémentation des *smart radios*. L'idée principale des *smart radios* est l'introduction de l'intelligence dans la puce radio de manière à ce qu'elle soit capable de prendre des décisions ainsi que d'exécuter plusieurs tâches de manière autonome et sans l'intervention du processeur principal. Cette dernière sera responsable du *bootstrap* du réseau et, après qu'un état stable est atteint, le processeur peut rester inactif la plupart du temps, alors que la puce radio continue à fournir un ensemble de services.

Le protocole proposé est appelé *Wake on Idle* et il fournit la maintenance de voisinage intégrée avec une méthode d'accès au canal. Ces services sont basés sur des transmissions analogiques qui sont codées dans le temps. De cette manière, dès que le réseau entre dans l'état stable (c.à.d. la topologie est formée et les noeuds sont associés et synchronisés), le traitement numérique de trames n'est pas nécessaire. Puisque *Wake on Idle* est basé sur des informations de bas niveau, il peut être facilement intégré dans la puce radio et fonctionner comme un coprocesseur qui fournit des services de haut niveau au processeur principal, comme la maintenance du voisinage et l'accès au canal. Grâce à une analyse théorique et une implémentation préliminaire, nous démontrons la faisabilité du protocole et nous montrons plusieurs caractéristiques intéressantes qui aident à atteindre l'efficacité énergétique et de bonnes performances.

Ensuite, nous exploitons la signalisation analogique afin d'optimiser le *duty-cycle* des protocoles d'accès au canal existants. Nous proposons également un mécanisme appelé *Sleep on Idle* qui est basé sur l'échange de signaux analogiques ou *busy tones*. *Sleep on Idle* peut être intégré dans la radio et il peut décider quand le processeur doit être réveillé. Enfin, nous avons intégré le mécanisme de notification dans le standard IEEE 802.15.4 et nous avons évalué ce mécanisme par des simulations et expérimentations. Les résultats montrent un gain important en termes de consommation en énergie et de réactivité du réseau.

Les mots clefs: Radio Intelligente, co-processeur, Maintenance de voisinage, *duty-cycling*, Accès au canal, Efficacité énergétique

Smarter Radios for Energy Efficiency in Wireless Sensor Networks

Abstract:

The constraints of Wireless Sensor Networks scenarios require the introduction of optimization techniques at different design levels: from the hardware to the software and communication protocol stack. In fact, the design of energy efficient WSNs involves an appropriate hardware/software co-design oriented to the concerned application. Given the event driven and multitasking nature of WSNs applications, one could think of adding different processing units that cooperate to manage events and tasks in an optimal way. Then, the complexity of tasks performed by the main processing unit can be reduced and energy efficiency can be achieved.

In this PhD thesis we study protocols that leverage the implementation of *smart radios*. The idea of smart radios is introducing intelligence into the radio chip; in this way, it will be able to take decisions and perform several tasks in an autonomous way and without any intervention of the main CPU. The CPU will be charged of bootstrapping the network and, after a stable state is reached, it can remain inactive most of the time while the radio chip provides a given set of services.

The proposed protocol is called *Wake on Idle* and it provides integrated neighborhood maintenance and low duty-cycle medium access control. These services are provided based on analog transmissions that are *time encoded*; then, as soon as the network enters the stable state (*i.e.* the topology is formed and nodes are associated and synchronized) digital processing of frames is not needed. Since it relies on low-level information, Wake on Idle can be easily implemented on hardware and integrated into the radio chip; then, it works as a coprocessor that provides high-level services (*i.e.* neighborhood maintenance and medium access) to the main processing unit. Through theoretical analysis and a preliminary implementation we demonstrate the feasibility of the protocol and we show several interesting characteristics that help achieving energy efficiency and good performance.

Then, we further exploit analog signaling to optimize duty cycle of existing medium access control protocols. We propose a mechanism called *Sleep on Idle* and it is based on the exchange of analog busy tones. Sleep on Idle can also be integrated into the smart radio to take decisions about whether the CPU has to be woken up. We apply the decision mechanism to the slotted IEEE 802.15.4 standard and validate it through simulations and experimentations. The results show an important gain in terms of energy consumption and network reactivity.

Keywords: Smart Radios, co-processing, Neighborhood Maintenance, duty-cycling, Medium Access Control, Energy Efficiency

Chapter 1

Introduction

1.1 Motivation

Emerging technologies and advances in low-power electronics design have caused the research community to move from wired solutions to the wireless domain. In fact, an increasing interest in Wireless Sensor Networks (WSNs) has been observed in the last decade. The flexibility of Wireless Sensor Networks, compared to the classic wired networks, makes them attractive for a wide range of domains, such as environment monitoring, tracking and localization, medicine and health, control and domotics.

However, contrary to their wired counterpart, Wireless Sensor Networks may suffer from several problems. The wireless communication channel has an unreliable nature, hence, effective information exchange and channel capacity are significantly reduced since:

- Transmissions may be affected by the medium *i.e.* attenuation due to propagation and obstacles, noise, multipath propagation. Besides, the communication channel may not be symmetric.
- All nodes share the same communication channel; hence, there is a high probability that transmissions happen simultaneously and data is corrupted.
- Hardware characteristics and environmental conditions such as temperature and humidity may have an effect in the communication quality.

Thus, in these scenarios, the quality of communication is greatly influenced by environmental, hardware and other external factors. One or several mechanisms should be added to cope with these issues, and they must be chosen according to the application, the available resources and the environmental conditions. Different protocols and mechanisms that guarantee the proper delivery of information are implemented at different levels of the communication protocol stack; the huge number of issues that these protocols cope with increases the complexity of the tasks executed by each node in the network.

Additionally, most of the WSNs applications require the nodes (or sensing units) to run unattended during long periods of time *i.e.* months or even years, in areas where no external source of energy is available and they should be small and cheap. Thus, nodes should perform sensing, processing and communication tasks in an optimal way, relying on a small battery and highly constrained computation resources. Moreover, techniques proposed to optimize energy consumption may increase the complexity of the layered protocol stack. Consequently, when deploying a WSN, the designer should face a set of contradictory constraints. There is a huge set of parameters that generates

a huge design space; and the proper compromise between this set of constraints and requirements must be found.

Current hardware used for a WSN node consists of one processing unit, *i.e.* a low-power microcontroller, connected to a set of sensors and/or actuators, a radio interface used for data communication, a storage unit *i.e.* an external flash memory, and additional peripherals such as push buttons, debugging interfaces, displays and LEDs. The processing unit is in charge of controlling every single task, from sensors data retrieval, to neighbor sensing and correct packet transmission/forwarding. Hence, while it executes the whole communication protocol stack *i.e.* radio chip control, medium access, routing and topology control, it must also execute the application, respond to events and take decisions.

Thus, the processing unit must execute several concurrent tasks, some of them with very tight latency requirements. In addition, this must be done with highly constrained resources *i.e.* reduced energy budget and computation power. To reach these requirements, researchers have proposed very light operating systems such as TinyOS [1] and Contiki [2]. These operating systems provide concurrency through the introduction of specialized features such as *Tasks* in TinyOS and *Protothreads* in Contiki. These features are thought to minimize context switching and task scheduling overhead. Despite this "solution" at the software level may be useful in some cases, processor workload may still be very high and the overhead of introducing an operating system may not be acceptable; system responsiveness and performance may be compromised. One could think to introduce a more powerful micro-controller able to manage more efficiently the different tasks; however, leakage power becomes very significant which is undesirable in a WSN scenario.

Considering the concurrency and the event-driven nature of Wireless Sensor Networks designers are starting to rethink current hardware architectures. Introducing dedicated and specialized processing units, rather than a unique general-purpose processor could be a suitable solution for several WSN scenarios. Further power saving can be reached by introducing low power design techniques at the circuit level. Besides, some tasks can be efficiently executed in the analog world; for instance, sensing some variables, such as temperature, luminosity and even the channel power level, can be kept in analog circuitry. Digital circuitry can be activated only when an event is detected. Energy consumption of analog circuitry is very low compared to digital circuitry, then, analog components can be active continuously and events can be detected with minimum latency and energy efficiency.

By introducing specialized hardware, flexibility is traded off for performance and energy consumption. Additionally, design and deployment may become a complex task. However, there are tasks that are common to a wide set of WSN applications; then, it is possible to design a hardware platform suitable for a given set of applications allowing hardware re-usability. We can conceive the platform as a general-purpose processor connected to one or several hardware accelerators each of them executing a given task. Flexibility can be kept by using reconfigurable hardware; ultra low-power FPGAs are suitable for several WSN applications and advanced design tools allow faster deployment.

1.2 Contributions and organization of this thesis

Motivated by the former discussion we propose protocols and mechanisms that can take advantage of hardware acceleration and analog sensing of the communication channel to account for low power consumption. Such mechanisms are thought to facilitate the introduction of specialized hardware into the radio chip. Hence, the latter will be able to take decisions and carry out a given set of tasks without any intervention from the main processor. Then, the radio chip provides some services releasing the processor from executing tasks related to the protocol stack. It can decide whether the microcontroller must be waken up to handle a detected event. These radios, that are able to take decisions in an autonomous way, are what we call "smart radio".

We start by describing "Wake on Idle". It has been thought to be implemented as a hardware module that executes a well-structured protocol. Such protocol provides a set of services to the main processing unit. These services are: i) topology control, ii) synchronization and iii) low duty cycle medium access. All these tasks are then executed without any intervention from the main processor. The main characteristic of Wake on Idle is that it provides these services based only on the exchange of analog signals. Then, it does not require any digital processing *i.e* decoding/encoding, modulation/demodulation of the signal. It only requires sensing the channel energy level or RSSI. This characteristic may lead to additional energy savings and makes the protocol robust to collisions and external interference. Once nodes are synchronized they can assess each other's availability without the necessity of waking up the whole system. Moreover, Wake on Idle performs fairly accurate local synchronization; hence, very low duty cycles are reached.

During the evaluation of Wake on Idle we have observed the benefits of using analog signaling; we have seen that by applying some mechanisms it can be robust to interference while keeping low power consumption, *i.e.* no (expensive and consuming) digital processing is needed. Besides, by the introduction of smart radios it is possible to perform several tasks without activating the main processor and other peripherals. Thus, we further exploit these characteristics to optimise existing MAC protocols. The optimization method is called "Sleep on Idle", and it reduces duty cycle and minimizes idle listening through the use of analog signaling. For evaluation purposes, we have based the implementation of Sleep on Idle on the slotted-IEEE 802.15.4 standard and we have shown a significant gain in terms of power consumption, specifically for low traffic applications.

We also study how these mechanisms based on analog signaling can be implemented in hardware and integrated with the radio chip.

This report is divided in three parts: the state of the art and related work, the contributions of this thesis, and a third part formed of a set of discussions around experimentations and conclusions of this work. The state of the art, composed of Chapters 2 and 3, summarizes mechanisms used to optimize energy consumption in WSNs applications. We give an overview of techniques used at the circuit level and hardware design level for energy optimization while performance constraints are respected. Also, we discuss communication protocols that account for low power consumption. Then, in Chapter 4 we describe and extensively evaluate the Wake on Idle protocol. We also present a proof of concept implementation and we discuss a possible hardware imple-

mentation of the Wake on Idle module. Chapter 5 describes the functionality of Sleep on Idle and presents an evaluation of the protocol through simulations and experiments. Then, Chapter 6 illustrates, through the description of two cases of study, the importance of real experimentation when evaluating protocols and features in Wireless Sensor Networks. It also presents the challenges and difficulties that arrive during real experimentation. Finally, we conclude the document with an overview of the work carried out during this thesis and present possible future work and directions.

Part I

State of the Art

Chapter 2

Hardware Optimization for Wireless Sensor Networks

Contents

Wireless Sensor Networks design and deployment is a challenging task. Energy consumption is the most constrained design factor and, in order to minimize it, optimizations must be introduced at different levels: from platform circuit design to the protocol stack and application. The design of energy efficient WSNs involves several stages and an appropriate Hardware/Software co-design oriented to the concerned application.

The limitations of current hardware platforms used for WSNs applications have motivated the studies done during this work. Then, in this Chapter, we attempt to summarize platform design propositions aiming to achieve energy efficiency while keeping high performance. Hence, we concentrate this discussion on alternative hardware platforms along with their characteristics, advantages and disadvantages.

2.1 Current Wireless Sensor Networks platforms

Common hardware used in a Wireless Sensor Network "node" integrates a general purpose, low-power, microcontroller or processing unit with a set of peripherals. A general view of this hardware is illustrated in Figure 2.1. Examples of these motes are the Telos Tmote Sky [3], MicaZ [4], WSN430 [5] and Ez430 [6]. All of these motes are based on low-power microcontrollers. The 16-bits Texas Instruments MSP430 and the 8-bits ATmega128L microcontrollers are widely used. These processors provide a fair compromise between performance and power consumption for simple applications. However, as the microcontroller must execute both the application and the protocol stack, processing performance and speed may become an issue. Then, designers are introducing much more powerful (32-bits) microcontrollers. Examples of these platforms are the SunSpots [7] that use an ARM920T processor and platforms based on the ARM-CortexM3 processor core [8]. These processors can execute more sophisticated tasks

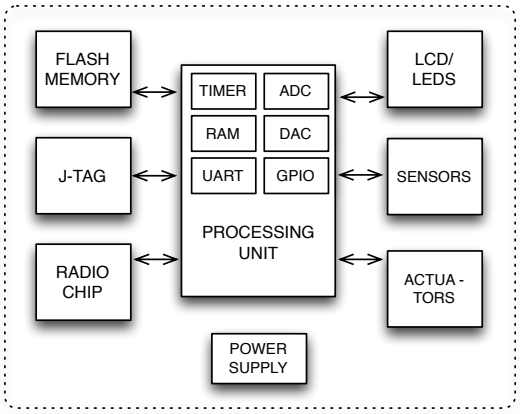


Figure 2.1: View of a classic mote

needed for cryptography, complex signal processing and advanced applications but, due to their power consumption, they may not be suitable for Wireless Sensor Networks [9].

2.2 Alternative hardware platforms for Wireless Sensor Networks

The necessity of having energy harvesting systems imposes highly constrained energy consumption requirements. For instance, when using vibrations for energy scavenging, designers have an energy budget of some hundreds of μW [10]. Even if software and protocol stack optimizations are used, current hardware platforms do not attain these power consumption requirements. Hence, researchers are starting to rethink existing hardware platforms used for Wireless Sensor Networks.

The use of a single, general purpose processor has been very attractive in Wireless Sensor Networks since a wide range of applications can be executed. Flexibility is obtained thanks to the ease of reprogramming the processor. However, these platforms are not suitable for Wireless Sensor Networks applications, which need the execution of multiple, simultaneous tasks and are event driven. Special lightweight operating systems have been proposed to account for performance and responsiveness [11]. These operating systems allow efficient execution of multitasking and event driven applications while using a single processor. Concurrency is achieved since context switching is done in an efficient way (or not done at all). For example ContikiOS [2] executes multiple threads by the use of protothreads, which are very light stack-less threads with very fast (and light) context switch. TinyOS [1] is an event-based operating system, where a unique thread is executed and tasks are called according to events. Development in such operating systems is not straightforward, some of them use special languages, developers have to take special care of resource allocation and they must have knowledge about the execution model followed by the operating system.

The overhead introduced by an operating system causes additional energy consumption and performance may still be an issue, due to highly constrained computational

resources. Then, flexibility could be traded off and the introduction of specialized hardware could help in order to reach the constraints imposed by a given Wireless Sensor Network scenario. These special hardware units could perform a wide spectrum of tasks, *i.e.* from sensing tasks to very complex signal processing tasks.

For instance, we can take as an analogy the design of Software Defined Radio (SDR) systems. Initially, SDR was thought to provide a completely programmable and flexible radio system [12]; hence, the initial idea was to implement the complete protocol stack (from the physical layer to the application layer) on a general purpose processor. Having a software implementation of the protocol stack was very attractive because of flexibility, but completely impractical since it requires very complex and computationally intensive tasks. For instance, the GNURadio [13] project attempts implementing the protocol stack on a very powerful general purpose processor with the help of minimum additional hardware; despite the efforts, GNURadio reaches data rates much lower than the ones imposed by the standards. Then, multiprocessor platforms and the use of specialized hardware units have been proposed to respect the standards' requirements while flexibility is sacrificed.

Proposed SDR platforms integrate a set of specialized hardware such as analog logic, ASICs and DSPs with reconfigurable hardware (FPGAs) and reprogrammable hardware (General Purpose Processor). Examples of these platforms are Sundance [14], KUAR [15], Lyretech [16], WARP [17] and FAUST [18]. Despite some limitations, these platforms reach very good performance since they exploit concurrency and parallelism.

This flexibility versus performance compromise can also be applied to Wireless Sensor Networks. A completely reconfigurable (waveform) radio cannot be used in this scenario given the constraints in power consumption. Then, tasks such as signal modulation/demodulation and encoding/decoding are executed on an ASIC or dedicated hardware provided by radio chip vendors. However, higher-level protocols can take advantage of a multiprocessor approach: "divide and conquer".

Following we try to extensively summarize the proposed hardware architectures for WSNs and we classify them according to their optimization strategy. We start by classifying hardware optimization techniques into two main categories: i) Analog techniques or analog wake up mechanisms and ii) digital techniques or hardware specialization. These two categories are not mutually exclusive and some platforms combine both techniques.

2.2.1 Analog wake up circuitry

One common application of WSNs is event detection; it requires monitoring of a certain set of parameters, *i.e.* temperature, humidity, vibrations, level, and so on. In a typical implementation, sets of sensors are connected to the microcontroller. Then, the microcontroller must wake up on a periodic basis to retrieve information measured by sensors. Sensors usually produce a voltage level proportional to the sensed value; then, an Analog to Digital Converter (ADC) and some signal conditioning are used to generate the proper digital value to be used by the microcontroller. After data retrieval, the microcontroller checks the sensed value and determines if an event has arrived, *i.e.* a defined threshold is exceeded.

In this approach there is a trade off between event detection speed or reactivity and power consumption. Reactivity of the system is determined by the sampling frequency, *i.e.* the frequency at which the microcontroller wakes up to retrieve sensor information. Frequent digital information retrieval may be very expensive in terms of energy consumption since it requires waking up the system and performing digital signal processing. Alternatively, high reactivity and low power consumption can be obtained by introducing wake up circuitry. The main idea is to keep environmental sensing in the analog world, then, by using a simple (tunable or configurable) comparator, a wake up signal is generated. The wake up signal triggers an interrupt that must be served by the microcontroller. Then, given its low power consumption, analog circuitry can be active all the time and the whole system can remain idle until an event is detected. Thanks to continuous sensing of parameters, events can be detected and served as they arrive.

TelosW [19] is an example of wake up circuitry. It is a modified version of the already existing Telos mote, *i.e.* it uses an MSP430 microcontroller and a CC1101 [20] radio chip. The novelty of the platform includes the introduction of a configurable wake up system for light and temperature sensing. The authors have shown the gains in reactivity and power consumption obtained thanks to wake up circuitry. Similarly, WINs [21] introduces continuous sensing and some hardware processing for event detection and identification.

An architecture that integrates wake up circuitry is illustrated in Figure 2.2. A set of sensors with their respective analog signal processing can be introduced to wake up the main processor. As seen in the figure, the radio chip can also be seen as a sensor; in fact, it senses the channel power level. Then, analog circuitry can be used to sense the desired frequency band and determine the presence of a transmission. If this is the case, the system is activated to start decoding incoming frames.

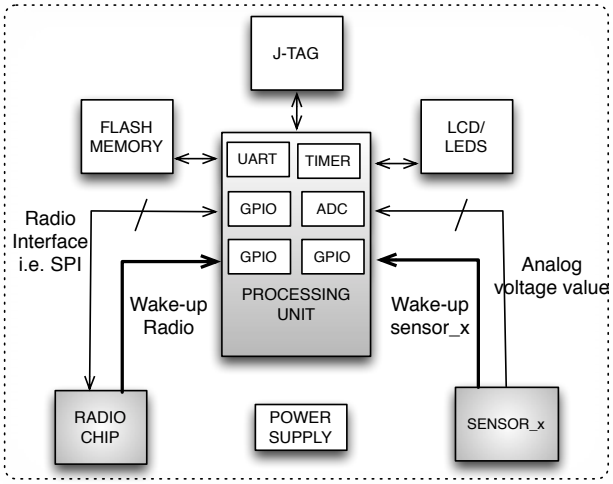


Figure 2.2: Overview of a wake up platform

Several radio wake up circuitry has been proposed. L. Gu *et al* [22] have proposed the integration of a radio-triggered wake up mechanism with existing platforms; such mechanism, composed of an antenna and basic analog circuitry, is able to trigger an interrupt signal to the CPU when a transmission is detected in the frequency of interest. Additionally, the wake up mechanism does not need any external power source since it is powered by the EM signal transmitted by the sender. Despite the authors did not present any implementation, simulations in Spice and NS2 have shown very promising results in terms of power consumption for low traffic WSNs. Later, a working prototype of a wake up radio device was presented by B. der Doorn *et al* [23]; using a simple modulation scheme (OOK: On Off Keying modulation) authors were able to provide wake up functionality in the 868 MHz frequency band. They have used a common radio chip connected to some analog processing circuitry and an additional low-power microcontroller.

Similarly, Hibernets [24] integrates ultra low-power analog circuitry such as band-pass filters with existing sensor nodes. The wake up system is seen as an analog co-processor. As a proof of concept, a filter bank has been implemented and integrated with existing hardware being able to detect low frequency signals at a very low power consumption. J. Ansari *et al.* [25] have proposed wake up device where a dedicated channel (and a second radio interface) is used to wake up the system. The authors use a simple pulse modulation technique to solve the addressing problem *i.e.* waking up only the packet destination. A second radio interface, using a more robust modulation, is used for communication.

V. Rosello *et al.* [26] have proposed another approach for efficient system wake up. The authors propose using a low-power (flash freeze) FPGA [27] to detect RF signals and destination address. Then, FPGA logic is in charge of generating the microcontroller IRQ and also the signal to wake up the receiver. The use of reconfigurable logic is very powerful and gives robustness regarding false wake ups. The table 2.1 summarizes energy consumption while listening and characteristics of some of the wake up radio approaches described here.

Table 2.1: Characteristics of some wake up radio solutions

Author	Range(m)	PowerWuRx(μW)	Characteristics
L. Gu [22]	9	145	RF powered
Doorn [23]	3	801	–
Ansari [25]	3	66	Addressing
Rosello [26]	3	94	Addressing + FPGA-based

Wake up radio circuitry is still very limited; sometimes it is limited to low frequencies or very short ranges. Further energy consumption optimization is required when energy harvesting is needed. Issues, such as false and missing wake ups and addressing, must be efficiently covered to have the desired performance and reliability.

2.2.2 Digital circuitry optimization

Different energy consumption optimizations can be introduced at the digital level. The selection of specialized hardware is closely related to the application and, according to complexity and constraints, a given hardware architecture must be chosen. We will classify these mechanisms into two different categories; these mechanisms can be combined to obtain further optimization.

a) Processor architectures for WSNs

Sets of applications in WSNs execute several common tasks: topology control and medium access protocols, data retrieval, transmission/reception and storage, cryptography, FEC and CRC coding. These tasks use a similar set of operations that are specific for WSNs. Hence, it could be possible to specify a processor architecture with a data path and a set of instructions suitable for a given group of WSNs applications. While there is a certain degree of flexibility, the micro-architecture will be optimized for some tasks. Significant gains in power consumption and code footprint can be achieved.

An early architecture implementation was done for the SmartDust project [28]. This microcontroller implements an 8-bit RISC load/store harvard architecture. A set of timers are in charge of managing independent units of the architecture; this characteristic allows clock gating of independent blocks. It is a very simple, low-performance architecture that reaches extremely low power consumption.

S. Mysore *et al.* [29] provide a design space exploration to characterize an ISA specialized for a class of WSNs applications. They have created a benchmark, called *WiseNBench* in order to design the proper architecture. L. Nazhandali *et al.* [30] have designed a pipelined processor architecture optimized for WSNs. The processor has a set of application specific instructions and addressing modes suitable for WSNs applications. Micro-architectural optimizations such as out-of-order execution and branch speculation have been added as well. Through simulations the authors have shown significant energy savings with respect to general purpose architectures.

Similarly SNAP/LE [31] and its successor BitSNAP [32] are pipelined processors with a set of special instructions optimized for WSNs applications. SNAP reaches energy efficiency and low latency by the use of an asynchronous architecture. This asynchronous approach eliminates critical path and other problems of having a common signal clock and makes the architecture suitable for event detection applications. Additionally, SNAP makes use of co-processing to process and generate data packets exchanged with other nodes. Also a timer co-processor unit is introduced. Extensive low-level processor simulation was carried out giving good results regarding energy consumption.

Table 2.2 summarizes the characteristics of some of the processors described here. Notice that the energy consumption given in the table is only for the microcontroller and not for the complete system. Despite the gain in power consumption, these architectures remain too general, the system must be woken up to respond to events and they keep the concept of having a unique processor managing all connected peripherals. The tasks carried out by the processor could still be too complex and cause performance and power consumption issues.

Table 2.2: Characteristics of some processors for WSNs

Name	(Max)Power(pJ/inst)	Width(B)	Characteristics
ATmega128L	3200	8	General Processor
TI MSP430	700	16	General Processor
SNAP/LE	218	16	GP RISC - Asynchronous
BitSNAP	152	16	GP RISC - Asynchronous
SmartDust	12	8	GP RISC

b) Architectures based on hardware acceleration

Hardware acceleration and specialized hardware can be introduced in order to release the main processor from several tasks carried out during the operation. It is then possible to have an event driven system where different processing units can take decisions independently from the main processor. The last can be turned on only when a rare condition is reached and complex signal processing is required.

For instance, hardware accelerators can execute simple and repetitive tasks that can be easily implemented in hardware logic. In fact, using a general purpose processor for these kinds of tasks causes a huge energy waste since the complete system is woken up, even if it is not used.

One simple example of the use of specialized hardware for co-processing is the Wake-on-Radio (WoR) functionality. Wake-on-Radio is a configurable module that wakes up the radio on a periodic basis. When the radio is woken up it senses the medium to determine the channel energy level. This periodic sensing is done without any intervention from the main processor. As soon as the radio finds the medium busy, *i.e.* there is an ongoing transmission, it generates an IRQ signal to wake up the processor. Hence, the tasks of waking up the radio when a timer event is triggered, programming a timer and polling the Radio CCA ping are moved from the processor to the radio device. The processor is woken up only to process data packets. The WoR functionality is being included in low-power, low data rate transceivers [33] and duty cycle protocols can take advantage of it [19]. A problem of existing Wake on Radio implementations is that, as packet destination is unknown, the microcontroller is activated often and unnecessarily. Energy consumption due to false and useless wake ups may be significant and additional logic to determine the packet recipient must be added.

Charm [34] sacrifices flexibility by introducing an architecture that implements in hardware a large portion of the communication stack. Hence the platform is formed by a set processing blocks, each of them implementing one layer of the OSI model. The application is then executed on a general purpose processor. Charm introduces a novel energy management system, reaching very low power consumption. However, significant standard and protocol changes require designing and resynthesizing new hardware, which could be expensive and time consuming.

The Harvard platform [35] is a system on chip that proposes integrating a set of hardware accelerators for tasks such as signal processing, network services and timer

programming. A processing unit for event detection and management is added and the main processing unit is used to serve rare events. A well defined architecture and an interconnection (bus) system are described and energy optimization techniques are implemented. Flexibility is reduced but authors argue that it is difficult to have a universal platform, given that application requirements may be very different.

LoMiTa [36] is a novel hardware architecture based on micro-tasks. These micro-tasks are processes implemented in hardware and that are controlled by a system monitor. Applications are executed on a general purpose microcontroller. Each micro-task executes a part of the WSN code and they respond to a given event; in this way, parallelism can be exploited reaching higher throughput. Micro-tasks are implemented using reconfigurable hardware *i.e.* FPGAs. LoMiTa eliminates the necessity of a central processor unit. Furthermore, it takes advantage of low-power techniques, such as power-gating, to reduce the power dissipated due to leakage current. Synthesis of some applications have shown significant power savings. A well-structured design flow has been studied by the authors [37, 38] and a framework for development is provided. Other authors propose the implementation of some components of the operating system in hardware to account for energy efficiency and performance [39]. Much effort is being put in unifying hardware and software design methodologies; for instance, Object-Oriented Programming (OOP) [40] and Aspect-Oriented programming (AOP) [41, 42] can be used to generate such hardware components.

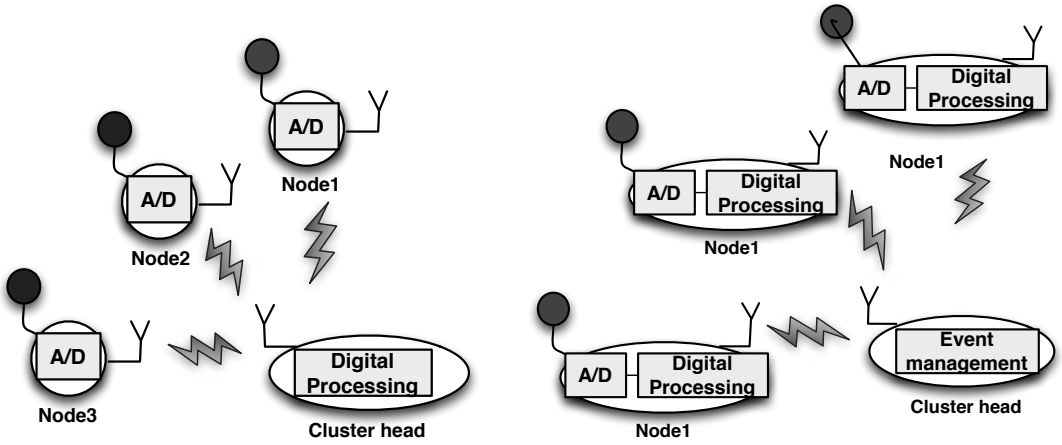
Communication is more expensive than computation. Transmitting a bit of information costs more than 100 times the cost of executing an operation on a low-power microprocessor [43]. Then the quantity of transmitted information must be minimized. Specialized hardware can be used locally to process information and communicate only when an event is detected. Applications that need complex processing, such as object tracking and recognition applications, can take advantage of specialized hardware such as DSPs, reconfigurable hardware or ASIPs to perform signal processing on-board, as shown in Figure 2.3.

DSPs are still too power hungry, but low-power reconfigurable technologies, such as flash-freezing FPGAs [44], are able to perform complex processing tasks while keeping low energy consumption. Rex *et. al* [45] have proposed implementing a distributed FFT computation where each (FPGA-based) node processes sensed data. Similarly, V. Vij *et al.* [46] propose the implementation of a Kalman filter for positioning systems in a distributed way, using FPGAs. Another example is presented by T. Kwok *et al.* [47], they have proposed executing the image processing required for object recognition in a distributed way, using an FPGA-based reconfigurable platform.

Leakage power accounts for a large fraction of total energy consumed by a WSN node. Then, further techniques to reduce leakage, such as dynamic frequency and voltage scaling and clock gating [48], can be added at the circuit design phase.

2.3 Summary and Conclusions

In this chapter we have summarized some optimization techniques introduced when designing a hardware platform for WSNs applications. We have shown that the introduction of additional analog processing or wake up circuitry may help reducing energy



(a) Centralized signal processing **(b)** Distributed signal processing

Figure 2.3: Distributed computations example: Communication can be reduced by processing sensed information locally

consumption and improving reactivity of the system. Existing wake up circuitry is still very limited and there are issues to be covered, but it allows reducing energy consumption by minimizing the time that digital circuitry is activated.

We have also presented some propositions aiming to design specialized processors for WSNs. Through the use of proper benchmarks and a design space exploration, a data path and a set of instructions can be specified. Another solution is using a general purpose processor connected to hardware accelerators or co-processors. Advances in FPGA design make possible the use reconfigurable logic with very low power consumption. These solutions lead to significant saving in energy consumption, but flexibility is reduced; then, for power optimization, hardware must be adapted to the application. The design flow may become a complex task since hardware resources must be efficiently exploited by the software. Also, a proper energy management system at the hardware and software level must be introduced to respect the highly constrained requirements of WSNs.

Chapter 3

Duty Cycling for Energy Optimization

Contents

The main sources of energy consumption in WSNs are leakage or stand-by power, computations and communication. In Chapter 2 we have presented propositions to minimize energy consumption during computations and to reduce leakage power.

Now, we move up the stack to the medium access control layer. The introduction of energy efficient medium access techniques is a fundamental task when deploying a WSN. The radio chip consumes a significant amount of power, then communication and data exchange must be as efficient as possible to reduce the time the radio is powered on. In this chapter we give an overview of medium access protocols proposed for WSNs and we classify them according to some criteria.

Since a part of our work is closely related to the IEEE 802.15.4 standard, we give an overview of it. Furthermore, we the describe IEEE 802.15.4e standard; it makes use of multiple channels and frequency hopping. These techniques introduce nice properties and characteristics that are discussed in Chapter 4.

3.1 Medium access in Wireless Sensor Networks

Efficient medium access is important to guarantee reliable delivery of packets and low energy consumption. The three main reasons of energy waste at the medium access control layer are [49]:

- *Collisions:* Two or more stations in the same transmission range transmit simultaneously. In this case, depending on the distance between nodes and the transmission power, one or several frames can be damaged and they cannot be correctly decoded by the recipient. Then, frames should be retransmitted.
- *Idle Listening:* The radio is turned-on, it listens to the medium, but there are no transmissions or exchanged data. This is the mayor source of energy waste.
- *Overhearing:* The radio receives a frame not intended to it. In this case the frame is processed and then it is discarded.

Collisions cause significant performance degradation, since they may avoid correct delivery of packets, introduce communication delay and further power consumption due to retransmissions. As the number of nodes (or traffic) present in the same transmission range increases, the probability of having collided frames increases as well. Collisions can be avoided by introducing arbitration mechanisms such as carrier-sense and/or resource allocation (TDMA, FDMA, CDMA). Collision avoidance has been extensively studied for classic wireless networks, and there is a wide variety of protocols. Due to its scalability and efficiency in terms of spectrum utilization and simplicity, carrier-sensing mechanisms are widely used and accepted. For instance, IEEE 802.11 [50] uses the CSMA/CA medium access protocol which is based on carrier-sense.

Overhearing and idle listening are not issues for common wireless systems where energy consumption is not a constraint. However, in a WSN these two sources of energy waste must be eliminated. Since the radio chip is the most power hungry element on a WSN node [51] designers aim at minimizing the time it remains active. Then, the energy wasted during overhearing and idle listening can be reduced by turning off the radio when there is not useful information to be exchanged: Nodes switch continuously between the active state and the inactive state following a determined schedule. This energy optimization mechanism is called *duty cycling* [52].

When using duty cycling, nodes willing to communicate must wake up simultaneously to exchange information. An optimal duty cycle medium access control protocol must turn on the radio only to: i) transmit frames while it guarantees that the proper recipient(s) is(are) listening and ii) receive frames that are intended to it. Moreover, collisions must be avoided.

The fraction of time the radio remains on is called *duty cycle* and it determines the energy consumed during communication. Minimizing the duty cycle is desirable since energy consumption is minimized but latency may be penalized. According to the application requirements and available resources, a compromise between energy consumption and latency or reactivity must be found.

A huge number of Duty Cycle MAC protocols has been proposed during the last ten years [53, 54, 52] and several classifications are found in the literature. For instance, according to the method used to schedule active/inactive periods we can classify these

protocols into three main groups: Synchronous, Asynchronous and Hybrid protocols. Also, according the way the communication is started, they can be classified into Sender-initiated and Receiver-initiated protocols.

Following we list some of the proposed protocols belonging to each category. We also list their advantages and drawbacks and we discuss the aspects that must be taken into account in order to have a working implementation of the protocols.

3.1.1 Synchronous duty cycle MAC protocols

In this family of protocols the sleeping/active periods of all associated nodes are synchronized. In general, nodes follow a common schedule. During the active interval, access to the medium can be coordinated by using either a contention-based mechanism, *i.e.* CSMA/CA or ALOHA, or a contention-free mechanism *i.e.* TDMA.

Advantages of synchronized protocols are:

- Since active periods are synchronized, the sender can be sure that the packet recipient is listening when the packet is sent. Also, overhearing is reduced.
- If synchronization is guaranteed it is possible to reach extremely low duty cycles. In fact nodes can attain very long inactive periods while correct delivery of packets is still possible.
- Keeping node synchronization and information about wake up schedules may help to maintain the network topology, *i.e.* synchronization beacons are periodically sent.

However, the necessity for synchronization may be quite costly. Resynchronizations are expensive thus a sufficiently accurate mechanism to account for clock imperfections must be implemented. The introduction of synchronization techniques increases the complexity of the protocol and adds some overhead due to the necessity of signalization. Moreover, the task of detecting new neighbors, synchronization and association could be long and energy consuming; then, flexibility and scalability are issues of these protocols.

a) Clock synchronization for Wireless Sensor Networks

Correct functionality of a synchronized low duty cycle MAC protocol requires the implementation of a synchronization mechanism. Some of the MAC protocols described here provide synchronization by sending periodic beacons called "synchronization beacons". Such frames are exchanged with the purpose of correcting timing errors added by clock imperfections *i.e.* clock drift and/or clock skew.

The *clock skew* is the difference of the speed between two oscillators. Then, a skew smaller than 1 means a slower clock while a skew larger than 1 means a faster clock. The *clock drift* is the difference between two clocks after a given interval of time. The clock drift causes that the value of two timers "drift apart" and it must be estimated and compensated on a periodic basis. Several implementations of synchronized MAC protocols introduce mechanisms to correct the estimation error, but, for simplicity, clock skew is not estimated. Thus, large margins, *i.e.* wake up in advance time, must be introduced to avoid loosing transmissions due to the difference of speed.

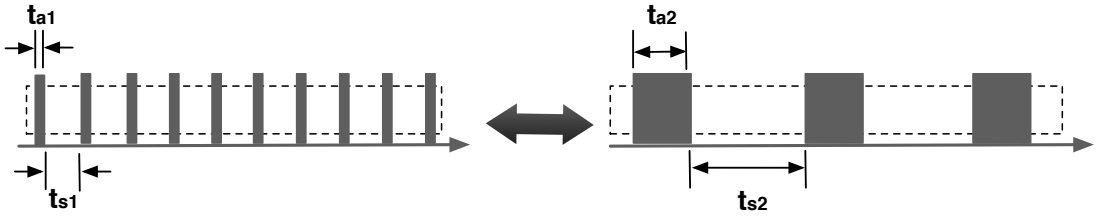


Figure 3.1: Duty cycle medium access control and synchronization: Tight synchronization allows frequent wake ups while respecting energy budget

Estimating and compensating clock skew adds some overhead but it helps to achieve a more precise synchronization.

Precise synchronization reduces energy wasted during idle listening due to wake up in advance margins. Then, nodes remain active during shorter periods and communication is still possible: duty cycle can be significantly reduced. Consequently, with the same energy budget it is possible to wake up the nodes more often while keeping the same energy consumption; hence, system reactivity can be improved. Figure 3.1 illustrates how reactivity can be improved thanks to tight synchronization; in fact, in the figure, the duty cycle in both cases is equivalent (Equation 3.1). However, nodes in the left side wake up more often and they can react faster to events and forward packets with a minimum delay.

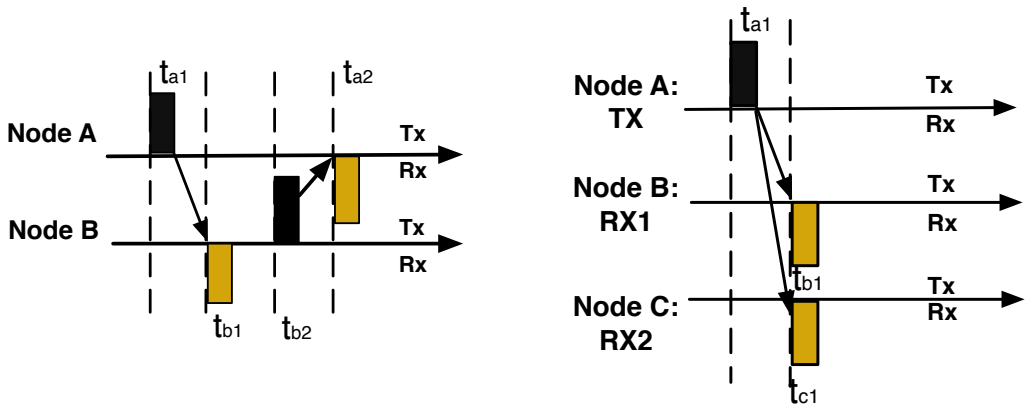
$$dc = \frac{t_{a1}}{t_{a1} + t_{s1}} = \frac{t_{a2}}{t_{a2} + t_{s2}} \tag{3.1}$$

The clock skew can be estimated by collecting a set of timing points. These points are generated by the exchange of frames and, when plotted, they form a line. The slope of this line determines the clock skew; thus, a method, such as a linear regression, may be applied to estimate this value. Then, clock drift is compensated periodically to keep synchronization.

For the sake of simplicity, WSN nodes perform local synchronization or post-facto synchronization. It means that, at the beginning of the association, nodes estimate the clock skew relative to their neighbors and keep the calculated value in memory. When an event arrives, *i.e.* a packet or a synchronization beacon is received, local timestamps are extrapolated to a reference time using local parameters. The difference between the local time and the time of interest is used to compensate the clock drift.

Synchronization has been widely studied, and many mechanisms have been proposed and implemented. Specifically, in the case of Wireless Sensor Networks, where energy consumption and complexity are constrained, several lightweight protocols have been proposed [55].

For instance, mechanisms such as Tiny-Sync [56] propose a simplification of a synchronization mechanism already proposed for classic network systems [57]. In fact, it uses a clock skew estimation mechanism other than a simple linear regression. Other mechanisms such as Timing Sync Protocol for WSNs (TPSN) [58] and Gradient Clock Synchronization [59] estimate clock skew by using a linear regression. Then, clock drift



(a) TX/RX packet exchange **(b)** RBS packet exchange

Figure 3.2: Packets exchanged to compute skew and offset: a)Typical Approach: Two-ways data exchange between the sender and receiver. b) RBS: Two receivers synchronize based on the sender transmission

is compensated on a periodic basis using synchronization frames or beacons. It has been shown that parameters chosen for linear regression may widely affect the synchronization accuracy [60].

On demand synchronization (ODS) [61] uses a white noise model procedure to estimate clock skew. Then clock drift is compensated by each node in an independent way. Contrary to common approaches, where clock drift is estimated on a periodic basis, an ODS node decides on a distributed way when clock drift should be compensated. This decision is done based on the previously computed white noise model.

For clock skew estimation, the frames exchange process is shown in Figure 3.2a. An alternative approach is presented in Reference Broadcast Synchronization (RBS) [62] which aims at synchronizing two receivers based on the arrival time of a broadcast frame sent by a third node, as shown in Figure 3.2b. After the broadcast reception nodes exchange their respective reception time-stamp. In this way, the (non-deterministic) sending time introduced by the radio chip and the medium access mechanism is eliminated and skew computation accuracy is improved.

In general, given the energy constraints, WSN nodes use low-power oscillators. These oscillators are inaccurate and very sensitive to changes in voltage and temperature. Then, Schmid *et al.* [63] propose using a more accurate Crystal Compensated Crystal based Timer (XCXT) in order to reach extremely low duty cycles while very accurate synchronization is kept. The authors study the cost of using more accurate oscillators in WSNs; however, these oscillators may be expensive and power hungry. Ganeriwa *et al.* [64] have proposed a method to determine clock uncertainty; hence, they are able to maintain synchronization with minimum signaling overhead and it achieves extremely low duty cycles with a minimum error.

b) Existing synchronous MAC protocols

Low-Energy Adaptive Cluster Hierarchy (LEACH) [65], Power Aware Clustered TDMA (PACT) and Traffic Adaptive Medium Access protocol (TRAMA) [66] are examples of contention-free synchronous protocols. These protocols use TDMA to coordinate transmissions during active periods common to all nodes. Due to the necessity of well bounded slots these mechanisms require tight synchronization; however, they do not provide a mechanism for synchronization. For correct schedule and slot assignment a synchronization protocol must be implemented. These protocols may be inflexible and, to allow efficient utilization of the spectrum, a dynamic slot allocation algorithm must be added.

Sensor MAC (S-MAC) [49] is the first contention-based synchronous mechanism proposed in the literature. In S-MAC nodes following the same schedule form clusters and medium access during the active periods is done using a CSMA/CA mechanism. To avoid collisions in high contention scenarios, a reservation method is added, *i.e.* RTS/CTS exchange, and synchronization is maintained by periodically exchanging SYNC packets used to compensate clock drift. Furthermore, S-MAC proposes a method to avoid overhearing by taking profit of the channel reservation mechanism.

The Figure 3.3a illustrates an example of S-MAC data exchange using overhearing avoidance. Synchronization in S-MAC does not take into account clock skew; hence, timing margins are relative large and it cannot reach very small duty cycles.

Timeout MAC (T-MAC) [67], Adaptive Duty Cycle MAC (AD-MAC) [68] and Dynamic Sensor MAC (DS-MAC) [69] aim at optimizing S-MAC duty cycle and/or latency by adapting it according to the traffic load.

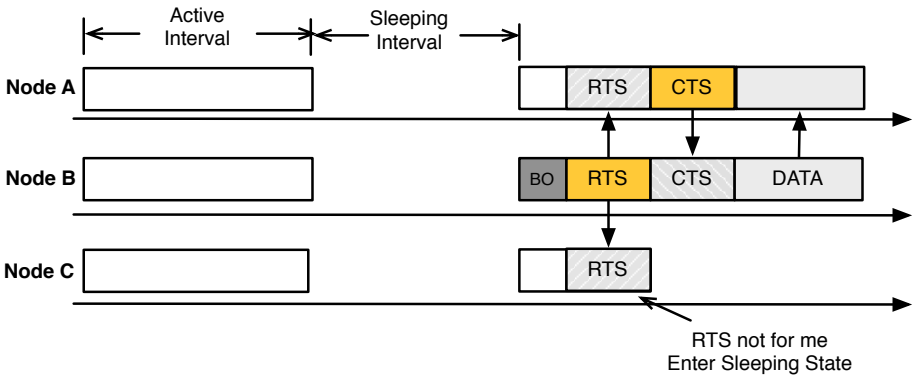
Scheduled Channel Polling MAC (SCP-MAC) [70] is the successor of S-MAC. It combines synchronization with Low-Power-Listening (LPL). LPL is a technique widely used in Wireless Sensor Networks, and it will be explained later. In SCP-MAC the listening periods of all nodes are synchronized. As soon as a node has a packet to transmit it wakes up its neighbors by transmitting a tone when nodes enter the listening period. When the tone is detected nodes remain active waiting for a transmission. As S-MAC, SCP-MAC uses CSMA/CA and a reservation mechanism during contention periods. Burst transmissions in SCP-MAC are possible by introducing an adaptive listening mechanism. Authors showed that, thanks to the introduction of LPL, SCP-MAC reaches a duty-cycle reduction by a factor of 10 with respect to S-MAC. Figure 3.3b shows a data exchange example using SCP-MAC.

Another example of contention-based, synchronous duty-cycle MAC is the slotted-IEEE 802.15.4 [71]. A detailed description of the slotted-IEEE 802.15.4 is given later in 3.2.

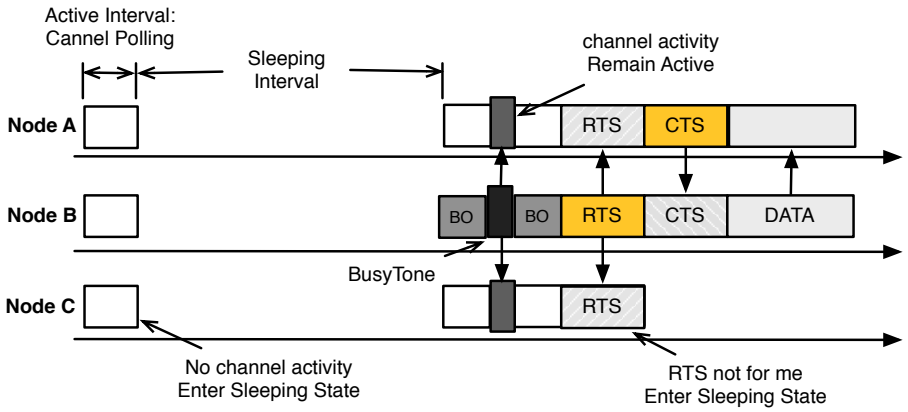
3.1.2 Asynchronous duty cycle MAC protocols

In an asynchronous protocol all nodes in the network have independent active/sleeping periods *i.e.* no synchronization of active periods is required. Then, as soon as a node has a packet to transmit it waits for the receiver to enter the active period or it attempts to wake it up.

The main advantage of asynchronous approaches is that there is no overhead to maintain synchronization, nodes just switch between active and inactive state on a



(a) S-MAC functionality



(b) SCP-MAC functionality

Figure 3.3: S-MAC vs SCP-MAC: a)*S-MAC:* Active period is fixed when no activity: Idle-Listening. b)*SCP-MAC:* Channel Polling allows early sleep when no activity

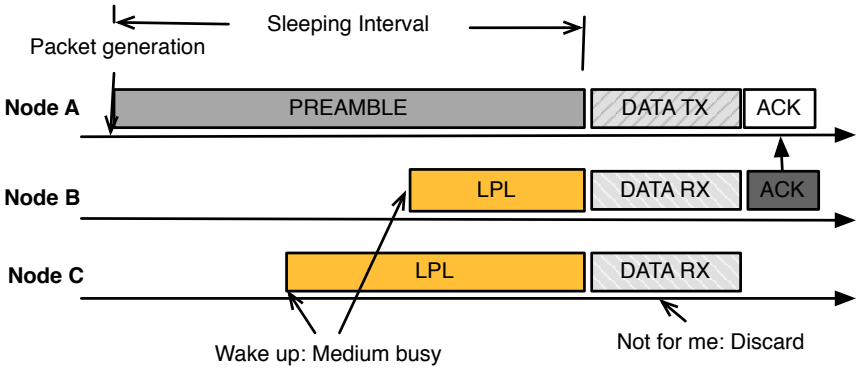
periodic basis. Also, they do not need to memorize any schedule or know timing parameters relative to each neighbor.

However, to allow communication, both sender and receiver must be in active state and, given the lack of synchronization, a mechanism to activate both nodes must be added. Also, asynchronous protocols may suffer schedule conflict problems as explained later in this section. Additionally, since all nodes follow independent schedules, the transmission of broadcast and multicast frames may be expensive.

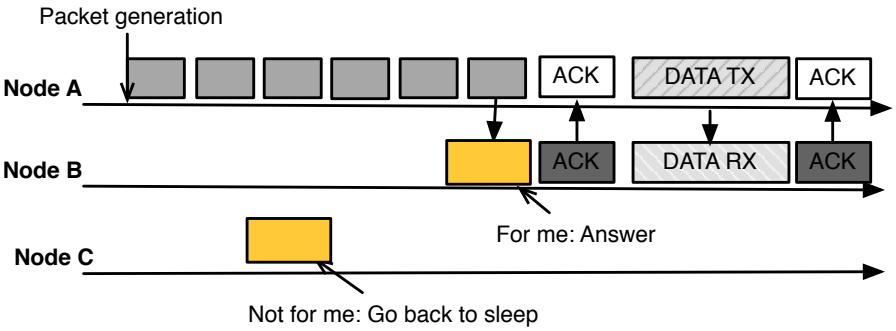
The pioneer of asynchronous protocols is Berkeley MAC (B-MAC) [72]. B-MAC introduces the notion of *Low Power Listening* (LPL) and *Preamble Sampling*. LPL is a wake up mechanism that helps optimizing power consumption; during LPL the node senses the medium to determine the channel energy level, based on this value the node decides if there is an ongoing transmission. In the case it finds the channel busy it remains awake to decode the packets being transmitted. During LPL nodes do not decode any frame.

In B-MAC as soon as the packet is generated, the sender wakes-up and starts transmitting a long packet. This packet is called *Preamble*. When neighbor nodes wake up they listen to the medium (LPL) and if there is channel activity they remain awake (Figure 3.4a) performing preamble sampling, *i.e.* checking the CCA signal, until a packet transmission is detected. Preamble transmission is done to wake up all neighboring nodes and its duration must be at least the duration of the sleeping period.

After the transmission of the (long) preamble, the sender is sure that the recipient is awake and it sends the packet. B-MAC is a very simple protocol since it does not need synchronization nor schedules storage. However, it has several drawbacks. Firstly, much energy is wasted due to the transmission and listening of long preambles. Secondly, all neighboring nodes are woken up and even if they are not the receiver they should remain awake and decode the frame after the preamble. In fact, a huge quantity of energy is wasted due to overhearing. Also, transmissions of long preambles increase contention and reduce throughput. Moreover, the protocol cannot be implemented in common, packet-oriented, radio chips, such as CC2420 [73]; since long preambles transmissions require sending packets of infinite length.



(a) B-MAC functionality



(b) X-MAC functionality

Figure 3.4: Examples of Asynchronous MACs (Node A wants to send a packet to Node B): a)*B-MAC:* LPL and Preamble Sampling b)*X-MAC:* Micro-frames

The successor of B-MAC is called X-MAC [74]. This protocol was designed to cope with the overhearing problem of B-MAC. In X-MAC, instead of transmitting a long preamble frame, the sender transmits a series of small frames, called *Micro-frames*. The Micro-frames contain information about the packet destination. As soon as a neighbor wakes-up, it decodes the Micro-frame and if it is the recipient it sends immediately an acknowledgement. If it is not the recipient, it goes back to sleep. When the sender receives the acknowledgement informing about the availability of the destination, it sends the packet (Figure 3.4b). This early acknowledgement mechanism reduces overhearing: only the packet destination is activated; also, the sender saves energy since, most of the time, it does not have to send the preamble during the whole inactive period. However, since schedules are completely asynchronous, the sender may consume lots of power trying to wake up the receiver. This power consumption increases as the inactive interval is longer. Also, X-MAC still suffers from high contention due to transmission of Micro-frames.

Box-MAC1 [75] tries to optimize power consumption by adding the data payload into the Micro-frames. Box-MAC2 [75] mixes low power listening with micro-frame transmissions: as soon as the node wakes up it listens for channel activity; if there is an outgoing transmission it starts decoding, else, it goes back to sleep.

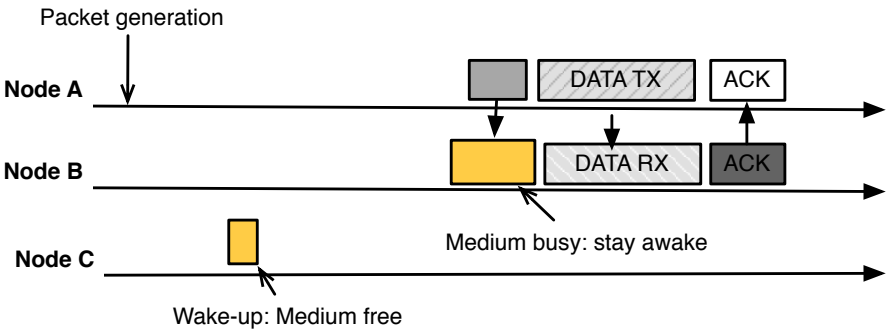
3.1.3 Hybrid MAC protocols

Using duty cycling while there is no knowledge about the schedule of neighbors may be inefficient. There is a huge quantity of energy wasted at the sender side and contention may be an issue. Also, the longer the sleeping period, the higher the energy waste due to the necessity to transmit wake up frames. Hybrid protocols overcome these issues by attempting to synchronize with neighbor schedules.

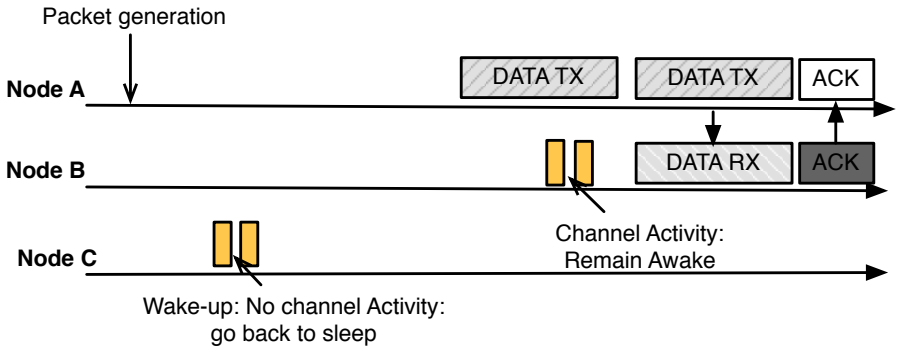
In this category, nodes still keep independent schedules, so they wake up in an asynchronous way; but, once they exchange information, they attempt to memorize wake up schedules of their neighbors. Then, a sender is able to predict the wake up time of the packet destination; when the predicted time arrives, the sender starts sending its wake up frame. The process of memorizing and predicting the wake up periods of neighbors is usually called *phase lock*.

Hybrid protocols have the advantages of both synchronous and asynchronous protocols. In this case clock drift must still be estimated to maintain "synchronization". However, if there is no traffic or data exchange between nodes after a long period of time, phase lock is lost. This is due to delays introduced by clock drift and skew that are not corrected. In fact, in a very low traffic application, the synchronization process must be restarted each time the nodes exchange packets.

The first proposed hybrid protocol is Wise-MAC [76]. As B-MAC, Wise-MAC uses long preambles to wake up the frame destination and, in addition, it implements phase lock to attempt synchronization with neighbors. Phase lock helps to minimize the time spent by the sender in preamble transmission. Similarly, W-MAC [77] adds a phase lock mechanism to the X-MAC protocol. Also, W-MAC adds a procedure to estimate clock skew relative to each neighbor reaching precise synchronization and very low duty cycles.



(a) Wise-MAC functionality



(b) ContikiMAC functionality

Figure 3.5: Examples of Hybrid MACs (Node A wants to send a packet to Node B): a) *WiseMAC:* Preamble sampling and phase lock b) *ContikiMAC:* Data packeting, phase lock and LPL

ContikiMAC [78] is another hybrid protocol that combines several techniques to allow very low-duty cycles. A ContikiMAC sender uses phase lock to account for synchronization and sends replicas of the data packet to wake up the receiver. A ContikiMAC receiver performs LPL to detect channel activity; then, it decodes the frame and answers with an acknowledgement if it is the intended recipient. When there are no exchanged packets, ContikiMAC reaches duty cycles of the order of 0.1%. The duty cycle value depends on the sampling frequency or how often the node wakes up to listen the medium.

3.1.4 Sender-initiated and receiver-initiated MAC protocols

In most of the protocols described above, when a node needs to transmit a frame it attempts to wake up the packet recipient, informing about its willingness to transmit. This way of communication, where the sender starts the communication, is called *sender-initiated* communication. These protocols make use of wake up mechanisms such as preamble sampling, micro-frame transmissions and LPL.

Sender-initiated protocols making use of LPL may suffer from some problems. For instance, transmissions from other nodes and external interference cause false positives waking-up nodes uselessly. To cope with the false positives problem, *receiver-initiated* protocols were proposed. In these protocols, nodes send a probe at the beginning of their active period, this probing mechanism is called *Low Power Probe* (LPP). The probe is used to inform neighbors that the node enters the active state and it is available to receive frames. As soon as a sender node has a packet to transmit, it listens the medium looking for a probe from the receiver node. When the probe is received it starts the frame exchange process.

Receiver Initiated MAC (RI-MAC) [69] is the first receiver-initiated protocol proposed in the literature. The authors do not implement any phase lock mechanism but they have shown that RI-MAC behaves better than its sender-initiated counterpart: X-MAC. Energy saving can be reached by avoiding useless micro-frame transmissions. Broadcast transmissions require sending the same data packet several times, until all receivers in the transmission range have received the packet. The successors of RI-MAC are O-MAC [79], Predictive Wake-up MAC (PW-MAC) [80] and Efficient Multichannel MAC (EM-MAC) [81]. The former protocols use wake up schedules based on pseudo random sequences and implement a simple version of the on demand synchronization mechanism where nodes decide when clock drift must be compensated. Authors have shown that these protocols outperform sender-initiated, phase locked protocols such as Wise-MAC. Also, it has been observed that the use of Pseudo Random sequences to generate wake up schedules can be very useful to cope with problems such as external interference and schedule conflicts. Also, as pairs of nodes are synchronized, frequency hoping can be easily added. PW-MAC and EM-MAC are closely related to our work, so they will be further discussed later in this document.

Dual Wake up (DW-LPL) [82] proposes integrating both receiver-initiated and sender-initiated mechanisms. Here, classical LPL is used for broadcast transmissions and LPP is used for unicast transmissions.

Another MAC (A-MAC) [83] is a receiver-initiated protocol that exploits hardware acknowledgements present in current radio-chips. Through a tricky manipulation of nodes address, A-MAC is able to reduce duty cycle while robustness to false positives is significantly improved.

3.1.5 Duty cycle schedule conflicts

In an asynchronous (or hybrid) duty cycle MAC each node chooses an independent schedule. When nodes chose exactly the same wake up/sleeping schedule communication is highly affected since they may interfere with each other during their active interval. Lei Tang *et al.* [80] have studied the effects of conflicting schedules in WiseMAC; they have shown that duty cycle is greatly increased and packet delivery drops. The implementation of a mechanism to "desynchronise" schedules and offer a conflict-free network is important. Protocols such as O-MAC and PW-MAC do not suffer from conflicts since nodes wake up based on a pseudo-random sequence unique to each pair of communicating nodes. Asynchronous Scheduled MAC (AS-MAC) [84] integrates LPL and LPP, and it aims at avoiding schedule conflicts by the use of hello frames announcing schedules; then, a node willing to join the network must choose a new schedule.

Additionally, mechanisms, such as DESYNC [85], provide a way to organize schedules so that transmission windows do not overlap.

3.2 The IEEE 802.15.4 standard: An overview

IEEE 802.15.4 [71] is a standardized MAC protocol for low-rate, Personal Area Networks (PAN). An important part of the work and experimentations carried out during this thesis work are related to the IEEE 802.15.4 standard; hence, we consider important to include a description of it.

The IEEE 802.15.4 standard defines two different operating modes: i) the *unslotted* mode, where no mechanism for energy consumption optimization is proposed *i.e.* nodes are active all the time, and ii) the *slotted* mode, which implements a duty cycle mechanism. Since this work is focused on power consumption optimization, only the second mode is described.

We will start by defining some basic concepts that are useful to understand the IEEE 802.15.4 standard functionality. Then we will explain how communication is done using the slotted scheme.

3.2.1 IEEE 802.15.4 components

In the standard two types of devices are defined: Full-Function Device (FFD) and Reduced-Function Device (RFD). FFD nodes can operate in three different manners: i) PAN coordinator, ii) Simple coordinator and iii) Normal device. RFD nodes are used as sensors that transmit sensed information to its coordinator (periodically or as a response to an event). Then, FFD devices can communicate with both FFD and RFD devices; but RFD devices can only communicate with FFD devices *i.e.* its coordinator. Examples of topologies that can be formed in IEEE 802.15.4 are shown in Figure 3.6. The slotted functionality allows only cluster tree and star topologies.

3.2.2 IEEE 802.15.4 PHY management services

These services are provided by the PHY layer and allow higher layers executing some control tasks and accessing important information. The main PHY management services are listed below.

- *Energy Detection (ED-RSSI):* It gives the power level (usually in dBm) of the radio channel.
- *Link Quality Indicator (LQI):* It indicates the quality of the received packets. Usually this value informs how difficult was to decode symbols in the packet.
- *Clear Channel Assessment (CCA):* It indicates if the channel is busy *i.e.* there is an ongoing transmission. Three CCA modes can be defined: i) *Energy Detection (ED) mode*: the medium is sensed as busy if the channel power level is higher than the predefined CCA threshold, ii) *Carrier Sense (CS) mode*: The medium is sensed as busy only if there is an IEEE 802.15.4 transmission going-on and iii)*ED plus CS mode*: the medium is sensed as busy if one or both conditions described later are true. Higher layers must be able to modify the CCA mode.

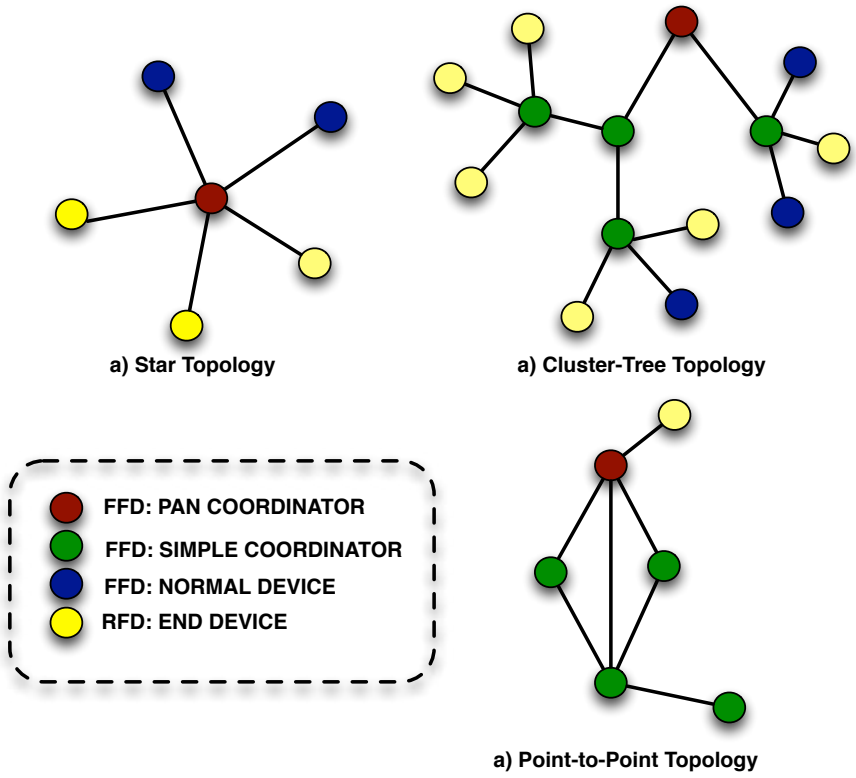


Figure 3.6: IEEE 802.15.4 topology examples

- *Channel, CCA threshold and Transmission power control*: These values should be chosen according to the application and the operation environment. Also they may be accessed and modified by higher layers at runtime. These parameters may allow frequency agility and the implementation of algorithms able to adapt to environmental and channel conditions.

An IEEE 802.15.4 compliant transceiver must provide these services in order to fully implement the standard.

3.2.3 Slotted IEEE 802.15.4 functionality

As mentioned before, the slotted IEEE 802.15.4 mode optimizes the energy consumption by implementing duty cycling. Here, associated nodes adopt the schedule of their coordinator (parent); hence, all associated and coordinator nodes activate their radio at the same time, in a synchronized way. The coordinator indicates the beginning of the active period by the transmission of a beacon frame containing information such as the coordinator schedule, the chosen values for different parameters, the coordinator identity and the features supported by the coordinator. Also, the coordinator adds in the beacon information about the frames waiting to be sent to the associated nodes. After the reception of a beacon, associated nodes are informed about the start of the active period where they can contend to access the medium.

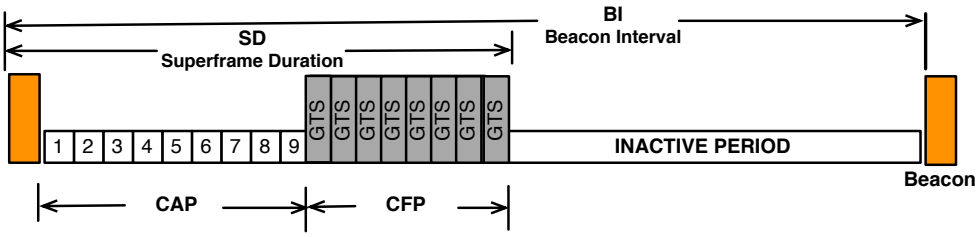


Figure 3.7: Superframe Format

The active period is divided into time slots, the set of time slots is denominated "Superframe". Following, the Superframe format and a brief description of the contention-based medium access mechanism used during the active period are given.

a) Superframe Format

The Superframe determines how the time slots are distributed during the active period. The format of a Superframe is illustrated in Figure 3.7; the relationship between time parameters can be observed in the figure.

The Superframe is formed of a Contention Access Period (CAP), a Contention Free Period (CFP) and an inactive period. The active period is divided into 16 slots whose duration is determined by the *aBaseSuperframeDuration* and the *Superframe Order* values. The first n slots are assigned to the CAP, the rest are assigned to the CFP. During the CAP, nodes use a special CSMA/CA protocol to communicate with the coordinator; optionally, nodes can reserve a set of slots for granted transmissions during the CFP. During the CFP period a TDMA-like protocol is implemented.

The *Superframe Order (SO)* parameter determines the duration of the active period (*Superframe Duration*) and the *Beacon Order (BO)* parameter determines the interval of time between two consecutive beacons (*Beacon Interval*). The timing relationship is given by equations 3.2 and 3.3.

$$SD = aBaseSuperframeDuration * 2^{SO} \tag{3.2}$$

$$BI = aBaseSuperframeDuration * 2^{BO} \tag{3.3}$$

From these parameters the duty cycle value can be computed ($DC = \frac{SD}{BI}$). All timing parameters described above are chosen at deployment time and they remain unchanged.

b) Data exchange during the Superframe

The communication between the Coordinator and its associated nodes is indirect. It means that when the coordinator has data to be transmitted to one node, the latter should request the transmission to the Coordinator *i.e.* data request frame. This request is done during the contention period. After the data request reception, the coordinator responds with the data frame. Nodes also transmit data packets to the coordinator during the active period. Hence, if the node does not have pending frames

to exchange with the coordinator, it can turn its radio off before the end of the active period; allowing further energy savings.

c) Slotted CSMA/CA Algorithm

During the CAP, nodes contend to access the medium using an special CSMA/CA algorithm. Here, before it starts transmitting, a node must compute a random value in the interval $[0, 2^{BE-1}]$ *i.e.* back-off value. This value determines the number of back-off slots (320μs each slot) the node must wait before it attempts to access the medium. After the back-off period, the node senses the medium (CCA) twice on the boundary of a back-off slot. If the medium is found free during the two sensing slots, the node can start transmitting its frame. Otherwise, it doubles the back-off interval, determined by BE (*Back-off Exponent*), it computes a new random value and it defers the transmission of its frame. If the node attempts to access the medium, without success, during more than a given number of times it discards the packet.

3.3 The IEEE 802.15.4e standard

The IEEE 802.15.4e standard [86] proposes the use of multiple channels (frequency hopping) as a way to increase robustness to external interference and to channel conditions that may avoid proper communication between nodes. The implementation of the standard imposes some challenges; for instance, to account synchronization and nodes rendezvous, nodes must exchange additional data, such as information about the pseudo random sequence for channel selection and timing information. Such information is encapsulated (*i.e.* forming an Information Element) and added into the beacon payload; then, it is processed by the MAC or higher layers. Beacons allowing the introduction of Information Elements are called *Enhanced Beacons*.

Two operating modes are described in the IEEE 802.15.4e standard: the *DSME* or Deterministic and Synchronous Multichannel Extension and the *TSCH* or Time Slotted Channel Hopping mode. Following there is a summary of the characteristics of each of these modes:

3.3.1 DSME

This protocol is similar to the classic slotted IEEE 802.15.4. The notion of Superframe is kept, nodes communicate based on multisuperframes or set of Superframes (Figure 3.8). As in IEEE 802.15.4, the Superframe is delimited by a beacon or an enhanced beacon, followed by a contention access period where nodes compete to access the channel using the slotted CSMA/CA mechanism and a contention free period where pairs of nodes can communicate during guaranteed slots on a given channel. Beacons are transmitted by coordinators; star, mesh and peer-to-peer topologies are supported. The beacon transmission and data exchange during the contention access period are done using a unique common channel.

Given the use of multiple channels, several pairs of nodes can communicate during the same time slot, using a different channel for transmissions. Each node maintains information about the slot times and the associated channel relative to a given neighbor.

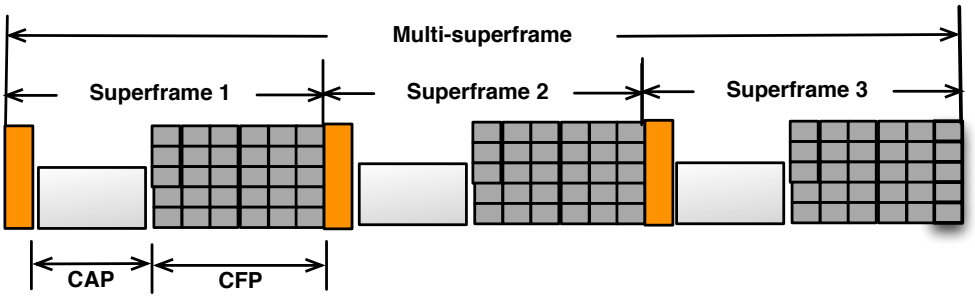


Figure 3.8: Multisuperframe Format

Furthermore, frequency hopping or channel adaptation can be added to account for channel degradation.

3.3.2 TSCH

In this mode, the concept of Superframe is replaced by *Slotframe*. A Slotframe is a slot of time (and channel) where nodes can communicate; the duration of a Slotframe is the minimum time needed to transmit a packet with its respective acknowledgement. Nodes are synchronized and they exchange information during common slots; they use a pseudo-random sequence to select the communication channel. Slotframes can be dedicated to a pair of nodes or it can be shared. In the last case, a simple CCA algorithm is used; exponential back-off is performed when frames are not acknowledged.

To keep a common sense of time nodes synchronize their clocks with at least one neighbor; in this way correct slot boundaries can be kept. Resynchronization (*i.e.* clock drift correction) can be done using exchanged packets or acknowledgements and it should be performed at least once per keep alive period. The keep alive period is defined when designing the network. Since multiple channels are used, several pairs of nodes in the network can exchange packets in the same slot of time (concurrent transmissions); they just have to agree on a common channel offset and a slot of time during which they want to communicate.

An example of a network using TSCH is shown in Figure 3.9. In the figure different patterns denote a given Slotframe where a given pair of nodes communicate; the dashed arrows illustrate nodes that are synchronized. Pairs of nodes agree on a rendezvous communication channel and time slot. The Slotframe assignment (*i.e.* time slot, type of Slotframe and channel offset) is not straightforward; this task is done by a high-level layer which must take decisions according to the application requirements such as delay, throughput and energy budget.

3.4 Summary and Conclusions

In this chapter we have discussed the optimizations introduced at the medium access level to allow low energy consumption. The characteristic common to all the proposed mechanisms is the idea of turning off the radio most of the time, *i.e.* duty cycling.

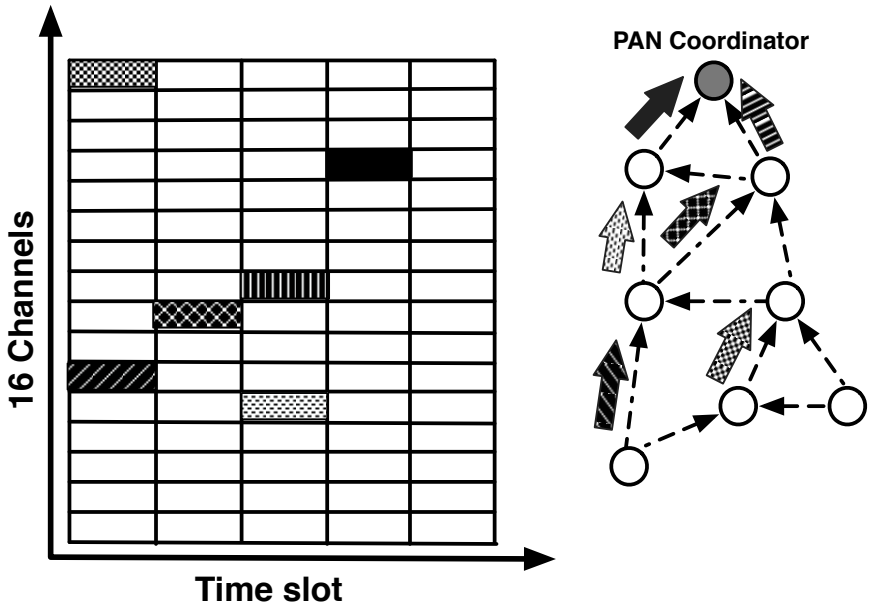


Figure 3.9: TSCH network example

We have then classified the proposed protocols and we have discussed the challenges of implementing them: synchronization, efficient wake up and schedules conflicts. Then we have given some details of the IEEE 802.15.4 and the IEEE 802.15.4e standards.

Each group of protocols presents advantages and disadvantages. We believe that the proper medium access control protocol must be chosen according to the scenario. However, energy consumption can be optimized *i.e.* overhearing and idle listening are minimized, only if a certain degree of synchronization is kept or if nodes have knowledge about the schedules of their neighbors.

Part II

Contributions

Chapter 4

Wake on Idle: Energy Efficient Neighborhood Maintenance and Medium Access

Contents

In Chapter 2 we have summarized some of the optimization techniques that can be adopted when designing hardware platforms for Wireless Sensor Network applications. We have also talked about "smart radios" or the possibility to introduce intelligence to the radio chip. The main objective of these smart radios is releasing the CPU from some tasks related to the communication process. Then, it is possible to keep the CPU inactive while some tasks related to the communication protocol stack are performed. In this chapter we propose a mechanism that has been conceived to be executed in the smart radio. The mechanism is called *Wake on Idle*.

Wake on Idle provides a set of services at a very low cost in terms of complexity and energy consumption. The main characteristics and services offered by Wake on Idle protocol are:

- It provides neighborhood maintenance or topology control through the use of analog channel sensing; it means that, by the use of short analog busy tones, nodes are able to track each other's availability.
- It maintains local synchronization between a node and its neighbors.
- It provides efficient wake up for communication relying only on analog signal detection. In this way medium access is provided while keeping very low duty cycle.

In this chapter we describe Wake on Idle and we explain how it exploits analog signaling. Then, we present an exhaustive evaluation of the proposed protocol through theoretical analysis, simulations and a proof of concept implementation. Finally, a possible hardware implementation of Wake on Idle is discussed.

4.1 Overview of Wake on Idle functionality

Wake on Idle was thought to be implemented at the very low-level, near the radio chip to leverage the conception of the smart radio. Hence, the protocol exploits efficiently analog signaling, *i.e.* simple transmission and detection of busy tones without any digital processing of transmitted signals. In Wake on Idle a pair of associated nodes agree on a common pseudo-random sequence; then, based on times generated by this sequence, nodes exchange beacons or maintenance frames. These beacons are used by the node to inform its neighbors about its presence and its position in the pseudo-random sequence. This information is used to synchronize the nodes.

If a node "detects" several consecutive beacons, it can deduce with a very high probability that the generator of the beacon is the respective associated neighbor. Hence the node identity is "time encoded" by transmissions at pseudo-random instants. In this way, nodes can track the presence of the associated node just by detecting a transmission at well determined times and no digital processing of the beacon is needed. Then, the tracking process requires only the detection and transmission of analog signals or analog

"busy tones". Wake up Radio circuitry described in 2.2.1 can take advantage of Wake on Idle, since it solves the addressing problem and permits efficient wake up of nodes.

Once nodes are associated, the process of exchanging beacons provides a mechanism for "neighborhood maintenance", *i.e.* tracking the presence of neighbor nodes, while local synchronization is kept even in the absence of data communication. The mechanism could be completely executed in dedicated hardware and integrated with the radio chip to reduce CPU utilization; it follows continuously a repetitive process: i) generate the pseudo-random value, ii) program a timer and iii) track and generate the busy tone when the timer fires.

4.2 Efficient neighborhood maintenance protocol

4.2.1 Motivation and background

Neighborhood maintenance or topology control mechanisms are introduced to assess the presence of neighbors and to have knowledge of possible packet recipients and potential routes. This is an important task to guarantee the proper delivery of information; especially in Wireless Sensor Networks that remain unattended during long periods of time.

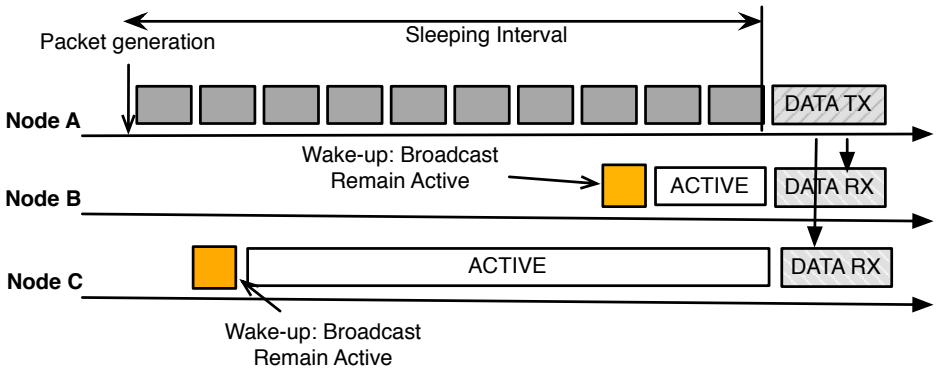
In a static scenario, once the topology is formed, one expects that it remains unchanged during long intervals of time and associations between nodes remain stable even though the signal quality might fluctuate. However, hardware failures or nodes displacement may arrive and nodes have to be aware of these events. The detection of such events is provided by the neighborhood maintenance protocol that, in general, is executed at high-level layers. Given the constraints of these scenarios, the addition of such mechanism introduces a significant overhead in terms of data exchanged between nodes and increases complexity when implementing the communication stack.

Typically, neighborhood maintenance protocols use periodic beacons or advertisements to allow nodes assessing the presence of neighbors and collecting information about possible packet recipients. In general, routing protocols are in charge of this task [87, 88] and they rely on the MAC layer to transmit these beacons.

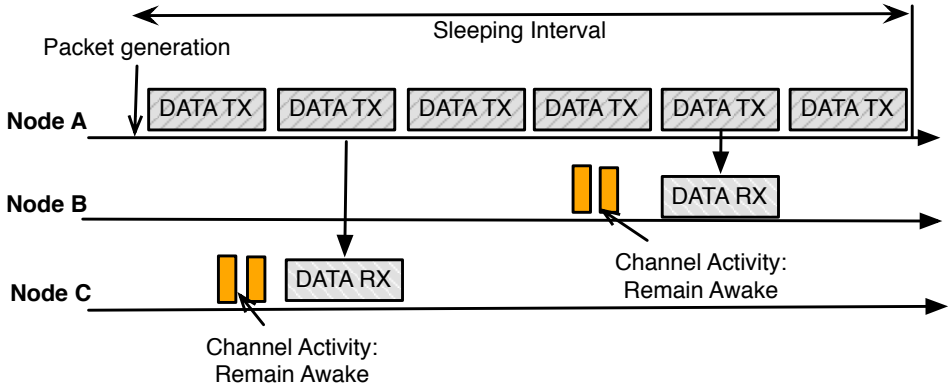
Thus, in a typical approach, each node must: i) generate maintenance beacons on a periodic basis and transmit them, relying on the MAC layer, as soon as the node enters the active state, ii) collect beacons from neighbors at the routing level, relying on the MAC layer; determine the beacon identity and set-up a time-out for each neighbor. If after this time-out no beacon is observed, the node infers that the neighbor is not present any more. Additionally, a mechanism to maintain routes and keep information about possible packet forwarders must be implemented. Routing and topology control may be complex and power consuming given the overhead introduced for signalization. Lets take a look to some aspects that motivates the introduction of Wake on Idle for neighborhood maintenance:

1. *Broadcast Transmissions are expensive*:

   Advertisements are transmitted as broadcast packets so that they can reach all nodes in the range of the transmitter. Routing protocols rely on the MAC layer to transmit these broadcast frames.



(a) WiseMAC



(b) ContikiMAC

Figure 4.1: Broadcast transmissions using hybrid duty-cycle MACs

To visualize the cost of a broadcast transmission, lets discuss the case on which an asynchronous, sender-initiated, duty-cycle MAC is used. In this case, all nodes have independent schedules; hence, when sending a broadcast frame, the transmitter must wake up all its neighborhood: it must remain awake, transmitting its wake-up signal, during the whole sleeping period. According to the chosen duty cycle protocol, the receivers must remain awake until all neighbors wake up and the broadcast frame is sent. Figures 4.1a and 4.1b show an example of a broadcast transmission when using WiseMAC and ContikiMAC respectively.

In fact, in all asynchronous duty cycle protocols, the power consumed during a broadcast transmission is important compared with the cost of a normal unicast transmission. Furthermore, for protocols such B-MAC and WiseMAC broadcast reception is also a highly consuming task since all nodes in the range of the transmitter remain awake until the reception of the frame. Including data into the wake up frames, as in ContikiMAC and BoxMAC, helps to minimize the cost of broadcast reception.

Then, energy and communication efficiency may be compromised when broadcast transmissions are done frequently because: i) Both transmitter and receivers must be awake during long periods of time, ii) Due to long wake-up signal transmissions *i.e.* preambles or micro-frames, contention is significantly increased and packets are very likely to collide. Every single node transmits its broadcast beacons; hence, as network density increases, high contention causes degradation of communication performance and increases energy consumption.

However, transmissions of broadcast frames may be helpful to discover new nodes and if they are not provided, a mechanism for neighborhood discovery must be implemented.

2. *Increasing complexity and useless tasks redundancy*:

   Since routing protocols rely on the MAC layer to perform neighborhood maintenance a lot of data is exchanged between layers. This fact results in a complex communication protocol stack that needs to be managed with very constrained resources.

   Also, in some scenarios, layers perform redundant tasks. For instance, synchronous medium access protocols, such as IEEE 802.15.4, provide a mechanism to maintain the topology by transmitting beacons at the beginning of the active period. But the routing layer, provides as well a mechanism for topology control. A proper protocol stack, where different layers cooperate with each other may help reducing this redundancy and complexity of communication tasks.

Some algorithms, such as Trickle [89], aim at minimizing the overhead introduced by broadcast announcements by adapting their transmission to the dynamics of the network. When a change or inconsistency is detected, the interval between consecutive announcements is reduced; conversely, it is increased as the topology reaches stability. Reactivity is traded off in order to reduce energy consumption and contention.

For neighborhood maintenance, by performing this task at the very low-level, Wake on Idle aims at optimizing energy consumption. Indeed, Wake on Idle reduces idle listening and relays on pure analog signaling; moreover, it provides synchronization and integrates low duty cycle medium access control. Following we give a detailed description of the mechanism; then, we discuss how it provides efficient medium access control.

4.2.2 Neighborhood maintenance protocol description

Wake on Idle performs neighborhood maintenance just on top of the physical layer. We demonstrate that, after passing through an association procedure, physical sensing of nodes can be done efficiently at the low-level and we can retrieve information that can be used by higher-level protocols. Then, the protocol provides the nodes the possibility to assess the presence of neighbors by sensing the channel energy level at well-defined instants.

Initially, two nodes wishing to associate and keep track of each other will go through a synchronization procedure. To do so, they agree on a common “seed” value. Such value is used to initialize a pseudo-random number generator whose function is common

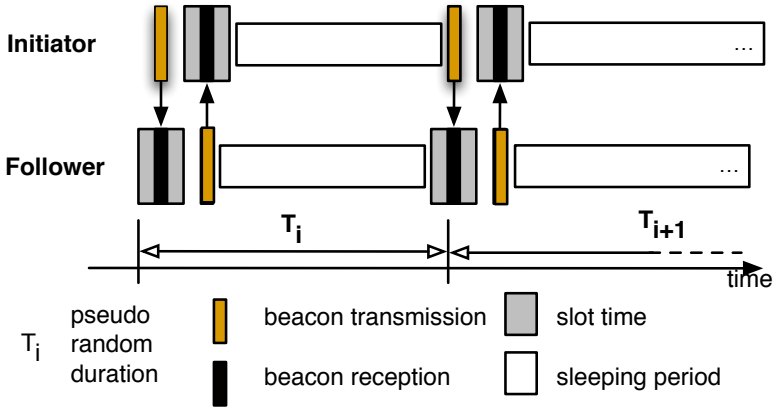


Figure 4.2: WoI Neighborhood Maintenance functionality: common synchronized pseudo-random schedules are used

to both nodes. Then, the nodes attempt to synchronize their wake-up periods, which are based on the pseudo-randomly generated values. We call the node initiating the association process the *initiator*, conversely the other node is the *follower*. After initializing the (common) pseudo-random sequence, the follower calibrates an estimation algorithm to account for clock drift and provide proper synchronization. After synchronization is reached, the nodes enter in a stable state where they follow repeatedly the process: compute the next common pseudo-random value, derive the next sleeping period duration from this value, schedule the next wake-up event and go to sleep.

Figure 4.2 shows the operation of a pair of associated nodes. Thanks to the synchronization procedure, they wake up almost simultaneously; then, the follower enters the listening mode, sensing the channel for an analog busy tone. Meanwhile, the initiator sends the busy tone intended for the follower. Then, in the next slot the symmetric operation occurs: The follower sends its busy tone and the initiator senses the channel looking for it. In this way both nodes can keep track of each other; and, since it is a symmetric operation, both nodes have knowledge about the presence of their respective associated node. Due to timing constraints, busy tones used for neighborhood maintenance are transmitted at the moment they are generated; hence, no CCA mechanism is implemented.

Besides busy tones are used to keep synchronization. Then, similar to existing synchronization approaches [56, 62] that use local synchronization, the busy tones can be used to estimate clock drift and compensate the error introduced by this value. Several instances of this procedure can be executed in parallel in the same node *i.e.* allowing to track several neighbors. According to initialization parameters, one node can be follower for some neighbors and initiator for the others. Hence, using Wake on Idle it is possible to form and maintain a (DO)DAG (Destination Oriented Direct Acyclic Graph) topology. The fact that a node can track several neighbors can be really advantageous when delivering packets. Unlike a cluster-tree topology formed by protocols such as IEEE 802.15.4, in Wake on Idle one node can have several possible packet forwarders. Some of the advantages of having these kinds of topologies are:

- It eases route diversity, giving the possibility to the routing algorithm to choose between different potential routes.

- It helps increasing the robustness of the network to node failures that causes routes to disappear.

- It allows load balancing, avoiding routing nodes or packet forwarders to empty their batteries due to very high traffic passing through them.

Following we give a description of the procedure followed by the initiator and the follower in order to reach the stable state (*i.e.* The synchronized state).

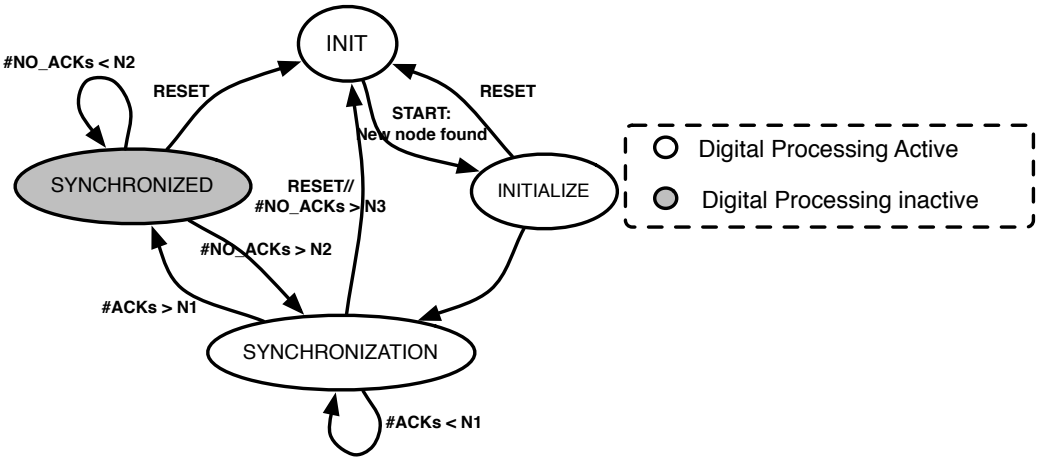
4.2.3 Initiator operation

In Wake on Idle, two neighboring nodes starting an association agree on a common seed, for example generating and transmitting a random number, or using a key exchange protocol. There is an extensive amount of work in the literature dealing with this issue [90, 91], and it is out of the scope of this work. Then, the initiator initializes its Pseudo-Random Number Generator (PRNG) and enters the *synchronization* state where it generates the first pseudo-random value and schedules its next wake-up time. When the time arrives, the initiator generates a modulated beacon containing information about its current position on the pseudo-random sequence and the identity of the follower. During synchronization, beacons must be digitally modulated to allow the follower calibrating its synchronization algorithm.

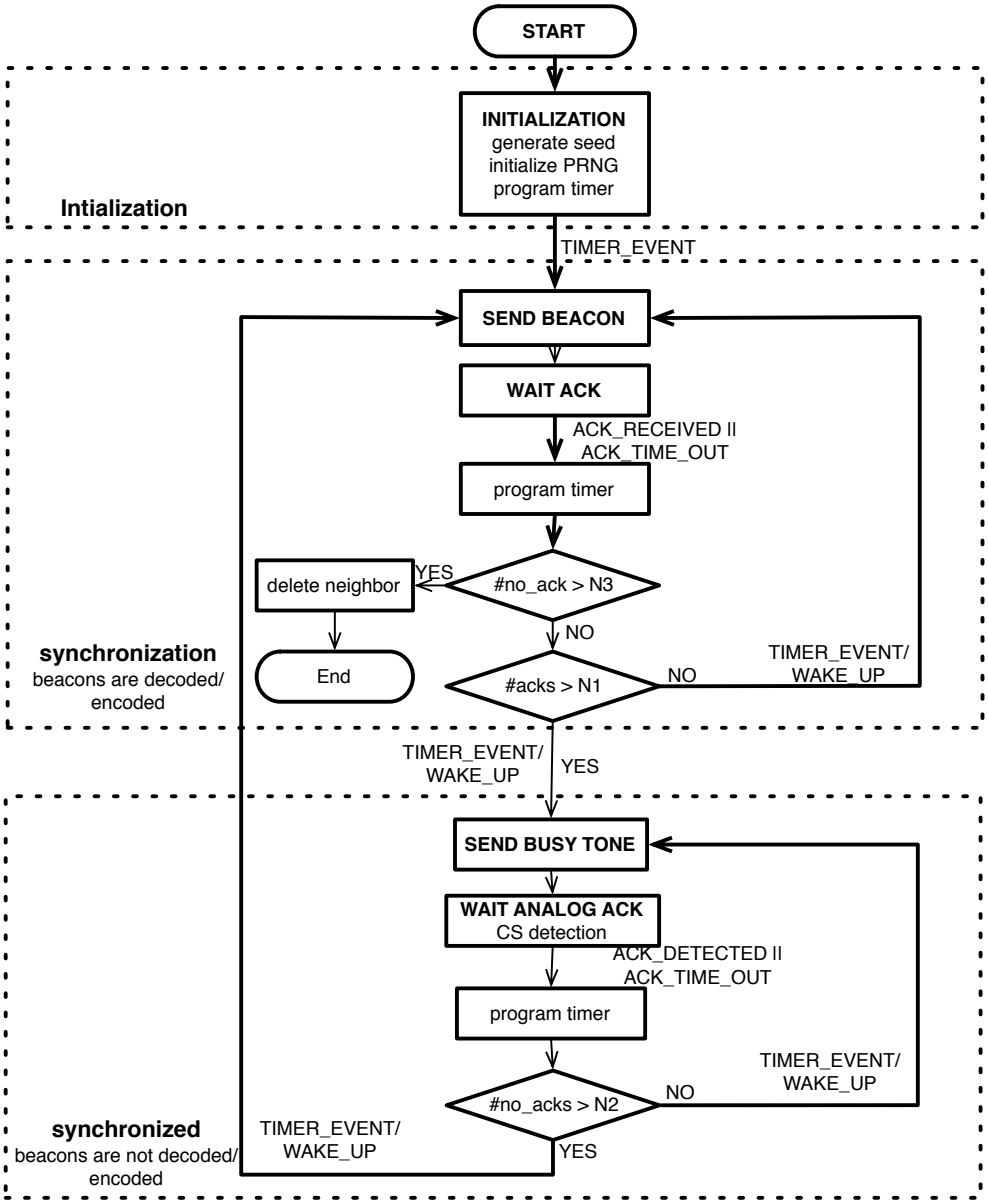
After the transmission of the beacon, the initiator senses the medium, looking for a beacon generated by the follower. As soon as the initiator detects a given number of beacons from the follower, it infers the follower has been able to reach synchronization; then, it enters the *synchronized* state where it disables the digital processing of its beacons and generates an analog busy tone whose minimum duration is determined by hardware characteristics *i.e.* the minimum time needed by the detection device to compute a valid RSSI value and detect a peak of power. Moreover, since it infers that the stable or synchronized state has been reached, it deactivates the digital part of its radio receiver and it does not decode beacons sent by the follower: it just detects the transmission by sensing the channel power level. Since several neighbors can be tracked simultaneously, the initiator keeps a list of timer events for all the nodes associated to it.

When several consecutive analog busy tone beacons are missed, the initiator infers that synchronization has been lost. The resynchronization procedure is left to the follower but, to allow this procedure, the initiator activates digital modulation and encoding of its beacons. If synchronization is not reached after a given time-out, the initiator decides that the follower is not available any more and discards the association.

The operation of the initiator can be illustrated (and implemented) using a Finite State Machine (Figure 4.3a) whose functionality is represented by the flow chart in Figure 4.3b. In the figures by ACK we refer to the beacon sent by the follower.



(a) Initiator FSM



(b) Initiator Flow Chart

Figure 4.3: Initiator operation

4.2.4 Follower operation

Once the seed value is agreed upon, the follower goes through successive states, described below, to synchronize with the initiator. To account for synchronization in real hardware it is important to account for hardware imperfections, specifically clock skew and drift. The details of the procedure to compensate these imperfections and chieve correct synchronization are described in section 4.3. After some time, the follower reaches a stable or synchronized state. The process described here is repeated for each initiator node soliciting association.

- *Characterization:* The follower collects timing information from received beacons. This information forms a set of points that are used to estimate clock skew with respect to the initiator.

- *Synchronization:* The clock skew value computed before is then used to estimate the time the initiator will send the next beacon. After this time the follower wakes-up and saves the beacon arrival time. The estimation error of the beacon reception is then used to add a correction factor of the clock skew; this correction factor accounts for errors introduced by clock drift or other delays.

  When the estimation error becomes smaller than an adjustable threshold, the node assumes that it is synchronized with the transmitter and enters the synchronized state. Then, it starts sending its (ACK) beacon after the reception of the beacon transmitted by the initiator.

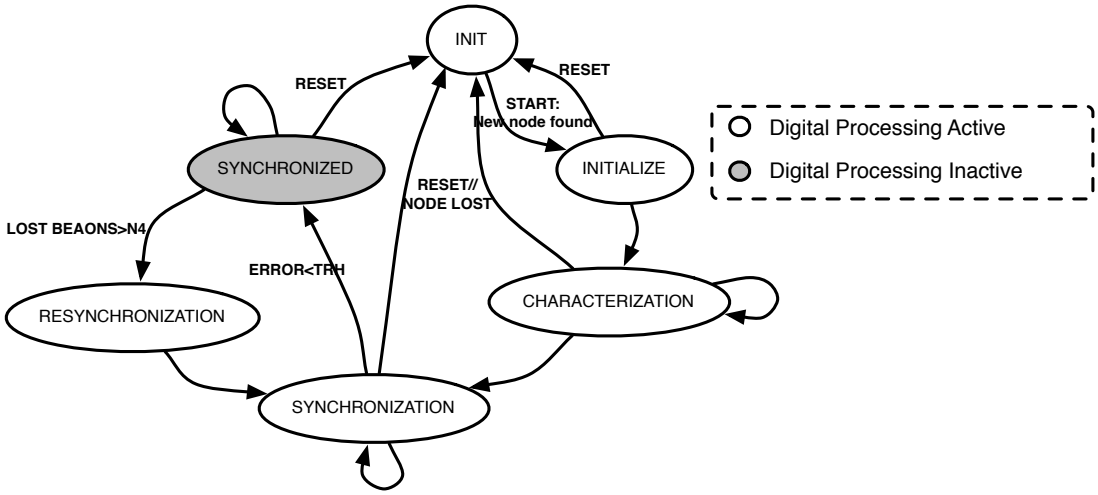
- *Synchronized:* This state is reached when the follower is synchronized *i.e.* estimation error is small, and it has reached the stable state. Based on the current estimation error and skew value, the follower estimates the arrival time of the next beacon, schedules this event and enters sleep state. Digital processing of frames is deactivated.

- *Recovery:* When synchronized, if the number of consecutive missed beacons is larger than a custom value (threshold), the follower enters the recovery state. It then attempts to resynchronize with the initiator. In this state, both nodes activate their digital processing of packets; the beacon payload is used to reinitialize the schedule, restart the PRNG and resynchronize. The cost of a resynchronization depends on protocol parameters.

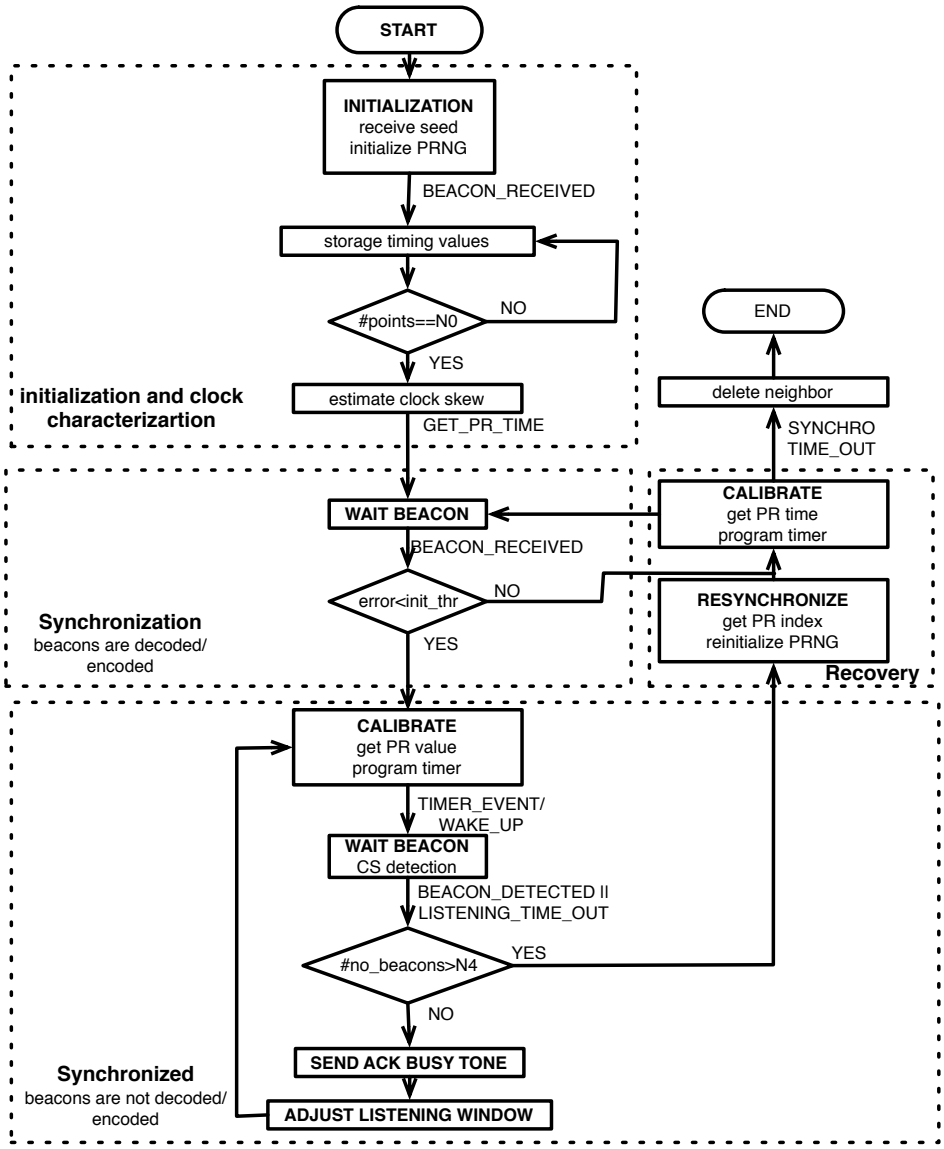
The follower operation can be as well implemented by a Finite State Machine. Since it implements synchronization, it is more complex that the initiator's FSM. An overview of the follower FSM is illustrated in Figure 4.4a and its operation is illustrated by the data flow shown in Figure 4.4b.

4.3 Implementation details

In order to have the protocol working in a real platform there are some aspects that must be taken into account. In this section we describe in detail our implementation on two different hardware platforms and we discuss some important points and challenges.



(a) Follower FSM



(b) Follower Flow Chart

Figure 4.4: Follower operation

4.3.1 Implementation overview

We have implemented our protocol on two platforms, the eZ430-RF2500 [6] and the WSN430 [5]. The characteristics of these platforms are summarized in Table 4.1. The platforms differ mainly on two aspects: i) the frequency bands in which they operate, the eZ430 operating in the same band as WiFi, (ii) their oscillator, the WSN430 being much more precise than the eZ430. Also the eZ430 has a microcontroller much more limited in terms of available RAM and flash memory than the one available by the WSN430; furthermore, the WSN430 has an external flash memory and a ROM memory.

We have based the implementation on the drivers code made available by SensLAB [92].

Table 4.1: Platform characteristics

	WSN430	**eZ430-RF2500**
Processor	MSP430F161	MSP430F2274
	8 MHz – 16 bits 48 KB FLASH 10 KB RAM	16 MHz – 16 bits 32 KB FLASH 1 KB RAM
Transceiver	CC1101	CC2500
	316-915 MHz	2.4 GHz
	up to 500 kb/s Wake on Radio	
Oscillator	32 kHz external	12 kHz internal VLO
Actuators	3 LEDs	2 LEDs
Sensors	1 push button Temperature and voltage sensors	

Since we did not develop custom hardware and the available platforms are not able to generate analog busy tones, we adapted the implementation to make it compatible with the platforms. Hence, nodes always send modulated beacons, but most of the time they are only detected and not processed *i.e.* no data is taken from the radio buffer.

We differentiate between the *reception* of a frame and the *detection* of a transmission. Detection is the action of sensing the medium to report if a certain power level of signal is present. This is present in most radios under the name carrier sensing (CS). The Carrier Sense pin is asserted when a certain configurable received signal strength indicator (RSSI) threshold is reached or exceeded. Then, detection does not require data decoding nor processing of frames. On the other hand, reception is the process during which the radio synchronizes itself with an ongoing transmission, decodes the frame and notifies the processor to analyze received data. The reception process starts after the transmission of a synchronization word is found; in this case, the packet synchronization (SYNC) signal generated by the radio is asserted.

The radio changes its operation between "detection" and "reception" according to its state in the synchronization FSM. Then, when the nodes reach the synchronized state, where no digital process of data is required, the radio enters the detection mode.

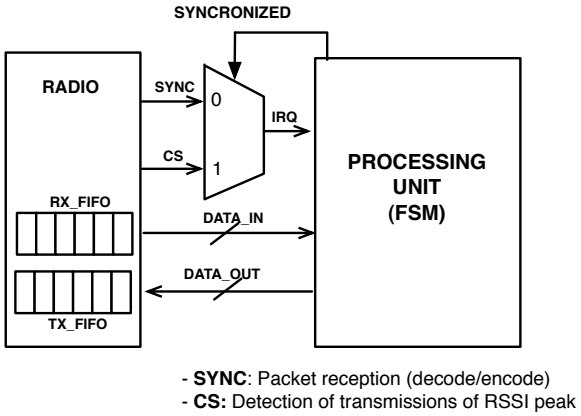


Figure 4.5: Reception vs detection of transmissions: Switching between reception of packets and detection of transmissions based on the synchronization state

Otherwise, it is in reception mode since, to reach synchronization, nodes must use the beacon data payload. Figure 4.5 shows how the switching between these two operation modes can be done. Once the stable state is reached, we expect that the radio remains in the detection mode most of the time. Since we are able to reach a fairly precise synchronization we expect the nodes to leave the synchronized state very rarely.

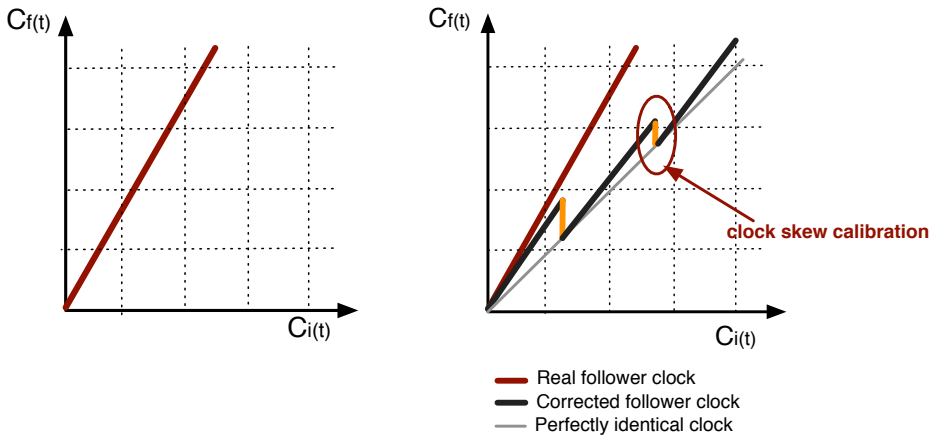
4.3.2 Clocks synchronization procedure

Perfect oscillators do not exist; as explained in 3.1.1, to reach synchronization we must take into account clock imperfections: *skew* and *drift*. We have already enumerated some mechanisms used to estimate and correct errors introduced by these imperfections; any of these methods can be integrated with the Wake on Idle neighborhood maintenance protocol. The complexity of the synchronization mechanism is given by the requirements in terms of accuracy and characteristics of the oscillators being used.

a) Clock skew calculation

The relationship between two clocks can be approximated by a line. Figure 4.6a shows an example of the clock value of a node acting as a follower (y-axis) vs the clock value of an initiator (x-axis). In the example, the follower's oscillator is running faster than the initiator's one. The slope of the line determines the relative speed or the clock skew between the two oscillators. By estimating the clock skew, the follower can interpolate its local time to the initiator time.

The line slope can be computed applying a least square linear regression, as proposed in TPSN [58] or a more accurate procedure, as the one proposed in Tiny-Synch [56], can be implemented. As shown by Duda et al. [57], the problem of the linear regression method is that, due to indeterministic delays introduced by the radio chip, it may result in a negative skew value and synchronization cannot be reached. The number



(a) Example of clock skew: follower is faster than initiator

(b) Correcting follower clock to follow the initiator clock

Figure 4.6: Clock skew and drift correction

of points used for skew calculation determines the accuracy of the estimation [60] and the time and energy spent by the follower during this phase. Since we do not use any CSMA mechanism before the transmission of (very short) beacons and we use low-level timestamps we can ignore the indeterministic delays and the problem of negative slopes should not happen. Also, delays introduced by hardware while decoding the frames can be ignored since we only observe the RSSI level in the channel.

$$a = \frac{n * \sum_{i=1}^{n} x_i * y_i - (\sum_{i=1}^{n} x_i * \sum_{i=1}^{n} y_i)}{n * \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \tag{4.1}$$

For practical purposes we have implemented a fixed-point version of Equation 4.1.

b) Clock drift correction

As time passes, there are delays added by clock drift, processing, external and environmental factors. If unaccounted for, these delays will add up over time causing loss of synchronization. A continuous adaptation process is necessary; for instance, every time a beacon arrives a corrective factor can be computed and added to previous estimations.

In order to compute the correction factor we have implemented an IIR filter whose transfer function is given by Equation 4.2. The filter coefficients can be chosen according to hardware characteristics; we have empirically chosen these values during experiments with the platforms. A more flexible implementation permits to change the filter order according to the desired complexity and precision.

$$H(x) = \frac{b_2 z^2}{z^2 + a_1 z + a_0} \tag{4.2}$$

Fixed-point arithmetic has been used as well for filter implementation.

Figure 4.7 plots the estimation error, *i.e.* the time passed between the follower wake-up instant and the arrival of the beacon transmitted by the initiator. These results were obtained with the ez430 nodes. During these experiments we have used the very low-power oscillator (VLO) whose nominal frequency is 12 kHz. Since this VLO is a cheap and inaccurate oscillator, the frequency varies significantly with changes in temperature and supplied voltage; also, clock drift effects can be easily observed. It can be seen that as time passes these error values become larger if no clock drift correction mechanism is introduced. The WSN430 nodes have a more accurate oscillator and variations are not so important; still, clock drift must be corrected to reach good synchronization.

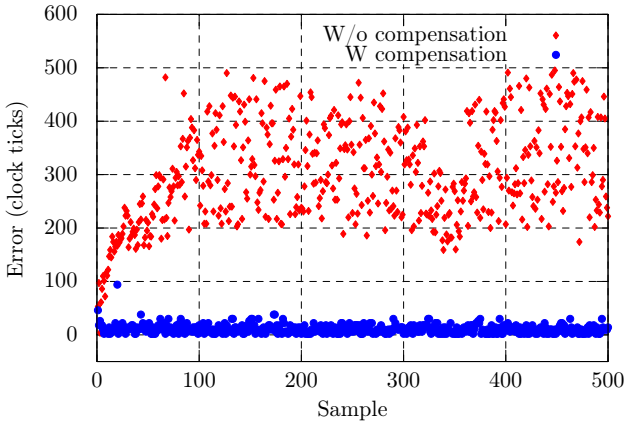
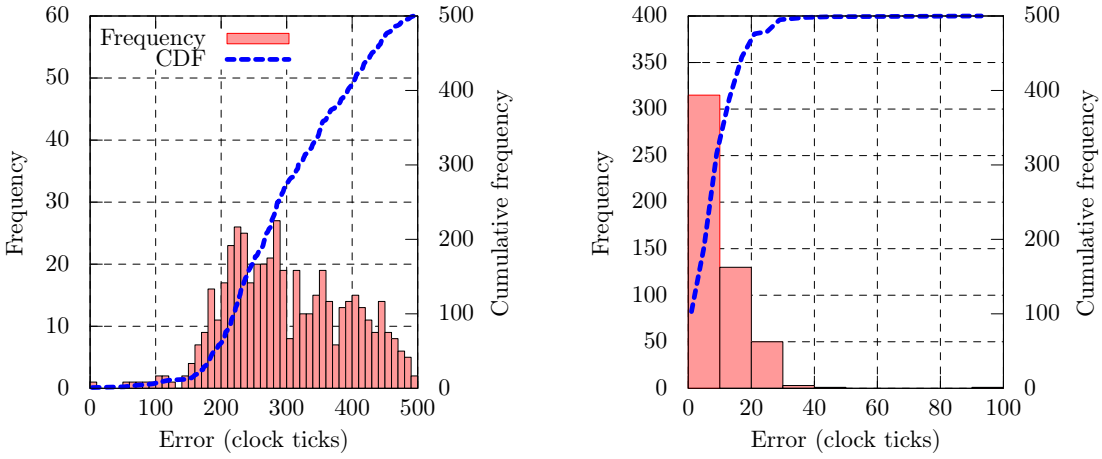


Figure 4.7: Wake-up estimation error evolution using the eZ430



(a) Without calibration

(b) With calibration

Figure 4.8: Wake-up estimation error distribution using the eZ430

To better observe the importance of clock drift correction we plot the distribution and the density function of these estimation errors. Figure 4.8a plots the distribution of the wake-up estimation error (in clock-ticks) when clock skew is estimated but clock drift is not corrected; errors introduced by clock drift and other factors become very large and synchronization can be lost frequently. Since resynchronization may be expensive, if clock drift is not corrected, energy efficiency may be compromised.

Figure 4.8b plots the distribution of the wake-up estimation error when the calibration mechanism is applied. It can be seen that the error is kept stable and close to zero. However to account for small variations a wake-up in advance margin and a listening window must be added. For the eZ430 we can choose a margin of 100 clock-ticks (8.3ms) for an inter-beacon time of 5 seconds. For the WSN430, having a more accurate oscillator, we can choose a much smaller margin. Proper margin values and the influence of this parameter are discussed in 4.3.3.

4.3.3 Protocol parameters

a) Timing parameters

To cope with hardware imperfections and effects of environmental conditions we need to define a set of timing parameters. These parameters make possible to have a functional implementation that is able to adapt to the given conditions. Figure 4.9 gives an illustration of these parameters.

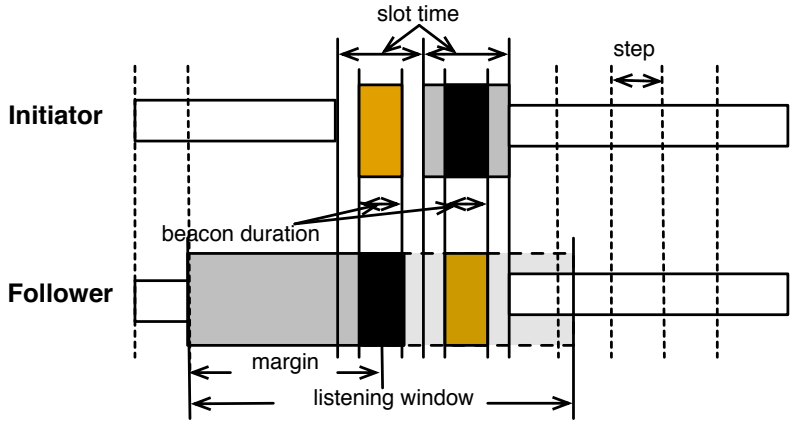


Figure 4.9: WoI timing parameters

The average time between two beacons may greatly affect the protocol performance. The longer a node sleeps, the larger is the error introduced by clock drift; this could lead to loss of synchronization. In this case, to avoid losing synchronization we could apply two possible solutions:

- Adapt the inter-beacon time to keep the clock drift small; this triggers more frequent radio activity and increments power consumption. However, system reactivity, *i.e.* time to detect a node failure or absence, is improved.

- Use a larger *wake-up in advance margin* value when listening for beacons. This measure results in longer idle-listening periods, but also provides robustness since the possibility to miss a beacon is reduced as a well as the probability to lose synchronization.

There is clearly a trade-off when tuning these parameters; their choice may depend on the application requirements and characteristics, hardware components and operating conditions. The timing parameters, shown in Figure 4.9, are listed below.

- *Step:* The *step* is the number of clock ticks per unit of time. Then, the duration of the step is hardware dependent and it depends on the frequency of the oscillator that is used. Pseudo-randomly generated values are given in units of time; then, these values are multiplied by the step. In this way we obtain the node wake-up time in clock ticks and we can directly program the wake-up timer. The objective of introducing this parameter is the possibility to adjust the average time between beacons or busy tones: the bigger the step, the longer the sleep time. For this parameter we use powers of 2 to simplify the implementation.

  A large step size results in a very low duty cycle, while a small step size results in better reactivity. Also, a small value for this parameter produces high contention and, as we do not use a clear channel assessment mechanism, beacons are more likely to collide.

- *Beacon duration:* This parameter is the length of the transmitted busy tone beacon. It depends on the radio hardware response time. Also, when it is a modulated beacon, the beacon duration is given by the beacon payload.

- *Slot time:* The slot time is the interval in which the beacon is sent or received. It accounts for processing delays in the radio, turnaround time of the transceiver, and other radio related timing parameters. Therefore, this parameter must be chosen according to the hardware being used for implementation.

- *Wake-up margin:* This parameter is the wake-up in advance margin added to cope with delays and clock errors. The *listening window* size is given by $2 \times \text{MARGIN}$ and this value includes the slot time, as shown in the figure; then, it is necessary that $\text{MARGIN} \geq \frac{\text{slot\_time}}{2}$.

  Increasing the listening window increases the duty-cycle, but the probability to miss beacons decreases. However, a wide listening window will affect synchronization since the receiver may detect beacons that are not intended to it and will base its clock drift estimation on this false beacon. We evaluate the impact of this trade-off on resynchronization occurrences in Section 4.4.

b) Decision parameters

These are the parameters used to take decisions such as state changing or deleting an association. These parameters are shown in the data flow diagrams on figures 4.3b and 4.4b, and they are listed bellow.

- *N0*: Number of collected points used during the linear regression for skew estimation.
- *N1*: Number of consecutive (ACK) beacons transmitted by the follower and detected by the initiator to infer the nodes are synchronized and entered the stable state.
- *N2*: Number of consecutive (ACK) beacons lost by the initiator to infer the follower has lost synchronization and it should start transmitting modulated beacons.
- *N3*: Number of consecutive (ACK) beacons lost by the initiator to infer the follower is not available any more and delete the association.
- *N4*: Number of consecutive beacons lost by the follower to call the resynchronization routine and start decoding beacons.

c) Adaptive listening window

This mechanism is added to speed-up synchronization. When the follower receives the beacons transmitted by the respective initiator, it attempts to synchronize and as soon as the wake-up estimation error becomes smaller than a threshold the follower introduces a listening window and enters the synchronized state. As the estimation error is reduced the listening window is reduced as well.

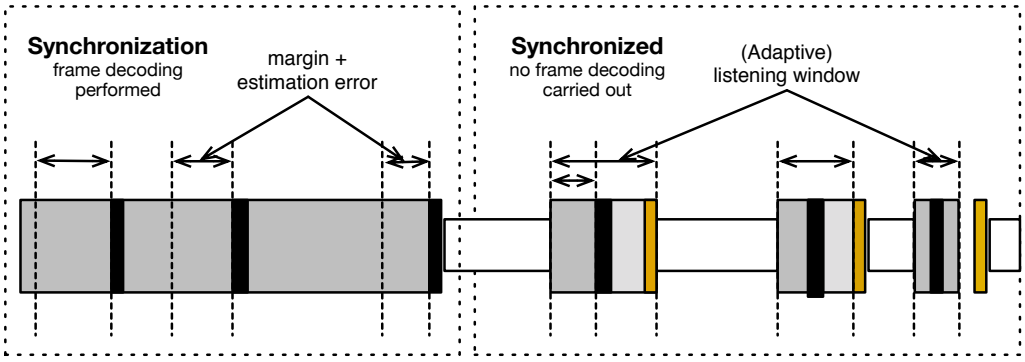


Figure 4.10: Adaptive listening window in WoI

The threshold value used to enter the synchronized state determines how fast this state is reached and it must be chosen according to hardware and implementation characteristics, such as oscillator accuracy and clock skew calculation accuracy. The listening window is then reduced until a minimum value *i.e.* twice the margin. Figure 4.10 illustrates how the adaptive listening window works.

4.3.4 Correct beacon detection

Detection of beacons is done by sensing the medium at proper instants; frames are not decoded and it is possible to confuse external transmissions or interference with

a beacon. To guarantee the proper functionality of the protocol, even in presence of interference, we must be able to correctly distinguish beacon transmissions from external interference. With this purpose we added two functionalities (see Figure 4.11):

- *A relative carrier-sensing threshold*, so that the receiver can detect an RSSI peak even if there is already a signal on the medium; this functionality is already present in the radio chip. The proper CCA threshold must be chosen according to the scenario and environmental characteristics. Also, by keeping a history of the RSSI value of the associated node we can distinguish beacons from different nodes in the same transmission range.
- *A time counter to determine the duration of the signal*, in this way the receiver, knowing the duration of a beacon, is able to determine if the RSSI peak is generated by an external node.

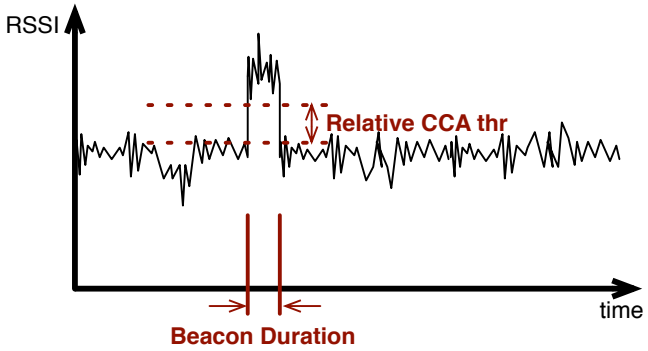


Figure 4.11: Beacon detection in WoI: using relative CCA threshold plus peak duration

The last mechanisms help mostly to detect external transmissions or interference and separate them from beacon transmissions. However, concurrent transmissions of beacons may happen. These collisions have the following consequences:

- A node listening for a beacon from a missing associated node would mistakenly detect its presence. Thanks to the introduction of randomness, such an event should repeat with an extremely low probability; although the consequences are very limited, only potentially deferring the detection of a missing node for a few periods.
- Two beacons collide, which in theory leads to no negative consequences. Both beaconing nodes get detected by their respective associated node. However, if two beacons "almost" collide, due to necessary guard times, this might have an impact on error estimation, biasing synchronization. This is what we call a *false positive*. In 4.4.1 we discuss the consequences of false positives.

An accurate clock skew estimation and clock drift compensation, that leads to a small wake up in advance margin value reduces the probability of overhearing the

wrong beacon. Also, implementing a good PRNG that yields to few collisions reduces the occurrence of the scenarios described above. In our implementation we have chosen the multiply-with-carry PRNG, presented by Marsaglia [93], it was called the mother of PRNGs. The recursive operation carried to compute the pseudo-random value is given by Equation 4.4.

$$x_i = (a \times x_{i-1} + c_{i-1}) mod(b) \tag{4.3}$$

Where c is the carry value and is given by:

$$c_i = \left\lfloor \frac{a \times x_{i-1} + c_{i-1}}{b} \right\rfloor \tag{4.4}$$

The constant b is given by the number of bits of the generated value, *i.e.* $b = 2^{\#\text{GeneratedBits}}$, and a is chosen such that $a \times b - 1$ and $\frac{a \times b - 1}{2}$ are prime.

The value chosen for a determines the period of the pseudo-random sequence. According to tests presented in the literature [94], the multiply-with-carry generator exhibits desirable randomness characteristics and generates sequences with very long periods. Moreover, it is simple to implement since it makes use of simple arithmetic operations.

4.3.5 Beacon frame format

Given that we do not have custom hardware, beacons that are transmitted are always modulated. However, most of the time, the data payload is not used by the receiver.

The beacon frame format is illustrated in Figure 4.12. The current implementation needs 7 bytes, excluding the synchronization word and the preamble used by the radio chip to detect a transmission and decode the information. The CC1101 and CC2500 radios use 8 bytes for both synchronization and preamble.

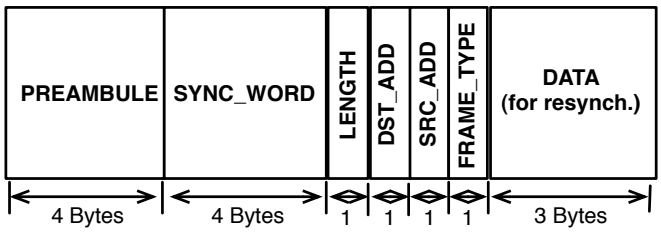


Figure 4.12: WoI Beacon frame format

To account for security, beacon data may be encrypted using the already exchanged seed value as a key.

When receiving a beacon, useless frame decoding consumes additional energy. Then, by manipulating the hardware filtering functionality provided by the radio chips available in the platforms, we prevent the radio to decode frames when it is in detecting mode. Beacons are still correctly detected, *i.e.* we use the carrier-sense signal, but

they are not processed by the radio. Then, by taking advantage of existing hardware functionality we are able to implement efficient "analog" beacon detection.

4.3.6 Neighbor discovery for experimentations

In order to form a topology, nodes need to discover each other. After a pair of nodes see each other they can decide, based on a set of metrics, if the association can be useful, i.e. the node can be used as a packet forwarder, and if it is worth to keep track of the node. Neighborhood discovery when using duty cycling techniques is a complicated task. A the end of this chapter we discuss the neighborhood discovery issue and we open some questions for future directions aiming at solving this problem.

For experiment purposes, at the start time, all nodes enter in a scanning stage. During this stage, nodes transmit beacons on a periodic basics and listen for beacons from neighbors. Based on some initial parameters and metrics (*i.e.* the RSSI value of beacons that were received) associations are formed between the nodes. During this scanning process, nodes are active 100% of the time. This solution is way too inefficient since it drains too much power during the initialization. However, it helps us to bootstrap the network and evaluate Wake on Idle protocol during the stable state.

4.4 Neighborhood maintenance protocol evaluation

4.4.1 Theoretical and simulations analysis

To evaluate the neighborhood maintenance protocol we start by giving an analytical evaluation of the probability of having a beacon collision or false positive. Validation of this analysis is done by performing some timing simulations.

The results presented in this section are done for the WSN430 motes, which use a 32 kHz oscillator; for the eZ430 results are similar taking into account that it has a slower oscillator. Then, step times are longer for the eZ430 platform. For instance, a step of 1 has a duration of $31.25\,\mu s$ on the WSN430 and $83.33\,\mu s$ on the eZ430. For practical reasons and to keep the figures clear, we have only plotted the results for step values of 1 and 4; longer values will have a negligible false positive probability since the interval between two beacons is long. In fact, for false positive analysis, a step of 1 is the most critical case. In a real implementation, the step can have much longer step values according to the characteristics of the oscillator being used; for instance, in the WSN430 nodes we have reached step values of the order of 2^9.

a) False positive probability

Using pseudo-random sequences to schedule transmissions guarantees some nice properties due to low correlation between different sequences. Hence, depending on the number of nodes in the transmission range, we expect having two overlapping transmissions with a very low probability. Timing parameters such as the step and the wake up in advance margin influence the probability of having beacon collisions and false positives.

The introduction of the wake up in advance margin, needed for a functional implementation of the protocol, greatly affects the probability of having false positives. Since

we do not decode the frames that are used for neighbor tracking, collisions and false positives must be considered during the evaluation of the protocol. Also the consequences of these events are studied.

Let's derive the probability of having one or more consecutive false positives. Beacon transmissions on each node are independent events; the probability that a node j detects a false positive is given by:

$$P_{fp}^{j} = 2 \times \sum_{i=0}^{n} P(t_{wu}^{j} \leq t_{tx}^{i} \leq t_{tx}^{j}) \times P(i \neq j) \tag{4.5}$$

Where t_{wu}^{j} is the instant on which node j wakes up and t_{tx}^{j} is the instant on which the beacon intended for node j is sent. The factor 2 is added because we consider n pairs of associated nodes, and both initiators and followers send beacons. Since $t_{wu}^{j} = t_{tx}^{j} - \text{MARGIN}$:

$$P(t_{wu}^{j} \leq t_{tx}^{i} \leq t_{tx}^{j}) = P(0 \leq t_{tx}^{i} \leq \text{MARGIN}) \tag{4.6}$$

For a PRNG generating k bits values, the pseudo-random generated values are uniformly distributed in the interval $[0, b-1]$, where $b = 2^k$. Hence:

$$P(tx = t) = \frac{1}{b} \tag{4.7}$$

and

$$P(0 \leq t_{tx}^{i} \leq \text{MARGIN}) = \frac{\text{MARGIN}}{b} \tag{4.8}$$

The second term in Equation 4.5 refers to the probability that the beacon is not targeted at node j and is given by:

$$P(i \neq j) = 1 - \frac{1}{n} \tag{4.9}$$

Where n is the number of pairs of associated nodes that are in the range of node j.

Finally, putting the equations 4.8 and 4.9 together we get:

$$P_{fp}^{j} = 2 \times n \times (1 - \frac{1}{n}) \times \frac{\text{MARGIN}}{b} \tag{4.10}$$

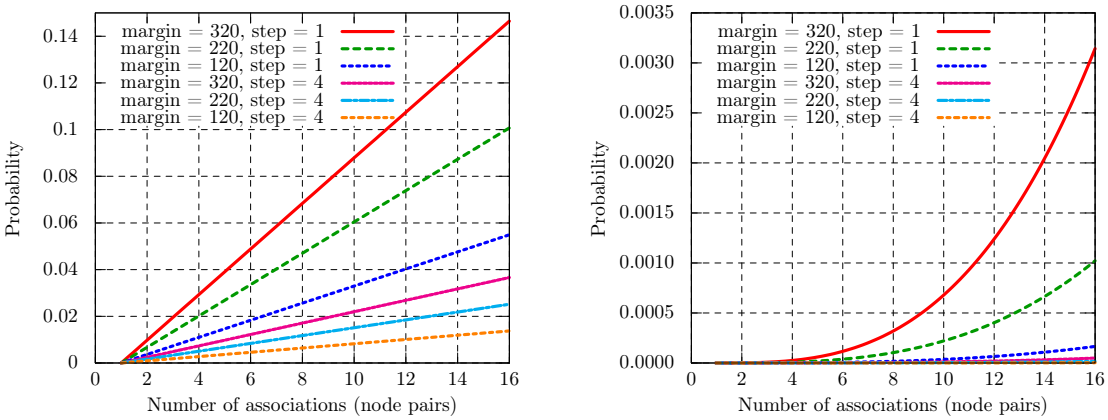
The probability value given by Equation 4.10 is for a step size value of 1 clock tick. As this parameter is increased the probability decreases:

$$P_{fp}^{j} = 2 \times (n-1) \times \frac{\text{MARGIN}}{b \times \text{STEP}} \tag{4.11}$$

When implementing the PRNG on a 16-bits microcontroller, such as the MSP430 available in our platforms, $k = 16$ is a reasonable value. Hence, with $b = 2^{16}$ generated values fall in the range $[0, 65535 \times \text{step}]$ clock ticks.

The false positive probability for a step value of 1 and 4 and different margin values versus the number of neighbor associations is plotted in Figure 4.13a.

A false positive happens when a beacon intended for another follower falls into the listening period of a node. Then, the node tries to synchronize with the wrong



(a) Theoretical false positive probability

(b) Probability of having three consecutive false positives

Figure 4.13: Probability analysis results

beacon skewing the estimation away from its stable point. We have seen in experiments and simulations that, with a suitable margin value, the estimation algorithm can tolerate a few consecutive false positives. So we can have one false positive without negative consequences but we must avoid having consecutive false positives. For instance, the probability of having k consecutive false positives can be computed based on Equation 4.11. Since the periods are pseudo-random values, the beacon transmission instants are independent events and this probability is given by:

$$P^j_{kfp} = \prod_k P^j_{fp} = (P^j_{fp})^k \tag{4.12}$$

Figure 4.13b plots the probability of having three consecutive false positives for different parameter values. It can be observed that this probability value is very low; so, false positives may cause loss of synchronization with a very low probability.

b) Timing simulations

To validate the analytical results discussed before we have carried out some timing simulations in Matlab. In these simulations we do not take into account oscillator behavior, *i.e.* clock drift or skew, nor hardware delays such as packet decoding/encoding and medium access delays.

We have simulated the timing behavior of the neighborhood maintenance protocol when we have up to 16 associations in the same transmission range; the topology, *i.e.* how associations are established, does not have any impact in the results. We have explored different step and margin values. For each experiment that we have conducted we have simulated 60 minutes of communication. For the case of WSN430, with a 16-bit timer counter and a 32 kHz clock, generating 16-bit pseudo-random values, a step of 1 means an average inter-beacon interval of around 1s, a step of 2 means 2s and so on.

- *False positives:*

  The rate between the number of false positives that occurred and the total number of beacons that were transmitted versus the number of associations is plotted in Figure 4.14. Choosing the same margin and step values used in Figure 4.13a, we get similar probability values in both cases. We observe a linear relationship between the false positive probability and the number of associations.

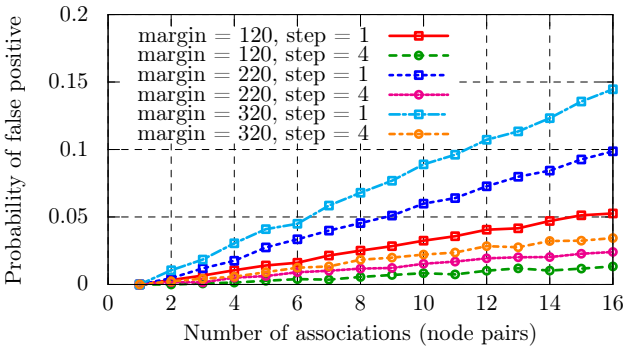


Figure 4.14: Rate of false positives in simulations

- *Occurrence of resynchronizations:*

  We observe the number of resynchronizations performed per node during the timing simulations. These values are plotted in Figure 4.15. Since there is no clock drift or other delays, resynchronizations are due only to biases introduced by false positives.

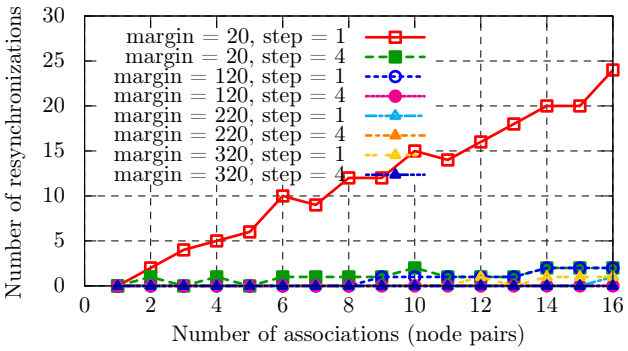


Figure 4.15: Number of resynchronizations in one hour

We observe that, for reasonable margin values, the number of resynchronizations is kept low. For example, no resynchronizations are observed for a MARGIN ≥ 120 and a step $= 4$. This value explodes for a small margin value; this is due to the

shifts caused by false positives: The node wakes up too late or too early, and due to the small margin it is not able to detect the beacon.

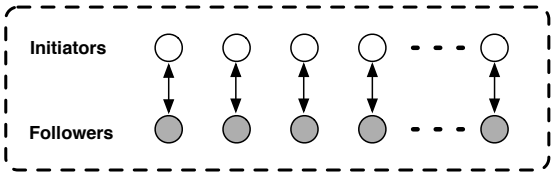
Very long margins may cause resynchronizations as well, since the probability of having a false positive is increased. Hence, the proper margin value must be chosen taking into account the oscillator characteristics while minimizing the number of resynchronizations.

4.4.2 Experimental evaluation

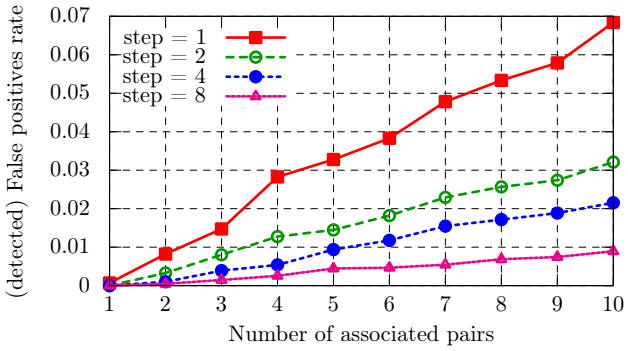
Finally, we have evaluated our implementation. With this purpose we start by evaluating the detection mechanism explained in 4.3.4, *i.e.* the capacity to detect a false positive. Then, we evaluate and discuss the cost of our protocol: Energy consumption and storage.

a) False positives detection

In order to test the capacity of our implementation to detect false positives we have set up several pairs of associated nodes in the same transmission range, *i.e.* we use the highest possible transmission power. The evaluated scenario is illustrated in Figure 4.16a.



(a) Evaluated scenario



(b) (detected) false positives rate

Figure 4.16: Detection of false positives in a real test-bed

In these experiments we have chosen a wake-up in advance margin of 120 clock ticks and we have varied the step value from 1 to 8. We have tested up to 10 associations. Each experiment has lasted 30 minutes, and they have been repeated 5 times. Figure 4.16b plots the rate between detected false positives and the total of detected beacons. We observe that the average rate of detected false positives at the followers,

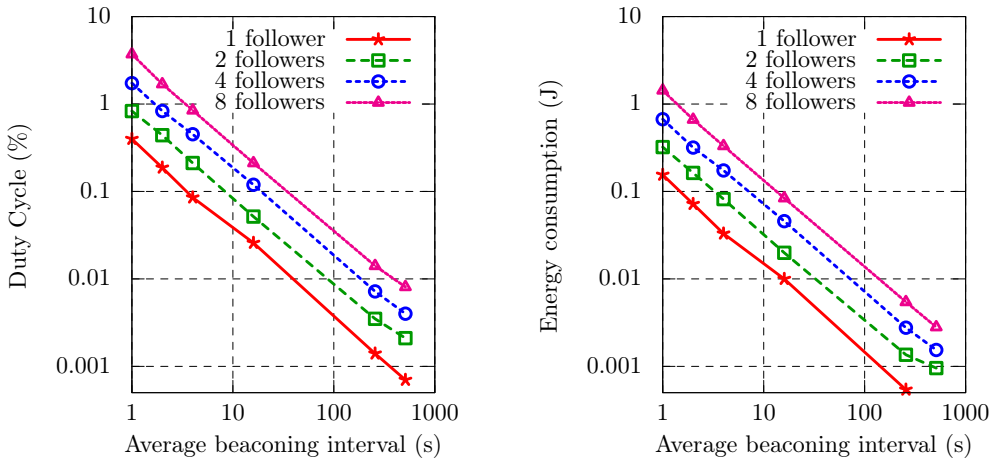
for the chosen margin value is very close to the theoretical and simulation results. The detection mechanism is able to correctly detect false positives generated by other initiators using relative CCA threshold, beacon duration and RSSI value related to the nodes of interest. However, in presence of very high interference, busy tones may not be detected and synchronization may be lost.

The number of detected false positives and the number of lost beacons can also be used as a metric to determine the quality of the association and how reliable it is to send packets.

b) Cost of neighborhood maintenance

- *Energy consumption and Duty Cycle estimation*:

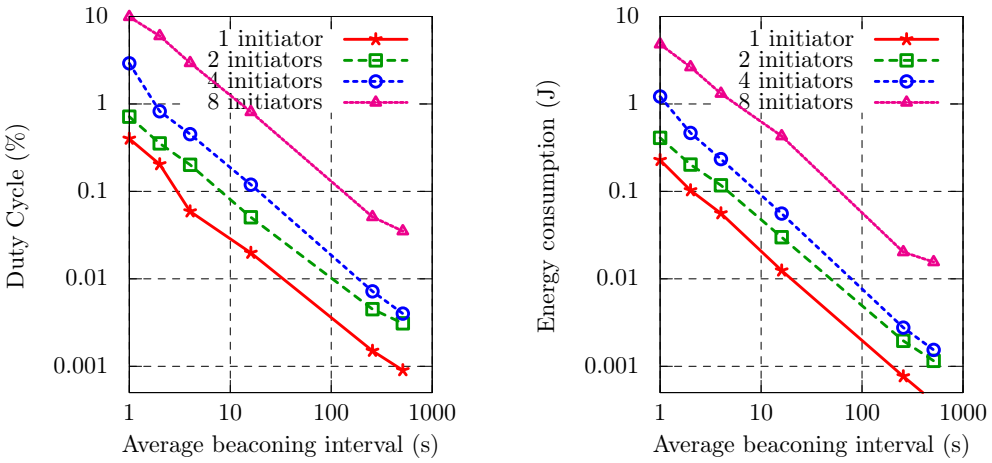
  We aim at estimating the energy consumed while neighborhood maintenance is performed. With this purpose, we have estimated the duty cycle, *i.e.* inactive time, reception time and transmission time, of initiators and followers when they are associated to a variable number of neighbors. With the purpose of estimating the duty cycle, each node logs the time the radio remains on each state; the duty cycle is then the fraction of time the radio remains active. We have also estimated the energy consumed by the radio based on the time the radio spends in each state: transmission, reception and sleeping. For these experiments we have used a margin value of 120 clock ticks, *i.e.* around 4*m*s.



(a) Duty Cycle (%) **(b)** Energy (J)

Figure 4.17: Initiator experimentation results

Figures 4.17 and 4.18 present the duty cycle (in %) and energy consumption (in J) of an initiator associated to a variable number of followers and a follower associated to a variable number of initiators, respectively. We have varied the number of associated nodes from 1 to 8 and the step from 1 to 2^9, which correspond to an average interval between busy tones of 1 s and 512 s respectively. Each



(a) Duty Cycle (%) **(b)** Energy (J)

Figure 4.18: Follower experimentation results

experiment was run during one hour. Notice the extremely low duty cycles that can be reached *i.e* for a step of 2^9 and only one associated node, the follower reaches a duty cycle of less than 0.001% and the initiator reaches a smaller value since it does not have to synchronize and keep wake-up in advance margins. Then, Wake on Idle is able to maintain nodes synchronized and tracking each other with a very low overhead in terms of energy consumption.

We also note that the duty cycle and energy consumption are linearly proportional to the number of associated nodes; also, the probability of losing synchronization increases with the number of associated nodes since contention is increased. Synchronization can also be penalized due to possible delays added when scheduling (and serving) events.

For a multi-hop topology, a node can at be the same time follower for some nodes (*i.e.* parents) and initiator for other nodes (*i.e.* children). When evaluating such topologies we have found out that the duty cycle and energy consumption can be estimated by summing up the values plotted in the former figures. For instance, when using a step of 9, if a node follows 2 initiators and at the same time has 2 followers, its duty cycle can be estimated from the figures above as: $0.0031 + 0.0021 = 0.0052\%$.

- *Resynchronization cost*:

  Although our drift correction mechanism is precise enough to keep synchronization, loss of synchronization may arrive. We have run several experiments during long times, *i.e.* several hours, and we observed a negligible number of resynchronizations. The probability of losing synchronization due to false positives has been studied above and it increases as the network density increases. The resynchronization mechanism requires turning on the radio during longer periods until the desired beacon is found. A soon as the beacon arrives, the adaptive window

mechanism will start reducing the listening time until the steady state is reached. During the experiments we have seen that the follower needs at most 3 wake-up intervals to resynchronize and enter the steady state. During these 3 cycles, it remains awake maximum 50% percent of the time seeking for the beacon. Hence, the cost of a resynchronization is proportional to the inter beacon interval. For example, the CC1101 chip radio consumes 16.9 mA in reception, hence, during a resynchronization, the radio will consume at most (in Joules):

$$Power_{resync} = 3 \times 0.50 \times 16.9 \times 10^{-3} \times avg\_beacon\_interval \tag{4.13}$$

c) Comparison with a common layered architecture

We want to estimate the cost of maintaining the neighborhood when using a typical architecture, *i.e.* using a common duty cycle MAC protocol to send beacons. With this purpose we have set up a star topology, where a node simulating the initiator sends a beacon to each associated node. For these experiments, we have chosen ContikiMAC. We have varied the number of associated nodes and the interval between beacons (equivalent to the step). The packet acknowledgement is used by the initiator to determine the presence of the respective follower. By transmitting these "unicast" beacons, nodes can track each other in both directions as it is done in Wake on Idle.

Table 4.2: Duty cycle(%) of ContikiMAC

Assoc.	Beacon Interval			
nodes	1s	2s	16s	256s
1	6.4821	5.6778	1.0456	0.2894
2	7.1441	6.6270	1.2378	0.2987
4	8.3861	7.5988	1.4698	0.3200
8	8.7661	8.7956	1.8567	0.3435

Table 4.3: Duty cycle(%) of WoI

Assoc.	Beacon Interval			
nodes	1s	2s	16s	256s
1	0.3910	0.1070	0.0260	0.0014
2	0.8310	0.4400	0.0516	0.0035
4	1.7310	0.8301	0.1191	0.0072
8	3.7141	1.6810	0.2120	0.0171

We have adapted the duration of the sleeping interval of ContikiMAC in such way that the initiator has the time to correctly transmit beacons to each associated follower.

Table 4.4: Beacon Delivery Ratio: ContikiMAC

Assoc.	Beacon Interval			
nodes	1s	2s	16s	256s
1	0.9874	1.0000	1.0000	1.0000
2	0.7870	0.8650	1.0000	1.0000
4	0.4908	0.6200	0.9333	1.0000
8	0.2845	0.3660	0.8421	1.0000

The sleeping interval is determined by the sampling frequency and affects the duty cycle value. Table 4.2 summarizes the duty cycle values when using ContikiMAC; to facilitate the comparison we have summarized the duty cycle values of our protocol in Table 4.3. We observed as well that as contention is increased, the performance of ContikiMAC drops dramatically. The ratio of correctly delivered beacons or Beacon Delivery Ratio (BDR) is summarized in Table 4.4. The transmission of one beacon per second adds a significant overhead; that is why algorithms, such as trickle, propose increasing this interval according to changes observed in the network.

In Wake on Idle, since nodes are synchronized they can easily determine the presence of neighbors while correctly coping with contention.

4.5 Efficient Medium Access Control with Wake on Idle

When using Wake on Idle for neighborhood maintenance, the digital circuitry of the radio is deactivated and there is no exchange of digital frames between nodes. Therefore, when a node wants communicate with a neighbor and exchange information it needs to notify its neighbor about its willingness to communicate and the necessity to activate the digital processing of packets.

We can profit from the "time encoding" based on pseudo-random sequences and tight synchronization provided by Wake on Idle. We then propose a "code violation" notification mechanism. In this mechanism, efficient wake up is performed by omitting the transmission of the corresponding busy tone. Then, when a node has a pending frame for a given neighbor it does not transmit its analog busy tone. When the receiver misses the analog beacon, it activates digital processing of data during the following time slot, to perform data exchange. The name of the protocol "Wake on Idle" comes from this "code violation" wake up mechanism: *wake up on the absence of signal*.

4.5.1 Related work

In Chapter 3 we have summarized proposed medium access control mechanisms for Wireless Sensor Networks. Wake on Idle mixes characteristics from synchronous and asynchronous protocols: pairs of nodes are synchronized and have common wake-up schedules that are independent from the schedules of other pairs of nodes. It minimizes idle listening since pair of nodes are tightly synchronized and it considerably reduces the

collision probability by introducing random channel access, *i.e.* it uses pseudo-random schedules.

Additionally, Wake on Idle integrates neighborhood maintenance and medium access control. As explained at the beginning of this chapter the integration of layers and tight interaction between them reduces the complexity of the communication stack. Mechanisms for cooperation and unification of layers to optimize communication tasks have been widely studied; these mechanisms are what is called "cross-layer" design.

Several previous work have proposed integrating medium access control and routing layers to account for simplicity and to reach energy efficiency. Examples of these integration approaches are AIMRP (Address-light, Integrated MAC and Routing protocol) [95], EEMR (Energy Efficient integrated MAC and Routing protocol) [96] and MeeCast [97]; these protocols have been thought for convergecast traffic and have been evaluated only through simulations.

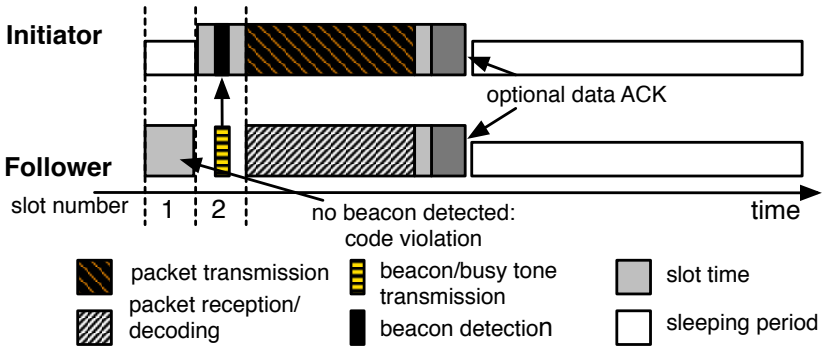
Synchronous MAC protocols, such as slotted IEEE 802.15.4, provide as well topology control through the transmission of periodic beacons. In this case the topology is limited to a cluster-tree; topology formation and route creation are managed by higher layers but maintenance of associations through the use of beacons is done by the MAC layer. In this case, neighborhood maintenance is unidirectional *i.e.* only associated nodes are aware of the availability of its coordinator but not vice-versa. FLAMA [98] provides a mechanism to form and maintain a cluster-tree topology by alternating random access periods with scheduled periods. Similarly, Dozer [99] integrates routing, topology control and medium access; Dozer provides local synchronization and achieves very low duty cycles. It uses a synchronous medium access with a cluster-tree topology and periodic beacons to maintain the topology. Nodes can have a parent and a set of children. Contrary to Wake on Idle, parent and children have a common schedule and medium access is done using TDMA. Bidirectional maintenance is not provided since only the parent transmits its beacon. Furthermore, Dozer provides only data gathering, *i.e.* upload traffic or convergecast, since children send information to its parent during the active period. Wake on Idle permits the formation and maintenance of point-to-point topologies and both data gathering and dissemination. The main advantage of Wake on Idle is the possibility to follow several neighbors (*i.e.* more than one parent) while no digital processing of packets is required. Then, extremely low duty cycles can be attained.

4.5.2 Medium access overview and implementation

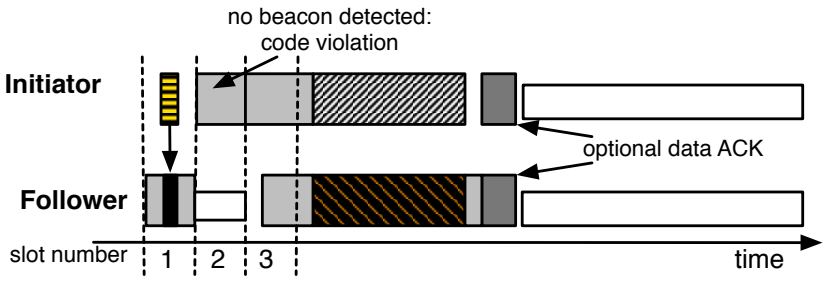
Figure 4.19 illustrates the procedure followed by pairs of associated nodes when data has to be exchanged. Frame acknowledgement can be added to account for reliable packet delivery. When associated nodes wake up to track each other there are four cases to handle:

- *Nothing to send:* no particular operation.
- *The initiator has a pending frame:* (Figure 4.19a) It discards its next beacon, thus the follower is informed that a frame transmission will happen immediately after its own beacon transmission. Then, the follower enters receive mode listening for a transmission.

- *The follower has a pending frame:* conversely, as shown in Figure 4.19b, the follower discards its beacon to notify the initiator. The later enters receive mode, but the follower will start transmitting only after an additional time slot.
- *Both nodes have a pending frame:* both nodes will discard their beacons. The additional slot introduced in the previous case gives priority to the initiator.



(a) Initiator transmitting



(b) Follower transmitting

Figure 4.19: WoI Medium Access Control: wake up mechanism

The association and synchronization procedures and implementation details were already described. Thus, for medium access we just have to take care of providing well-bounded time slots to avoid missing packets and collisions with the respective associated node. Moreover, packet and acknowledgement reception can also be used to correct clock drift and maintain correct synchronization.

Another important aspect is the mechanism applied when accessing the medium; we have already defined a slotted mechanism to avoid collisions with the associated node. However it exists the possibility of collision with beacons and packet transmissions from other nodes in range. Given that CCA is not performed before sending a beacon, collisions of an ongoing packet with a beacon cannot be avoided. We evaluate the medium access control mechanism by studying the probability of collisions with busy tones and with other packets.

4.5.3 Theoretical and simulations evaluation

We now try to evaluate the performance of Wake on Idle. We are interested in quantifying the probability of having a collision at the MAC level; this helps us to

determine if it is necessary to add a CCA procedure before sending packets. We will start by a theoretical analysis of collision probability, then, we present simulations and real test-bed results. We also evaluate important factors such as duty cycle and packet delivery ratio.

a) Packet collision probability

Similarly to the analysis performed in the last section, we can estimate the probability of having a collision between a beacon and a packet transmission. This is the probability of having a beacon transmission while a packet is being sent. The expression is given by:

$$P_c^j = \sum_{i=0}^{n} P(t_{stx}^j \leq t_{tx}^i \leq t_{etx}^j) \times P(i \neq j) \tag{4.14}$$

This takes into account collisions generated by beacons transmitted by a pair of associated nodes. The term inside the *Sum* represents the probability of having an exchange of busy tones while a packet of a given size is being sent. The value t_{stx}^j is the moment node j starts the transmission of a packet and t_{etx}^j is the end of the transmission. The value t_{tx}^j is the time a node different of node j starts transmitting a beacon. The first probability value is estimated in Equation 4.15. And the second term was already computed in Equation 4.9.

$$P(t_{stx}^j \leq t_{tx}^i \leq t_{etx}^j) = \frac{P_{duration}}{b} \tag{4.15}$$

Where $P_{duration}$ is the duration of a packet transmission and depends on the data rate (250 kb/s in in our implementation) and the packet length. The final expression is given by:

$$P_c^j = n \times (1 - \frac{1}{n}) \times \frac{P_{duration}}{b} \tag{4.16}$$

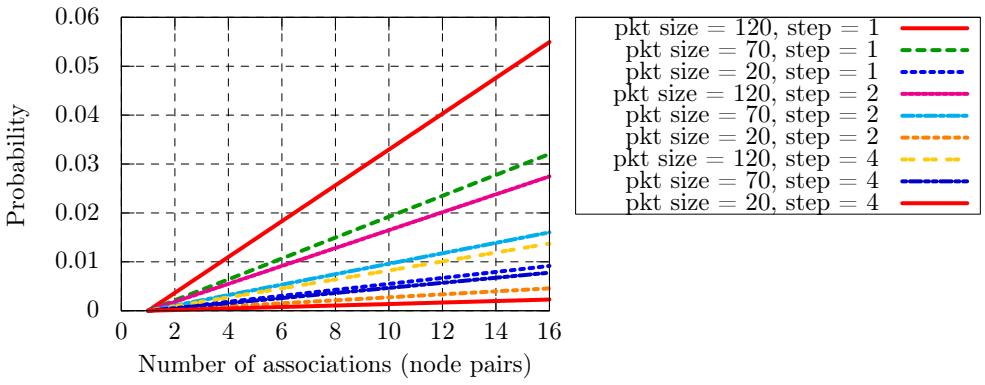


Figure 4.20: Packet-Beacon collision probability

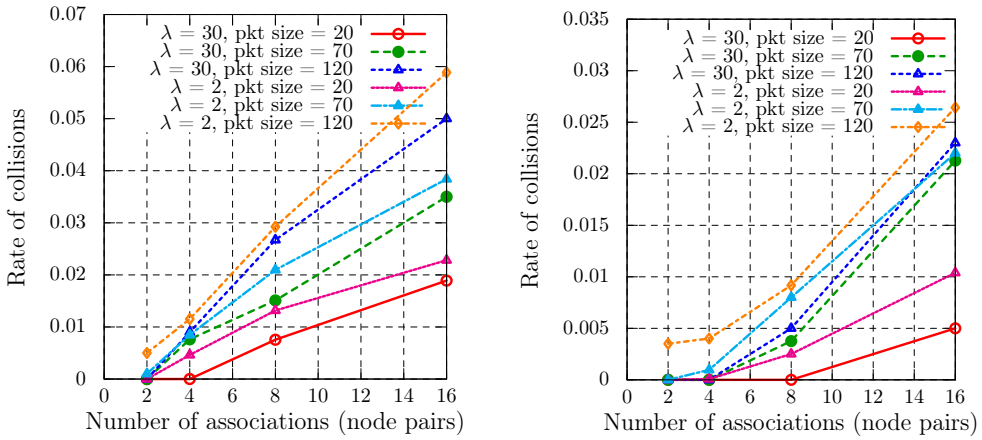
The probability of having a packet colliding with a busy tone for different step values and packet sizes is plotted in Figure 4.20. The x-axis corresponds to the number of

associations in the same transmission range. As expected, as the step value decreases and the number of associations increases the probability of colliding is larger. The worse case, *i.e.* long packet, 16 associations and step=1, leads to a collision probability of around 5.5%; the introduction of acknowledgements and retransmissions may help coping with errors introduced by these collisions.

b) Timing simulations

We perform timing simulations in Matlab for different traffic intensities and parameters such as packet size and step. All nodes generate the same traffic and they simulate sets of associated pairs in the same transmission range. We consider collisions involving MAC frames, either with another MAC frame or with a beacon.

The average rate of collided packets during one hour simulations are plotted in Figure 4.21. Packet transmissions are generated according to a Poisson process only on the initiator, for which we vary the intensity λ. A step of 1 gives one beacon every 1 s on average; similarly step of 2 gives an average one beacon every 2 s.



(a) step=1 **(b)** step=2

Figure 4.21: Probability of collision at the MAC layer

The worse simulated case is plotted in Figure 4.21a, *i.e.* step equal to 1, 120 bytes packet and λ equal to 2 seconds. For this case, the rate of collisions reaches a maximum of around 6% for 16 associations.

For more realistic Wireless Sensor Networks scenarios, one packet every 30 s for every node on average and lower association densities, the collision probability is below 1% for 120 bytes packets, and almost 0 for 20 bytes (Figure 4.21b). Hence, for a typical Wireless Sensor Network application we can have reliable medium access; in fact, the use of pseudo-random schedules has very nice properties, one of them is the small probability of colliding with other transmissions. The use of CCA may be useful for high density or contention, *i.e.* several associations or very small step value.

4.5.4 Experimental evaluation

For real test-bed evaluation we provide a simple implementation of the medium access based on the implementation of Wake on Idle described in detail in the former chapter. Synchronization is performed based on time estimations obtained from the exchange of analog busy tones and from the exchange of data packets. At the moment of the evaluation, no acknowledgements or retransmissions were implemented. By Frame Delivery Ratio we mean the number of packets that were received correctly at the destination. The use of acknowledgements may help to reduce the probability of loosing synchronization since these frames may be used, as well, to calibrate the synchronization mechanism.

a) Comparison with PW-MAC

PW-MAC or Predictive Wake up MAC is a receiver-initiated MAC protocol proposed by Tank *et al.* [80]. Similarly to our proposition, PW-MAC uses pseudo-random sequences to schedule wake up intervals. However, the novelty of Wake on Idle comes from the idea of waking up a packet recipient on the absence of (analog) signal, and the possibility of continuously tracking the presence of neighbors in a bidirectional way. PW-MAC provides as well synchronization between nodes, using an on-demand mechanism. The synchronization mechanism requires large wake-up in advance margins ($\sim$ 30 ms) compared with the margins required by Wake on Idle. Besides, neighbors are not continuously tracked.

Despite we do not have an implementation of PW-MAC we want to compare it with Wake on Idle. Hence we have attempted to reproduce the experiments carried out by the authors when evaluating PW-MAC [80].

- *Conflicting schedules*

  One of the advantages of using pseudo-random schedules is that it avoids conflicts of schedules. These conflicts arrive when nodes in the same transmission range choose overlapping schedules; hence, they wake up during common instances causing collisions and contention.

  To demonstrate this characteristic, the authors of PW-MAC have set up four nodes in a common range. Two of them are senders and the other two are receivers, and they form pairs of communicating nodes. The authors have compared PW-MAC and WiseMAC; also, they have initialized WiseMAC nodes in a way that they have conflicting schedules. We have set up the same scenario, using two followers and two receivers as shown in Figure 4.22.

  We have used an average inter beacon interval of 500ms, *i.e.* step equal 0.5 . Then, we generate traffic based on a Poisson distribution with an average (λ) of 1 second. The values were averaged from a set of 5 experiments with a duration of 300 s. The results obtained for PW-MAC, WiseMAC and WoI are summarized in Table 4.5. Despite we did not implement retransmissions we have observed a FDR larger than 97%. PW-MAC reaches 100% by introducing a retransmission mechanism.

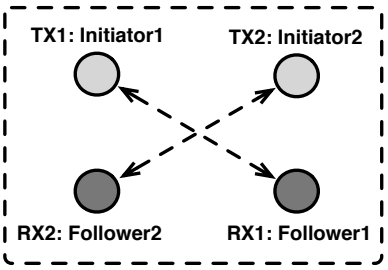


Figure 4.22: Evaluated scenario with conflicting schedules

Table 4.5: Sender duty cycle and latency in a conflicting schedule scenario [80]

	Sender duty cycle	Packet latency
WiseMAC	78.1%	17425ms
PW-MAC	5.5%	579ms
WoI	2%	424ms

- *Hidden terminals*

  The performance of contention-based protocols may be significantly affected by *hidden terminals*. A hidden terminal is a station whose transmissions are not sensed or detected by a node but interfere with the node's transmissions. Hence, transmissions from hidden terminals may collide at the receiver and frame delivery can be dramatically reduced. In general, nodes' wake up intervals in Wake on Idle and PW-MAC do not coincide; then, transmissions from hidden terminals should not cause performance degradation.

  The authors of PW-MAC have set up a scenario with two senders and two receivers. Senders constitute a pair of hidden terminals. They have shown that the performance of WiseMAC drops significantly in this scenario while PW-MAC keeps almost the same duty cycle and delay values with respect to the experiments presented before.

  We have tested the same scenario using Wake on Idle, we have observed no effect of hidden terminals on communication performance: duty cycle, delay and FDR remained almost unchanged (Table 4.6).

Wake on Idle keeps the same characteristics as PW-MAC. Furthermore, thanks to a more precise synchronization mechanism, Wake on Idle is able to reach lower duty cycles while handling the same traffic. In addition, synchronization is kept even when very long sleeping periods can be used; hence, very low duty cycles can be achieved. Long inactive periods are desirable for applications that do not require low latency but are highly constrained in terms of power consumption. Still, since synchronization is kept, communication remains efficient.

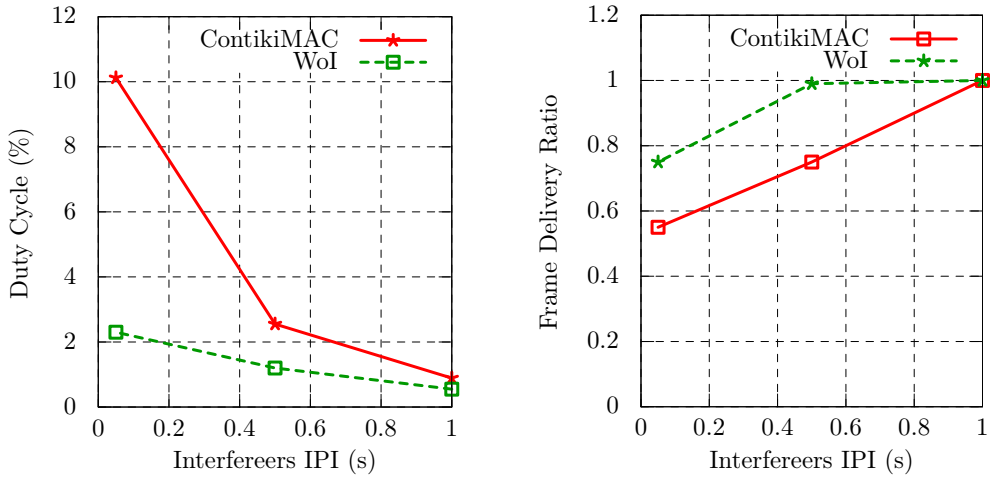
Table 4.6: Sender duty cycle and latency in a hidden terminal scenario [80]

	Sender duty cycle	Packet latency
WiseMAC	81.6%	10905ms
PW-MAC	6.8%	670ms
WoI	2.2%	430ms

b) Multihop topology evaluation and Comparison with ContikiMAC

To evaluate the duty cycle when using Wake on Idle for medium access we have generated the delivery of packets in a simple multihop topology; we have set up three lines formed of four nodes each. On each line we deliver 128-bytes packets from end to end of the line with a frequency of 5 seconds (*i,e.* generated in a random way). We have generated simultaneous flows of packets. The scenario has been evaluated using both Wake on Idle and ContikiMAC. For Wake on Idle we have used a step of 1 and we have used a sampling frequency of 2 Hz for ContikiMAC. These values generate an equivalent wake-up period.

Both implementations are executed in the WSN430 nodes, in SensLAB. However, the implementation of ContikiMAC is available for the CC2420 radio. For practical reasons and characteristics of the radio chip, Wake on Idle uses the CC1101 radio. Both radios have a data-rate of 250 kb/s (the data-rate and modulation scheme can be changed in the CC1101); radios differ on the communication frequency band and channel bandwidth, the modulation and encoding scheme, the CC2420 radio being more robust to external interference and compliant with the IEEE 802.15.4 standard.



(a) MAC Duty Cycle (%) **(b)** Frame delivery ratio

Figure 4.23: MAC layer results vs interference intensity

We have set-up four interferers surrounding the network and we have varied the

intensity of interfering traffic. Figure 4.23a shows the average duty cycle values in percentage, as a function of the interference or inter-packet interval (IPI) of the traffic generated by the interferers. Figure 4.23b plots the Frame Delivery Ratio (FDR) in presence of interference; by FDR we mean the fraction of frames correctly delivered from end to end (without any retransmission).

The duty cycle increment in ContikiMAC can be explained by the congestion generated by interference which causes the nodes to wake up even if there are no outgoing transmissions. In Wake on Idle, the duty cycle is increased due to loss of busy tones caused by false positives generated by interference; such losses causes increment of listening window and sometimes it causes loss of synchronization. Packet losses are due to collisions, false positives and interference.

4.6 Frequency hopping extension

4.6.1 Advantages of using frequency hopping

Several desirable characteristics can be added using frequency hopping. The main causes of communication failures are due to channel conditions: *multipath fading* and *external interference*.

External interference can be caused by transmissions from external networks that use the same frequency band. In a real life scenario we expect several networks to coexist in the same environment and it is necessary to cope with interference in a way that performance degradation is minimal. There are channels that can be more interfered than others and transmissions are very likely to fail under external interference. Changing constantly the communication channel *reduces the probability of colliding concurrent transmissions.*

In a single channel approach links are coherent along intervals of time; it means that when a transmission fails there is a high probability that retransmissions fail as well (*i.e.* bursty channels). When using a different channel for retransmissions, channel failures are independent events and depend only on the condition of the new channel. As the quality of a given link is averaged over multiple channels, it is possible to have *more stable links and routing paths, and a more stable topology.* Furthermore, based on observations, a "black" list of channels can be maintained in such a way that stations are able to avoid bad channels and increase the probability of successful communication. However, it is necessary to introduce a proper procedure to put back these channels into the "white" list when they become available. Watteyne *et al.* [100] have evaluated the impact of channel hopping at the routing level; they have shown a significant improvement of network efficiency and stability when channel hopping is introduced, compared to a single channel approach.

Furthermore by adding some cryptographic mechanisms for PRNG seed exchange and sequence generation a robust mechanism for *secure communication* can be provided [101]. *Jamming can be avoided* with the use of multiple channels and further techniques for hopping pattern generation may help to avoid *intrusions and other attacks.*

4.6.2 Hardware support for channel hopping

Most of the radio chips available in the market posses frequency agility. There is an overhead in terms of delay and energy consumption due to the necessity of changing the transmission channel every time a beacon is exchanged between a pair of associated nodes. Changing the frequency requires calibrating the oscillator and other components in the radio chip; this calibration process may require some time. Usually this procedure is performed very efficiently; for instance, the CC1101 and CC2500 provide three modes for calibration when implementing frequency hopping. Delay and complexity of this calibration task must be traded off. The method chosen in our implementation requires between 220 μs and 232 μs.

4.6.3 Wake on Idle and channel hopping

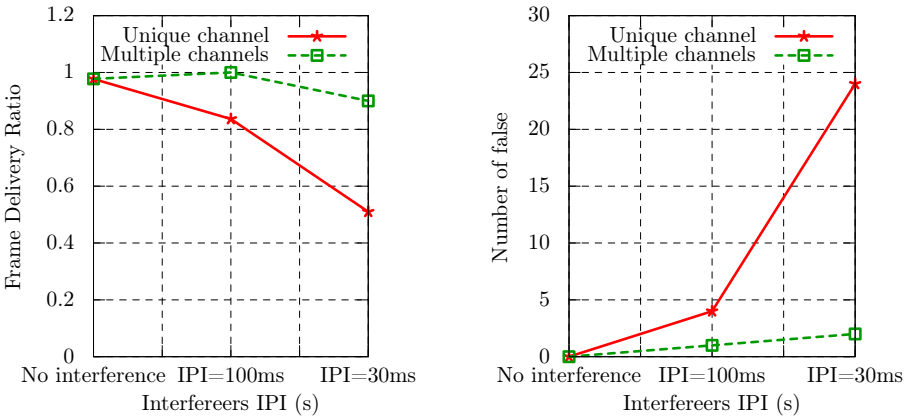
For proper rendezvous, frequency hopping requires nodes to be tightly synchronized. Wake on Idle provides synchronization between nodes; additionally nodes are synchronized according to a pseudo-random sequence. Such sequence can be used as well to generate a rendezvous channel for busy tone exchange and possible packet exchange. Then, pairs of nodes access a random channel at a random instant. Given a set of available channels, the next rendezvous channel can be computed according to the pseudo-random generated value as:

$$\text{Next\_channel} = (\text{Next\_pseudo\_random\_value})\% \ (\text{Number\_of\_channels}) \tag{4.17}$$

Such a random access, which exploits both time diversity and frequency diversity, provides further robustness to external interference, reduces the probability to collide with other transmissions and provides further security since the process to intercept consecutive transmission becomes complex (*i.e.* time and frequency encoding). EM-MAC is an extension of PW-MAC which introduces frequency hopping. EM-MAC [81] has shown to outperform PW-MAC in presence of contention and interference. We then expect to improve the performance of Wake on Idle by adding frequency hopping.

4.6.4 Evaluation of frequency hopping

We have implemented the frequency hopping extension of Wake on Idle; we aim at evaluating the performance of the protocol when frequency hopping is added. With this purpose we have set up a pair of associated nodes and we have placed two interferes generating high intensity traffic on a given channel. For the multi-frequency case, interference is generated in different channels on a random way. We have tested both single and multiple channel (using 12 channels) implementations of Wake on Idle and we have used a step value of 1 (*i.e.* average inter-beacon interval of 2 s). During the implementation of frequency hopping we have observed that the delay introduced when changing channels is very short compared to margin values, furthermore it remains almost unchanged; hence, we keep the same wake-up in advance margin we have used for the single channel implementation. We already have discussed that Wake on Idle can be robust to interference, even when a single channel is used; in fact, we have observed that, by setting the correct CCA threshold value, synchronization can be kept and analog busy tones can be detected in presence of interference.



(a) MAC Frame Delivery Ratio

(b) Number of false positives

Figure 4.24: MAC layer experimentatal results according to interference intensity (with and w/o frequency hopping)

However, at the MAC layer the effects of interference can be more significant; since in this case packets need to be decoded, it is possible that interferences damage the digital packets and they cannot be decoded and delivered. We have already observed in 4.5.4 that performance of Wake on Idle, in terms duty cycle and FDR, decreases in the presence of interference. The results in terms of Frame Delivery Ratio versus the intensity of the interference traffic (Inter Packet Interval) are plotted in Figure 4.24a. It can be observed that the FDR value at the MAC layer (without the addition of ACKs and retransmissions) is significantly improved using channel diversity. We have used 12 channels during the experiments; we expect that as the number of possible channels increases the FDR is closer to 100% . This is due to the fact that the probability to transmit on the same channel as the interferer is reduced.

Figure 4.24b plots the average number of detected false positives in an interval of 5 minutes; when using multiple channels, the probability of having a false positive decreases as well as the probability of loosing synchronization.

4.7 Wake on Idle hardware module

The Wake on Idle mechanism has been conceived to work efficiently in the stable state; where the absence of analog signal, at precise instants, informs about the necessity to listen to oncoming packets or the loss of synchronization with a given neighbor.

We have presented an implementation of Wake on Idle in current platforms. In this implementation, the main processing unit (microcontroller) must wake up and activate the radio to transmit/detect the analog busy tone. Then, it estimates clock drift, generates the next pseudo-random value and programs the timer module; this process is continuously repeated during the stable state. As in all typical implementations, the microcontroller is in charge of controlling all peripherals and executing all tasks: from applications to radio chip configuration and control.

The operation of Wake on Idle is very similar to the Wake on Radio (WoR) feature. As explained earlier, when using Wake on Radio the radio chip wakes up periodically to sense the RSSI level of the channel (LPL) and wakes up the CPU if it detects an RSSI level higher than a given threshold. In Wake on Idle we want the opposite operation: the radio is activated at well-defined instants, it senses the medium and if it does not find an RSSI peak it wakes up the CPU. Then, the CPU must analyze the frames that were received or activate the resynchronization procedure. We then discuss how Wake on Idle can be implemented and integrated with existing radio chips.

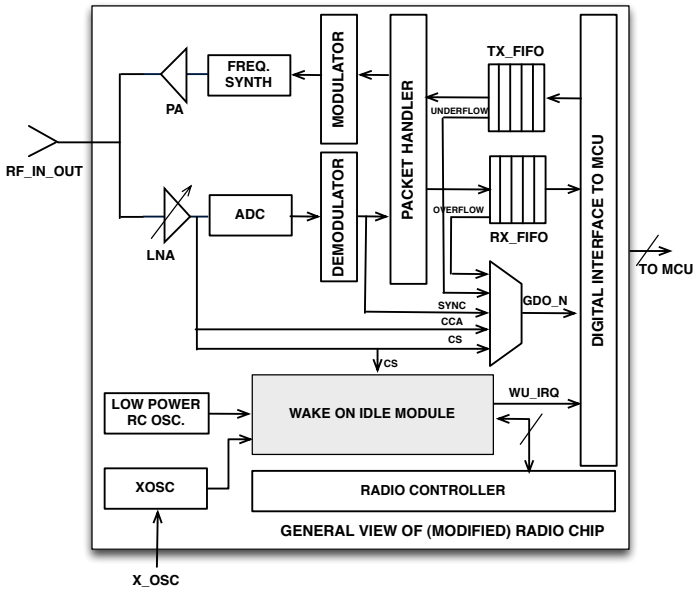


Figure 4.25: Simplified block diagram of radio chip with Wake on Idle

Figure 4.25 shows a simplified block diagram of a common radio chip integrated with Wake on Idle. The Wake on Idle block must be interconnected to the radio front end, it must be able to activate it and retrieve the RSSI value and carrier-sense signal in order to detect the analog busy tone. Then, when necessary, *i.e.* absence of analog busy tone, it must be able to generate an interrupt that, through the microcontroller interface, wakes up the microcontroller. Additionally, it needs a low-power oscillator source in order to account for inactive periods, while energy consumption is minimized. This low-power oscillator is already present in some existing radio chips and it is used for the Wake on Radio functionality. Finally a set of registers to configure the Wake on Idle module must be added.

We design the Wake on Idle block as a specialized hardware unit that is configured trough a set of registers and accessed through a general input/output interface. It generates the IRQ to be sent to the microcontroller. A general overview of the Wake on Idle hardware block diagram is illustrated in Figure 4.26. The figure represents a possible hardware implementation: a control unit connected to different hardware blocks. The control unit consists of the initiator and/or follower finite state machine.

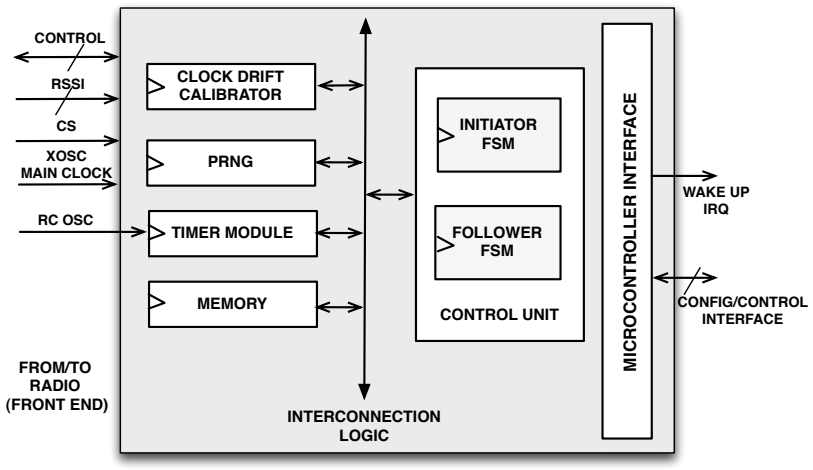


Figure 4.26: General overview of Wake on Idle module

The different blocks and their possible implementation are described below. The IP cores can be implemented and interconnected in several different ways. According to the technology and area constraints more reduced data paths can be used, at the cost of lower performance.

- *Clock drift calibrator*

  As explained in 4.3.2 the error introduced by clock drift is estimated every time an analog busy tone is received. We propose a second order IIR filter to compensate such error. A filter can be implemented using the data path illustrated in Figure 4.27. A Multiplier-Accumulator a set of registers and multiplexers are enough to execute filtering; for a transfer function such as the one shown in Equation 4.2 four Multiply-Accumulation (MAC) operations are needed. Even if multiplication is an expensive task in software, hardware multipliers are fast end efficient.

  The filter controller is in charge of choosing the Multiplier-Accumulator operands, controlling the multiplexers and arithmetic units. It can be implemented by a very simple FSM: minimal combinational logic combined with sequential logic *i.e.* a flip-flop.

  The filter order and coefficients depend on the characteristics of the low-power oscillator being used in the implementation. These parameters could be configurable according to operational conditions.

- *Pseudo-random number generator*

  The multiply-with-carry PRNG described by Equation 4.4 can be implemented as shown in Figure 4.28. It uses a data path similar to the one used by the drift compensation filter: A Multiplier-Accumulator, a set of multiplexers and an additional shifter to access the carry value. Then, the PRNG and the clock drift correction units may use the same data path.

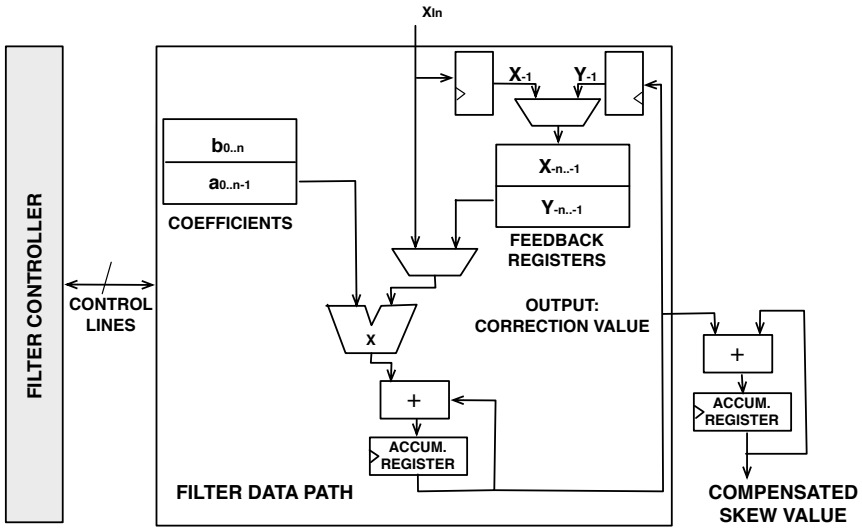


Figure 4.27: IIR filter implementation for clock drift calibration

- *Timer module*

  The source of this timer module is a low-power oscillator; existing radio chips already integrate these oscillators. This is the only module that remains active during the sleeping interval and it is in charge of waking up the whole Wake on Idle module as soon as the timer fires.

- *Control unit*

  The control unit is in charge of controlling the modules presented before: pseudo-random number generation, clock drift correction, radio front-end controlling and timer events programming. Events and tasks carried out depend on whether the node behaves as a follower or as an initiator, or both. Also, it decides whether the microcontroller must be waken up. The main control unit can be implemented by the finite state machines described in 4.2.4 and 4.2.3.

- *Packet management unit*

  Using the PRNG seed and/or information about the packet source and destination, this unit can filter packets that are not destined for the node; also, it takes the position in the pseudo-random sequence to reinitialize the PRNG and resynchronize.

- *Storage*

  To maintain a neighbor, both, follower and receiver, need some memory or storage. The memory requirements depend on design factors such as the PRNG implementation, the addressing format and the drift correction accuracy. For instance, the initiator needs to keep the PRNG seed value (2 bytes in our implementation), the node address or id (1 to 4 bytes according to the application), and two registers to keep the current PRNG carry value (4 bytes in our implementation) and the

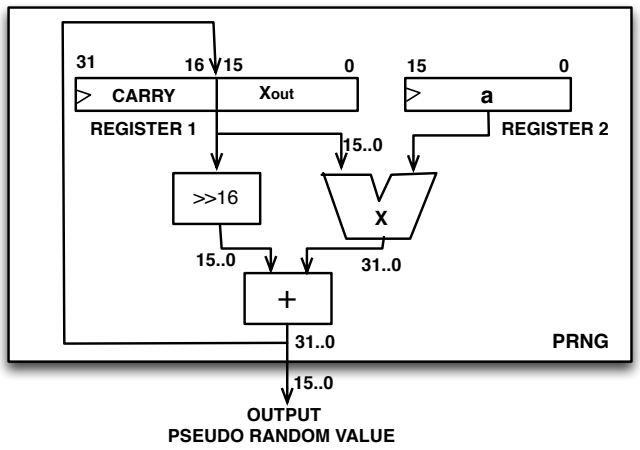


Figure 4.28: Multiply-with-carry PRNG implementation

position in the PRNG sequence (2 bytes). Then, in our implementation, the initiator needs at least 9 bytes and at most 12 bytes per follower.

Regarding the follower, it needs some additional memory in order to keep synchronization. Thus, besides the registers mentioned above, it needs to keep the slope value (2 to 4 bytes according to the precision, in fixed-point representation); also, if the clock drift correction mechanism uses a filter it needs one register per filter order; for example, if a second order filter is used, two registers are needed to keep the past estimation error samples (2 to 4 bytes each register according to the precision). Then, the follower needs at least 4 and at most 12 (for a second order filter) additional bytes per initiator in order to keep synchronization.

A specialized processing unit (an ASIP or a low-power DSP) can also be introduced into the radio chip. Such processors allow to execute the Wake in Idle protocol providing more flexibility thanks to re-programmability.

4.8 Efficient neighborhood discovery

We study the problem of neighborhood discovery in Wireless Sensor Networks since it remains an open issue of Wake on Idle. In this section we present an overview of existing mechanisms and we discuss possible solutions. The necessity of efficient neighborhood discovery opens future directions for the work presented in this thesis.

4.8.1 Related work

Neighborhood discovery when using a duty cycle medium access protocol is not a trivial task. In order to have two nodes to discover each other they must be active at same time at least once. Several questions can be formulated here:

- How to guarantee that all nodes in the same transmission range have at least one common active period?

- Given that nodes have no knowledge about the number of available nodes in the network, how long the discovery process should last? How do the nodes determine whether the network is fully connected?
- How a node decides if a detected node can be a potential neighbor?
- In case of loss of connectivity or new nodes association, how the association process should be done?

The use of "wake up" radio circuitry presented in Chapter 2 would facilitate the neighborhood discovery procedure; it gives nodes the capacity to wake up neighbors on its transmission range and immediately start the association procedure. However, these technologies are not still suitable for Wireless Sensor Networks applications and an alternative solution has to be found.

Extensive work has been done with the purpose of solving the neighbor discovery problem; however, no optimal solution has been found. There is a compromise between power consumption, complexity and latency; as for every design choice in a Wireless Sensor Network application, the proper solution must be chosen according to the scenario, the communication protocol stack, the constraints and so on.

Proposed discovery mechanisms for Wireless Sensor Networks can be classified according to the rendezvous mechanism. The simplest approach to guarantee deterministic neighbor discovery is keeping the nodes awake for at least half of the time, *i.e* duty cycle of 51%; however, power consumption can be unacceptable for most Wireless Sensor Network applications. Some optimizations to the 51% percent method have been proposed but they lead to a very long discovery time. *Birthday protocols* [102] use a slotted approach where nodes choose, based on a probabilistic method, whether they transmit, listen or sleep during a given slot. Randomization guarantees that after a given period of time, active periods of all nodes have overlapped at least once and they are able to discover each other. Another example of random protocols is presented by Borbash S. *et al.* [103]. For proper functionality, these protocols require that each node knows about the total number of nodes present in the network.

Other protocols [104] use patterns for schedules that have intersecting active periods with respect to a rotation and guarantee a minimum duty cycle; despite this deterministic approach achieves efficiency in terms of discovery delay and energy consumption, its software implementation can be very complex and impractical given the constrained computation resources available in most Wireless Sensor Networks platforms.

Other methods make use of co-prime schedules; these protocols use prime numbers to determine the number of slots the node remains inactive. Such mechanisms rely on the Chinese Remainder Theorem [105] where nodes choose slotted schedules based on relative prime numbers. They guarantee that, after a bounded amount of time, active periods of two different nodes overlap and they are able to discover each other. The main problem arrives when nodes must choose their schedules: *i.e.* for a node to discover a neighbor its schedule must be co-prime of the neighbor's schedule. If prime numbers are chosen, and two nodes choose the same number they will not be able to discover each other, some approaches try to cope with the problem of selecting the schedules [106, 107]. Also, according to the duty cycle, latency to discover neighbors may be too long.

Asynchronous MAC protocols using wake up techniques such as preamble sampling or low power listening may provide neighborhood discovery through the transmission of broadcast advertisements to create new associations. However broadcast transmissions in these kinds of protocols are expensive and the cost in terms on energy consumption is very high.

For protocols using frequency hopping or multiple channels the problem is even more complex. For instance, the Bluetooth [108] standard proposes an asymmetric discovery process where one node, called the active node or the *inquiry* device, sends beacons and listens to replies from neighbor nodes. A second device or passive device, listens to beacons and sends responses to these beacons to start the association procedure. Discovery is guaranteed by the fact that the inquiry device moves much faster than the passive one in the available frequency channels; in such a way, after some time, both nodes coincide in the same frequency. This asymmetric approach has been though for a single hop topology. In a multi-hop scenario nodes need to alternate between the *inquiry* and *non-inquiry* operation mode [109]; this fact greatly increases the complexity of the procedure. Fast frequency hopping guarantees rapid neighbor discovery but it is power consuming. Drula *et al.* [110] have proposed an adaptive algorithm for neighbor communication establishment in Bluetooth adhoc networks. The authors study the impact of different parameters when performing neighbor discovery in Bluetooth networks; then, an adaptive algorithm has been proposed: according to the probability of discovery success, parameters are set to optimize power consumption during the discovery stage. Other protocols propose the use of a control channel or operation channel for neighbor discovery [111, 112, 81], this channel can be unique to the network, to a cluster or to a node. However, some conditions such as channel failures or high contention in the control channel may avoid proper discovery of neighbor nodes.

Static scenarios may take advantage of an efficient neighborhood discovery protocol since at the moment of deployment nodes are able to form a fixed topology that will change with a very low probability; hence, after the formation of the topology, nodes perform tasks such as synchronization and data exchange and remain stable during a long interval of time; since usually neighborhood discovery is performed only at the beginning, nodes can maintain long sleeping periods. Hence, they reach very low duty cycles while connectivity is maintained. If the network is presumed to have changed and connectivity is lost, the discovery mechanism could be re-executed.

4.8.2 Neighbor discovery and Wake on Idle

The complexity of the neighbor discovery procedure is increased when using Wake on Idle; in fact, as soon as nodes enter the stable state the digital processing of the radio chip is deactivated most of the time. Then, nodes are not able to detect new nodes willing to join the network.

We strongly believe that for efficient neighbor discovery, when using a protocol such as Wake on Idle, we need to continuously switch between discovery periods where new nodes can join the network and form a new topology, and stable periods where nodes are synchronized. During discovery, a random access protocol (*i.e.* an asynchronous MAC) can be used by nodes to send and listen for broadcast (*i.e.* hello) frames and discover neighbors. The combination of different medium access protocols has already

been proposed in protocols such as FLAMA [98] and MaCARI [113] where random access is used to detect neighbors, form the (cluster-tree) topology and synchronize nodes. Then a synchronous (TDMA) protocol is used for data gathering. Nodes switch continuously between the two operating modes.

We are working on the integration of the Wake on Idle protocol with the "hybrid" neighborhood discovery mechanism and we discuss some aspects in Chapter 7.

4.9 Summary and Conclusions

In this chapter we have described Wake on Idle: a neighborhood maintenance and medium access control protocol thought to be implemented in hardware and integrated to the radio chip as a "smart" radio. We have designed and explained in detail a protocol for neighborhood maintenance and medium access based on analog channel sensing and pseudo-random wake up periods. We have shown that the use of pseudo-random sequences for wake up scheduling introduces desirable properties for node sensing and efficient medium access. The protocol has been extensively evaluated through probabilistic analysis, timing simulations and a real implementation.

The main advantage of Wake on Idle is the possibility of having neighborhood maintenance without the necessity of waking up the system (*i.e.* the CPU) or any digital processing of data. This characteristic leads to very low power consumption and, since the exchange of busy tones is managed directly by the radio, delays can be reduced resulting in very low duty cycles and very tight synchronization.

To show the feasibility of the protocol we have presented a proof of concept implementation using available platforms (WSN430 and eZ430). We have provided a simple mechanism to allow neighbor discovery and association of new nodes. The results show that, as soon as the stable state is reached, Wake on Idle achieves very low duty cycle values and accurate synchronization. We show that integrating MAC layer and neighborhood maintenance helps to optimize energy consumption. Optionally, further robustness to external interference is provided thanks to the frequency hopping functionality.

The implementation of the Wake on Idle module is also described; we have shown that it can be easily implemented in hardware and integrated within exiting radio chips, similarly to the existing Wake on Radio module.

Finally, we have talked about the problem of efficient neighborhood discovery; we discuss a possible solution and we leave its implementation and evaluation as an open issue and future direction of this work.

Chapter 5

Sleep on Idle: Duty Cycle Optimization

Contents

Similarly to Wake on Idle, Sleep on Idle (SoI) is a mechanism that exploits analog channel sensing for efficient medium access control. It uses a very simple decision algorithm that helps determining whether the node must remain active, listening for transmissions or if it can go to sleep to account for low energy consumption.

Following we describe Sleep on Idle; also, we present and evaluate its implementation. We discuss some important points to account for a proper operation of the protocol. The evaluated implementation of Sleep on Idle is based on the IEEE 802.15.4 standard.

5.1 Motivation and related work

The beacon-enabled IEEE 802.15.4 protocol has been described in detail in Chapter 3. To motivate the introduction of the Sleep on Idle mechanism, we recall the communication process between a node and its coordinator in the IEEE 802.15.4 protocol. The active periods, common to the coordinator and the associated nodes are delimited by the transmission of a beacon. The beacon is transmitted by the coordinator. Communication and data exchange can only be initiated and requested by the associated nodes or children; the coordinator can send packets to an associated node only if the latter has sent a packet request frame. In this way, before the transmission of its frame, the coordinator knows that the packet recipient is available during the current Superframe. This procedure is what we call *indirect communication*. This procedure is necessary

because the beacon signaling mechanism is *asymmetric*; a difference to protocols such as Wake on Idle where the exchange of beacons is done in a bidirectional way, in IEEE 802.15.4 only the coordinator transmits beacons.

At the beginning of the Superframe or active period, associated nodes check their transmission buffer and the data pending field of the received beacon. If there is data to be exchanged with the coordinator it remains awake until the transaction is finished; otherwise, it turns off its radio to save energy. However, the coordinator does not have a mean to know if there will be data exchanged during the Superframe and it must remain awake during the whole Superframe duration, listening for transmissions, even if it will be idle.

As we are interested in a multi-hop scenario, a big fraction of the nodes act as routers for packet forwarding, meaning that they have the role of coordinator and children at the same time. Then, during their own Superframe, nodes may have long idle periods leading to a significant waste. This waste of energy can be easily observed in low traffic applications, *i.e.* intermittent traffic or periodic traffic with a very long period, where several consecutive Superframes may remain idle. Since coordinators suffer the idle listening issue, they may exhaust their batteries faster than leaf nodes.

The community has tried to address the coordinator idle listening issue of IEEE 802.15.4. For instance, the IEEE 802.15.4 standard proposes the Battery Life Extension (BLE) feature; BLE aims at reducing the time the coordinator remains awake during a Superframe. A Superframe duration shorter than the minimum allowed by the standard can be set.

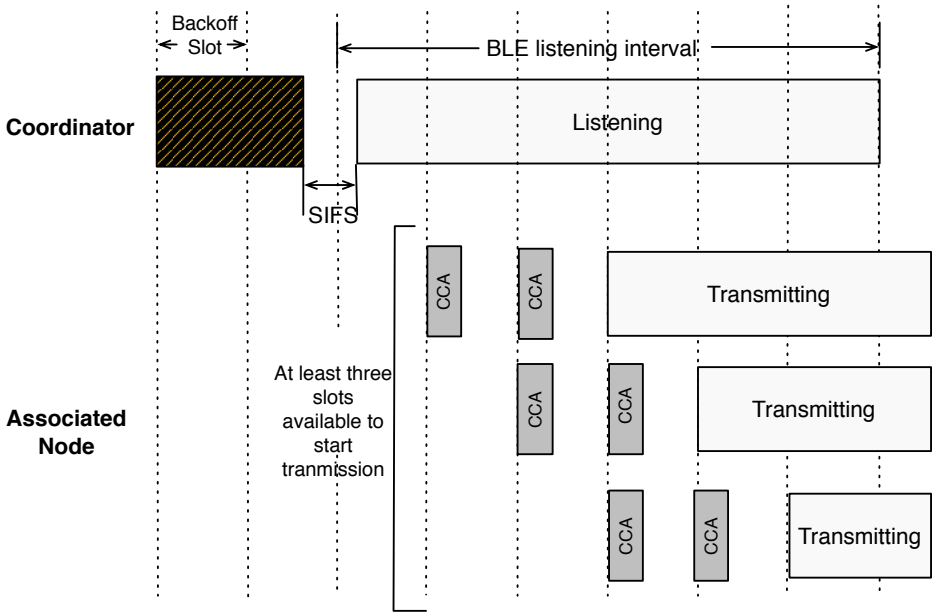


Figure 5.1: BLE functionality: The coordinator listens during 6 back-off slots

Figure 5.1 illustrates the functionality of BLE for both associated nodes and coordinator: as soon as the beacon is transmitted, the coordinator listens to a transmission *i.e.* looking for a SFD (Start of Frame Delimite), during the first n back-off slots.

The value of n is determined by the physical layer being used and ranges between 6 and 41; also, n should be long enough to execute the CSMA/CA algorithm and to transmit a PHY header, *i.e.* preamble and SFD. For the 2.4 GHz band, using O-QPSK modulation, n is equal to 6. The figure illustrates the case on which the shortest beacon is sent; after the beacon transmission the node waits 6 back-off slots. It can be observed that, in order to send a frame, the node must start its transmission at least three slots before the end of listening window (*i.e.* n slots). This is the minimum time to perform CCA and detect the transmission of a PHY header.

Then, if the coordinator detects a transmission during these n slots, it remains awake and decodes the packet; otherwise, it turns the radio off. Given the highly reduced contention period, the CSMA/CA parameters, such as the bounds for the back-off exponent (BE) value, are modified for faster access to the medium.

Despite BLE helps reducing idle listening, it may lead to several problems. For example, as back-off times are shortened, contention may increase as well as the probability to collide. If several nodes try to communicate during the same Superframe, correct delivery of frames may not be possible and frame retransmissions should be done in the next Superframe. As nodes can only send one frame per Superframe, the resulting delay may become too long. Consequently, the BLE mechanism can be useful only if the traffic pattern includes at most one transmission per Superframe, *i.e.* it is not possible to transmit more than one frame per Superframe and no burst transmissions are possible given the early deactivation of the coordinator. However, we believe that this functionality can be useful for energy harvesting systems: BLE can be activated when the energy budget is very low and the battery needs to be recharged; then, during this period, nodes do not exchange information but the exchange of beacons helps keeping the topology until a given level of energy is reached.

Additional work int the literature addresses the idle listening problem of IEEE 802.15.4. Some works propose dynamic adaptation of the duty cycle or Superframe duration according to the traffic. Examples of these algorithms are Beacon Order Adaptation Algorithm (BOAA) [114], Duty Cycle Adaptation (DCA) [115], Duty Cycle Learning Adaptation (DCLA) [116], Joint Duty Cycle and Link Adaptation [117], and among others. Even if these mechanisms can optimize the duty cycle and reduce idle listening at the coordinator, they can be very complex, limit scalability, or they may require changing the existing physical layer implementation.

Sleep on Idle aims at minimizing the energy wasted due to idle listening during the Superframes. The idea behind Sleep on Idle is introducing a signalization method to inform the coordinators if the current Superframe will be idle. A similar idea has been proposed by Dutta *et al.* [83] during the design of the A-MAC protocol. At the end of this Chapter we will discuss and analyze some details of the mechanism proposed by the authors of A-MAC.

5.2 Sleep on Idle as an extension of IEEE 802.15.4

When using Sleep on Idle nodes are able to notify their coordinator about their willingness to communicate during the Superframe. This notification is done by transmitting an analog busy tone at the beginning of the Superframe immediately after

the transmission of the beacon. Similarly to the Wake on Idle protocol, nodes do not perform any digital processing of these transmissions.

If two or more associated nodes have data to exchange during the Superframe they will send their analog busy tone immediately after the reception of the beacon. Then, at the coordinator side these transmissions overlap and a peak of RSSI will be observed. Some factors may introduce delays causing busy tone transmissions to be shifted one with respect to the other. These factors can be:

1. The signal *propagation time* which is negligible since distances between nodes are relatively short,

2. The *hardware delays* introduced when generating the signal. Since there is no digital processing of the transmitted signal, such delays are negligible as well,

3. The *software delays* introduced when managing different events and peripherals, *i.e.* delays introduced by the operating system. Also, associated nodes must decode and analyze the beacon before it decides if it has to transmit the analog busy tone. Such procedure may introduce some indeterministic delays.

The (quasi) superposition of busy tones causes a cumulative effect, *i.e.* an OR operation over the air. Then, the coordinator detects an RSSI peak and it will know that there is at least one frame to be exchanged during the active period. If the coordinator does not detect a busy tone during the sensing time it infers there will no be communication during the Superframe and it enters the inactive state until the next Superframe.

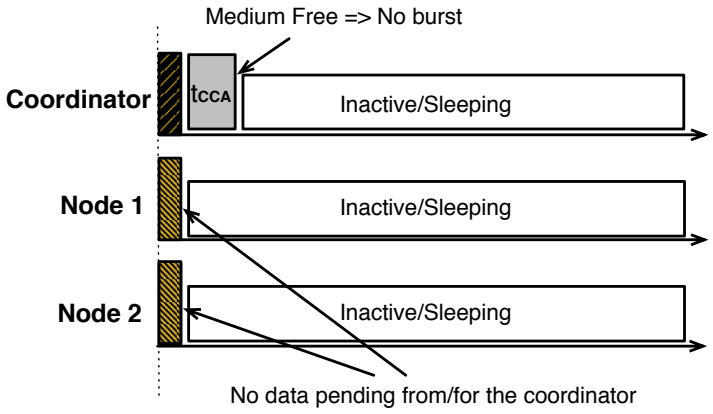
The integration of SoI with IEEE 802.15.4 is illustrated in Figure 5.2. In the figures, we assume that two nodes are associated with a Coordinator that periodically sends beacons. At the beginning of the Superframe two cases may arise:

- *No data to be exchanged during the Superframe:* (Figure 5.2a)

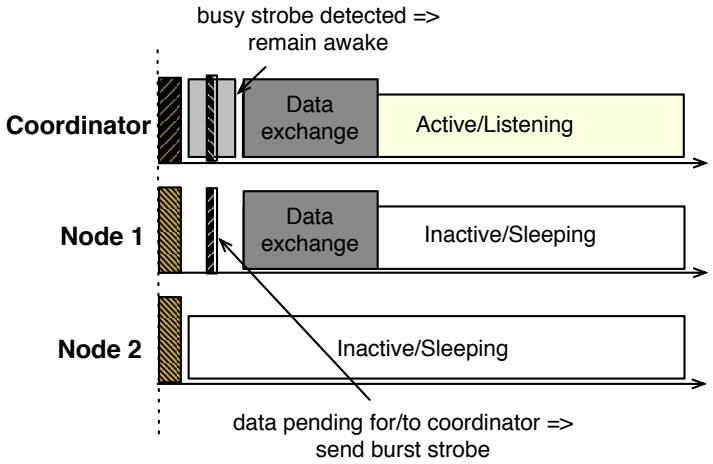
  Any of the associated nodes has a pending frame in its transmission buffer and the data pending field on the transmitted beacon is empty; then, there are no packets to exchange during the Superframe. In this case the associated nodes behave as in normal IEEE 802.15.4, they enter the sleeping state until the next beacon transmission. The Coordinator remains awake during a determined interval of time listening the medium and checking the RSSI level; after this time, the Coordinator determines that the medium is free and no packets will be sent by associated nodes. Hence, it turns off its radio until the following beacon transmission.

- *There are pending packets:* (Figure 5.2b)

  At least one child node has pending packets in its transmission buffer or there is data pending to be retrieved from the Coordinator. In this case the node(s) having data to exchange transmits an analog busy tone informing the Coordinator about its willingness to exchange data. Then, the Coordinator remains active after detecting the busy tone ready to communicate.



(a) Superframe without traffic



(b) Superframe with traffic

Figure 5.2: SoI operation

Using this method, we can obtain important energy saving, especially for low traffic applications. Then, contrary to classic IEEE 802.15.4 protocol, when using Sleep on Idle the energy consumed by coordinators is comparable to the energy consumed by leaf nodes.

5.2.1 Implementation details and discussion

In this section, we discuss the issues related to the implementation of Sleep on Idle. We have implemented the method in Contiki OS [2] based on the implementation of the beacon enabled mode of the IEEE 802.15.4 standard implemented in our group.

a) Protocol timing parameters

Figure 5.3 presents the details of timing parameters of Sleep on Idle. Following, we enumerate them and we discuss an example of possible values.

- *Beacon duration (t_b)*: The time it takes to transmit/receive a beacon. It depends on the data payload size of the beacon.
- *Sensing time (t_{cca})*: The time the coordinator listens to the medium and performs a CCA looking for an RSSI peak.
- *Analog busy tone duration (t_{ab})*: The duration of a transmitted analog busy tone.
- *Back-off slot*: The duration of a back-off slot defined in the IEEE 802.15.4 standard, *i.e.* 320 μs for the 2.4 GHz O-QPSK PHY layer.

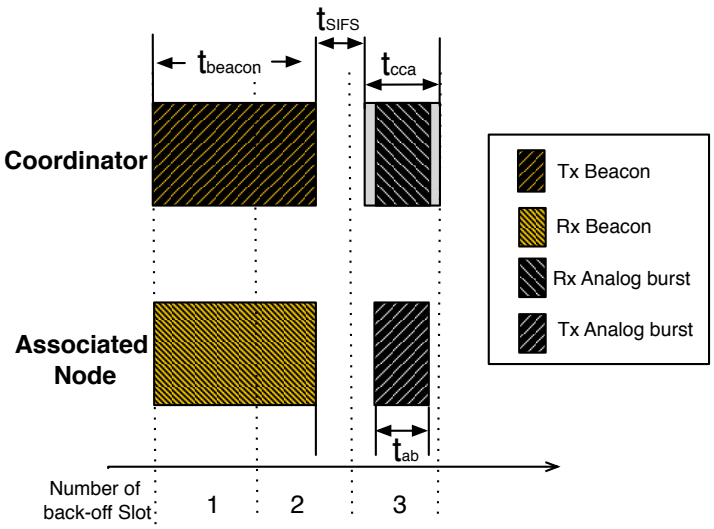


Figure 5.3: SoI timing parameters

The figure places the intervals on a grid of IEEE 802.15.4 back-off slots and shows the shortest possible beacon (13 bytes of MPDU; beacons have a variable length). As the beacon duration in the example is shorter than *aMaxSIFSFrameSize*, thus the busy tone is expected to arrive after the beacon with a time equal to a *macMinSIFSPeriod*. Then, t_{cca} should start *macMinSIFSPeriod* after the beacon transmission (the minimum turnaround time is 192 μs in CC2420). We should choose the duration of the busy tone t_{ab} and the CCA sensing time t_{cca} according to the characteristics of the hardware used for RSSI detection because it needs to have a minimum time required to detect an analog pulse. For instance, the CC2420 chip radio needs to listen to the medium during at least 128 μs to measure a valid RSSI value. With these parameters, Sleep on Idle should end at the boundary of the third back-off slot.

b) Impact of CCA threshold

The CCA threshold has an important impact on the protocol performance. The issue comes from the fact that coordinators detect a busy tone by sensing the medium and looking for an RSSI peak. Since no digital processing is carried out and no information is extracted from the busy tone it is important to distinguish external interference from from a busy tone. The proper CCA threshold must be chosen such that it minimizes:

- *False positives*, or false wake ups. This event leads to an overhead in energy consumption due to the fact that the coordinator may remain awake during the idle Superframes.
- *False negatives*, or missing strobes. This event leads to missing transmissions since the coordinator enters the sleeping state when it does not correctly detect the analog busy tone. The communication during the Superframe will not be possible and packets sent to the coordinator will be lost.

The problem of false positives and false negatives are common to all protocols using LPL and preamble sampling, *i.e.* B-MAC, BoX-MAC and ContikiMAC.

We carried out a simple experience using the WSN430 nodes to assess the influence of the CCA threshold on the number of false positive detections. The optimal threshold value has to distinguish between noise or interference and the reception of the analog busy tone while minimizing the number of false positives. We used the highest possible transmission power (0 dBm) of the CC2420 radio chip. Figure 5.4 shows the rate between the total number of transmitted beacons and the number of detected busy tones, *i.e.* false positives, measured in three cases:

- *No external interference*: In this case a CCA threshold larger than -80dB is enough to correctly detect the busy tones independently from the power of the transmitted busy tone.
- *In presence of* IEEE *802.11 traffic*: We have placed two laptops using IEEE 802.11g and generating saturated (UDP) traffic. We observe that for CCA thresholds greater than -80 dBm, the node experiences almost no false positives. Given the short duration of the busy tone and the back-off times and very short packet duration of IEEE 802.11 the probability of having them overlapping is very low.
- *Generated interference:* We placed four IEEE 802.15.4 nodes sending 1 frame every 400 ms, they do not perform CCA before sending and behave as hidden nodes. We can see that the impact of these transmitters is significant, *i.e.* 20% of false positives. However, as the interferers in our experiments do not perform any CCA nor CSMA/CA, such a situation is not highly probable in real conditions.

These results suggest the introduction of an adaptive CCA threshold mechanism according to interference conditions. Also, the CCA threshold for the busy tone may differ from the corresponding CSMA/CA threshold, *i.e.* frames exchange during the Supeframe. The former must be set to minimize false positives and false negatives, the second must be set to correctly cope with contention and minimize the occurrence of collisions.

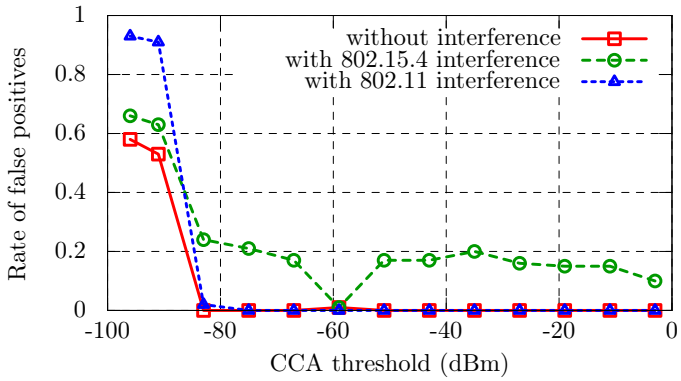


Figure 5.4: Probability of false positives vs. CCA threshold

c) Timing for interference robustness

Further robustness to external interference can be obtained by adding a timer that estimates the duration of the busy tone, hence, if a longer transmission is detected it can be interpreted as external interference.

This mechanism has also been added to WoI, and we have shown it helps to perform better in presence of interference. Also, ContikiMAC proposes a similar approach called *fast sleep* feature [78].

d) Energy budget and reactivity

An IEEE 802.15.4 network with a highly constrained energy budget needs to operate at very low duty cycles that can be achieved by increasing the beacon interval or the sleeping interval of nodes. However, long beacon intervals result in increased latency especially in multihop scenarios. For many applications, low latency is a requirement along with low energy consumption.

Since Sleep on Idle significantly reduces idle listening and duty cycle we can wake up the nodes more often, *i.e.* reduce BO, while the energy budget is respected. Then, when introducing Sleep on Idle, we can improve the reactivity of the network while being efficient in terms on energy consumption.

5.2.2 Evaluation of Sleep on Idle

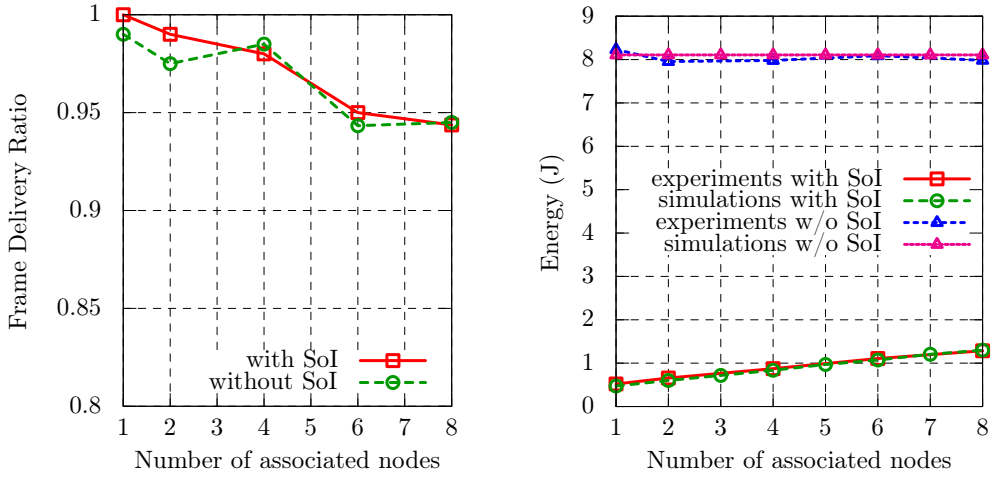
We have evaluated the energy efficiency and performance of Sleep on Idle on a real testbed using Senslab and simulations using WSNet [118]. The implementation is available for WSN430 nodes usign the CC2420 chip radio.

We have performed simulations and real test-bed experiments. We compare the performance of our method in terms of Frame Delivery Ratio (FDR), energy consumption, and delay with the standard IEEE 802.15.4 protocol and a preamble sampling duty cycle MAC: ContikiMAC. We consider two topologies: a star and a cluster tree.

a) Star topology evaluation

In this scenario we have set up a star topology and we have varied the number of nodes associated with the coordinator. Each associated node generates frames based on a Poisson process with an average of 1 packet every 240 seconds, *i.e.* $\lambda = 240$. Each experiment involved 10 measurement sessions of 2 hours each. For these experiments we have chosen BO=8 and SO=3.

Figure 5.5a presents the Frame Delivery Ratio (FDR) in function of the number of associated nodes. We can observe that the FDR is similar for both the standard IEEE 802.15.4 and IEEE 802.15.4 with Sleep on Idle. Hence the introduction of the busy tone does not affect the correct delivery of information.



(a) FDR vs associated nodes. Standard IEEE 802.15.4 vs. Sleep on Idle (star)

(b) Energy (J) vs associated nodes. Standard IEEE 802.15.4 vs. Sleep on Idle (star)

Figure 5.5: Star topology results

We are interested in the energy consumption. We have chosen a relatively low traffic scenario to better observe the energy gain when Sleep on Idle is introduced. When using the standard IEEE 802.15.4, the power consumed by the coordinator is independent of the traffic: it is determined by the values chosen for BO and SO since it remains awake during the whole Superframe duration. In contrast, child nodes can turn off their radio during the Superframe if they do not have frames to exchange with their parent. Thus, in this case, the coordinator is the most power-consuming device and it is desirable to reduce its duty cycle to maximize the network life time.

Figure 5.5b shows energy in Joules (J) consumed by the coordinator in function of the number of associated nodes. It shows results from both simulations and experiments. In the standard IEEE 802.15.4, the energy consumed in reception remains constant regardless of the number of associated nodes. We can explain the constant energy consumption by the fact that the coordinator is active during all the Superframes and the active period does not depend on the traffic nor on the number of associated

nodes.

By introducing Sleep on Idle we were able to save a significant amount of energy: the energy consumption is divided by up to 9 times. As expected, the consumption increases with the number of associated nodes and generated traffic. When traffic increases, *i.e.* one frame is sent in every Superframe, the energy consumption of Sleep on Idle is comparable to the one of IEEE 802.15.4.

b) Cluster-Tree topology evaluation

In this experiment, we have set-up a cluster-tree topology represented in Figure 5.6. The network contains 12 nodes with a maximum of 4 hops. The leaf nodes send traffic towards the PAN coordinator (sink). Traffic is distributed according to a Poisson process with intensity of 1 packet every 240 seconds. We have tested the network of nodes operating according to three MAC protocols: standard IEEE 802.15.4, IEEE 802.15.4 with Sleep on Idle, and ContikiMAC.

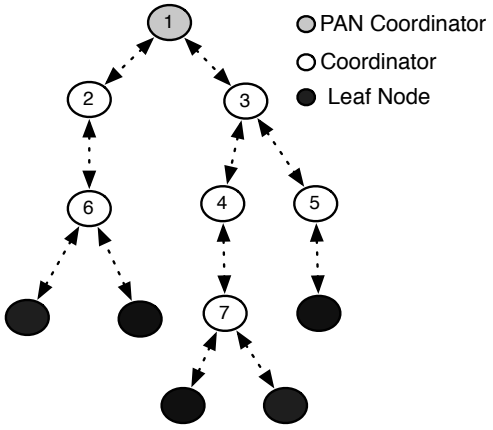


Figure 5.6: Evaluated Cluster-Tree topology

We focus on evaluating uplink traffic towards a sink over a given MAC protocol. Hence, we are not interested in routing mechanisms, neighborhood discovery, or maintenance. Instead, we build a static topology in which node association is created at the beginning of the experiment and remains fixed.

Our biggest interest is to determine the gain in power consumption of adding Sleep on Idle with respect to the standard IEEE 802.15.4. For this, we have varied the Beacon Order of both implementations from 5 to 8. Each experiment runs for 2 hours.

Figure 5.7 plots the average energy consumption in Joules (J) for different BO values. As expected, as we increase BO the energy consumption is reduced. The gain of adding Sleep on Idle reaches significant values: we obtain 10-fold energy savings. Energy savings depends on traffic intensity.

We have also evaluated the packet end-to-end delay presented in Figure 5.8. It can be observed that the delay is the same for both the standard IEEE 802.15.4 and Sleep on Idle for the same BO values. Thus, as expected, the introduction of Sleep on Idle does not affect the frame delivery latency. We can observe that passing from a BO

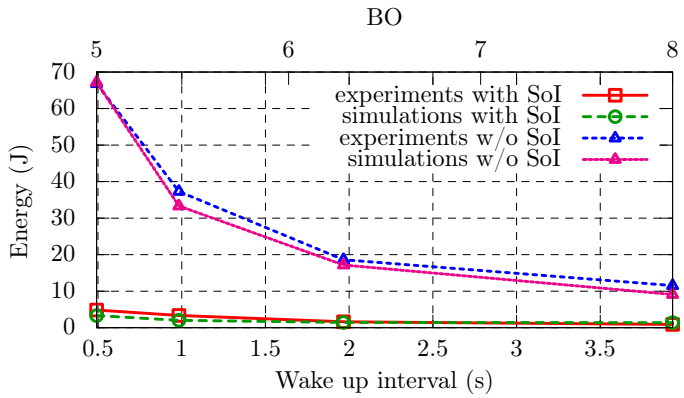


Figure 5.7: Energy (J) vs. Beacon Order. Standard IEEE 802.15.4 vs. Sleep on Idle (Cluster-tree)

equal to 8 to a BO equal to 5 reduces the delay by a factor of almost 10.

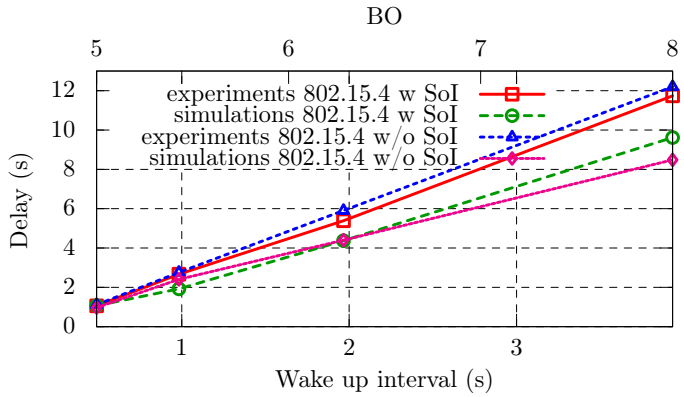


Figure 5.8: Delay (s) vs. Beacon Order. Standard IEEE 802.15.4 vs. Sleep on Idle (Cluster-tree).

Then, low traffic applications with low latency requirements can take profit of these energy savings. Using Sleep on Idle, the frame delivery latency can be reduced by 10 times, compared with classic IEEE 802.15.4 while nodes keep the same (or even less) power consumption. For instance, as observed in FIgure 5.7, a node implementing Sleep on Idle consumes far less energy with a BO equal to 5 than a node implementing classic IEEE 802.15.4 with a BO equal to 10.

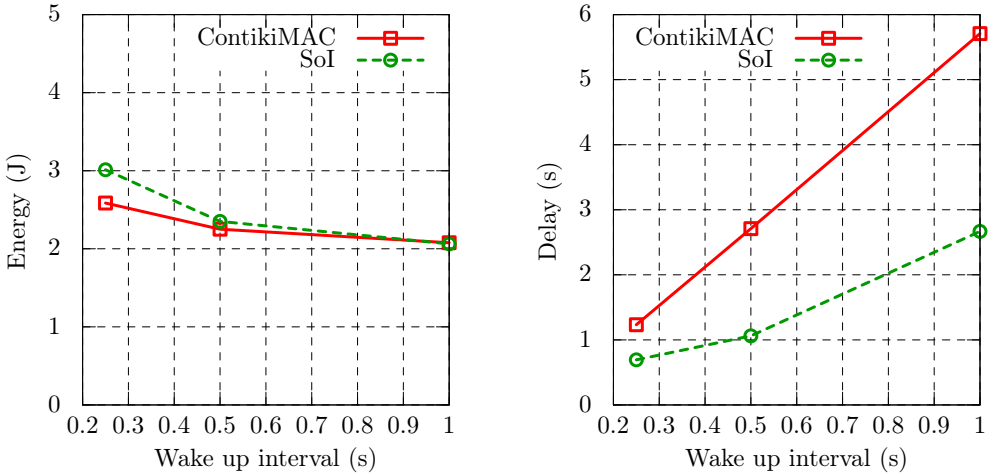
5.2.3 Comparison with ContikiMAC

We compare Sleep on Idle with ContikiMAC. Hence we confront an asynchronous protocol with a synchronous protocol. We have already listed the characteristics and the drawbacks of the different type of protocols; we believe that the medium access

technique must be chosen according to the application and the requirements. We have used the ContikiMAC implementation available in the Contiki OS.

In ContikiMAC, the power consumption, *i.e.* duty cycle, and reactivity are determined by the sampling frequency, *i.e.* the number of wake-up periods per second. We can choose the value according to application requirements. The minimum possible sampling frequency is 1 Hz, i.e. a maximum wake-up period of 1 second. Due to the problem of overlapping Superframes from different coordinators in IEEE 802.15.4 we can reduce the Beacon Interval to a minimum Beacon Interval of 250 ms (BO= 4). We have tested equivalent sampling frequencies in ContikiMAC, *i.e.* from 1 Hz to 4 Hz. A sampling frequency of 4 Hz means a BO of 4 and 1 Hz means a BO of 6.

Figure 5.9a plots the average energy consumed by each node when using ContikiMAC and Sleep on Idle, the experimental scenario, i.e. traffic and topology, is the same as presented above. We can observe that for short sleep cycles Sleep on Idle consumes more than contikiMAC, but as the sleep intervals increase, the energy consumption get closer between the two protocols.



(a) Power (J) vs. sleeping period(s). Sleep on Idle vs. ContikiMAC

(b) Delay (s) vs. sleeping period (s). Sleep on Idle vs. ContikiMAC

Figure 5.9: SoI vs ContikiMAC

We can explain the higher energy consumption on Sleep on Idle by the fact IEEE 802.15.4 is a synchronous protocol where the network is maintained by beacon transmissions. Hence nodes remain synchronized even without any traffic. In ContikiMAC, phase lock synchronization may be lost when no data is exchanged for long periods of time. Then resynchronization must be performed once traffic is generated; then, at the beginning of communication we observe an increase of duty cycle and power consumption. When using an asynchronous (hybrid) duty cycle MAC, such as ContikiMAC, topology control must be performed by higher layers with additional maintenance costs, which are not accounted for in this study.

Figure 5.9b shows that Sleep on Idle behaves better in terms of delay for sleep intervals longer than 250 ms. This can be explained by the fact that nodes are synchronized

and no wake up mechanism has to be implemented. Furthermore, as it is a synchronous protocol, Sleep on Idle achieves very long sleeping periods, *i.e.* up to 4 minutes. Then, very low duty cycles and further energy savings at the cost of longer delays can be obtained. Applications that have relaxed latency constraints can reach very long life of batteries.

5.2.4 Simulation of large-scale topologies

To evaluate the performance of Sleep on Idle on topologies with a larger number of nodes, we have used WSNet simulations of a cluster tree topology with 100 nodes. As previously, nodes send packets to the sink. BO is set to 8 and SO is set to 3. Figure 5.10 presents the average energy consumption per node obtained from simulations for a varying inter-packet interval.

For both IEEE 802.15.4 and Sleep on Idle we have observed a FDR of 100%. The figure shows that even in a large-scale topology, Sleep on Idle achieves energy savings up to 80% with respect to IEEE 802.15.4 without affecting the FDR.

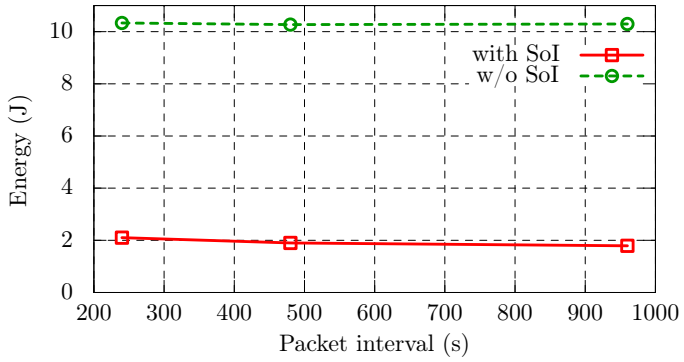


Figure 5.10: Energy consumption vs. interpacket interval. Standard IEEE 802.15.4 vs. Sleep on Idle.

5.3 Discussion and comparison with A-MAC

Similarly to Sleep on Idle, A-MAC proposes a mechanism to inform packet recipients about the presence of pending packets. A-MAC is a receiver-initiated protocol based on LPP (Low Power Probing), where all nodes transmit a probe, *i.e.* beacon, at the beginning of active periods. As the IEEE 802.15.4 protocol, nodes inform their neighbors that they are ready to communicate. A-MAC is an asynchronous (hybrid) protocol, hence nodes listen for beacons or probes only if there is data to be sent and potential receivers announce their availability through the transmission of beacons.

Figure 5.11 illustrates how the exchange of data is done in A-MAC. When a node has a packet to transmit, it listens for a probe sent by the packet recipient. A soon as the probe is received, the transmitter node sends a frame (or acknowledgement) to inform the sender of the probe that it has a packet to send. Nodes use phase locking

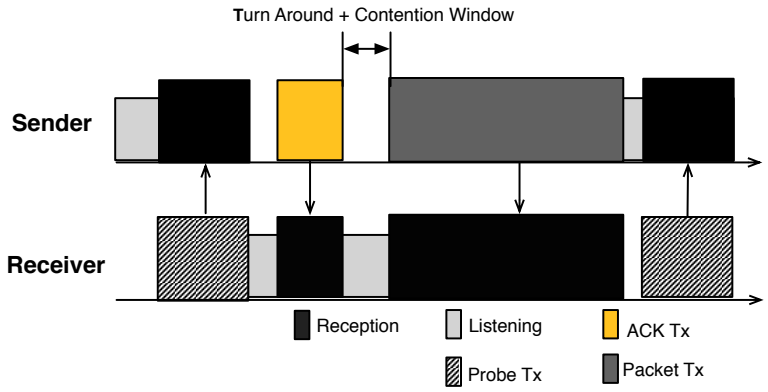


Figure 5.11: Packet exchange example in A-MAC

to optimize duty cycle. Thus, a node entering the active state can decide if it remains active or not, according to a pre-established signalization procedure.

The design of A-MAC is another example of the advantages of having a radio chip able to take decisions without the intervention of the CPU or any software mechanism. It also introduces a signaling mechanism that helps reducing idle listening. Given the similarities of A-MAC with Sleep on Idle, we dedicate this section to discuss about the implementation of A-MAC and its advantages and disadvantages with respect to Sleep on Idle.

5.3.1 The notification mechanism

Senders in A-MAC use acknowledgments frames to notify the receiver about data exchange. Hence, acknowledgement frames are modulated frames that need to be decoded and analyzed.

Hardware modules available in some existing radio chips, such as the CC2420, generate acknowledgement frames fast and efficiently since it does not require any processing of frames at the software level. This acknowledgement mechanism is based on address filtering where the radio is able to determine if the node is the destination of the received frame and, if requested, it automatically generates and transmits an acknowledgement frame. Acknowledgements are thought for unicast frames and probes have a broadcast nature. In order to exploit hardware acknowledgements, A-MAC proposes to tweak nodes addresses in such a way that broadcast probes are seen as unicast frames for nodes wanting to communicate with the generator of the probe. Then, when a probe acknowledgement is received by the transmitter of the probe, it remains active to communicate. Hardware address manipulation becomes complex and tricky, the idea of acknowledging broadcast frames is not compliant with any standard or communication protocol.

A question that arises here is what happens if two or more nodes have a pending frame for the same destination: the destination sends its probe and it will be acknowledged by more than one node and acknowledgement frames will collide. Will the destination be able to decode the acknowledgement frame? as acknowledgements are generated in hardware, non-deterministic delays introduced by software or the ac-

cess to the radio are eliminated, meaning that nodes generate and transmit identical acknowledgement frames at exactly (or almost) the same instant. Hence acknowledgement collisions are not harmful [106] as long as they are identical. Even better, they generate constructive interference and the acknowledgement frame can be correctly decoded. Then, after the acknowledgement is received, the destination remains active and the senders attempt to transmit their frame using a contention-based mechanism.

No much details are given about the contention mechanism used during the active period; the receiver node introduces into its probe the value of the contention window to be used; then, sender nodes will compute a back-off time based on the contention window value. Nodes only attempt to send a frame after the reception of a probe; if they fail, they have to wait for another probe that can happen during the current active period or during the next active period of the receiver node. Some details are missing about the strategies adopted when specific cases arrive. For example when there are several consecutive CCA failures or when multiple (unsuccessful) retransmissions have been performed.

The principle of Sleep on Idle is much simpler; frames do not need to be decoded and there is not need of manipulating hardware addresses and "bending" standards. One, two or several nodes can send their busy tone without worrying about collisions since the coordinator only checks the RSSI level. Thanks to synchronization (*i.e.* common active periods between coordinator and associated nodes) Sleep on Idle guarantees that the generator of the busy tone of a given (known) duration is an associated node; hence, the probability of having a false positive is low while keeping simplicity. Another interesting characteristic of Sleep on Idle is that its implementation remains completely compatible with the IEEE 802.15.4 standard. Hence, there are no ambiguities about its implementation and coexistence with already deployed networks is possible.

5.3.2 Synchronous vs asynchronous communication protocol

A-MAC is a "hybrid" protocol. Phase lock with known neighbors is maintained only if there is constant traffic; nodes need to wake up often to send their probe and its performance is highly affected due to contention (*i.e.* collisions with probes, schedule conflicts, hidden terminals and so on).

Sleep on Idle is based on a synchronous mechanism and on a well-defined standard, extremely low duty cycles can be reached due to synchronization and the use of common active periods.

5.4 Summary and Conclusions

In this chapter we have presented Sleep on Idle. The main idea of Sleep on idle is to minimize the idle listening of contention based medium access protocols. By using simple analog signaling, Sleep on Idle is able to significantly reduce idle listening specifically for low traffic applications. As a proof of concept we have applied the Sleep on Idle principle to the slotted IEEE 802.15.4 standard and we have evaluated the method on a real platform and simulations. We have demonstrated that it reduces energy consumption by an order of magnitude that depends on the traffic. We have also discussed the parameters to have a working implementation using existing hardware

platforms and we have evaluated the robustness of the approach to external interference and noise. Moreover, we have shown that Sleep on Idle can be used for applications that require high reactivity keeping energy consumption much lower than standard IEEE 802.15.4.

The principle of Sleep on Idle is based on the use of low level information provided by the radio chip. The working principle of the protocol is very simple and it could also be implemented in the radio chip as a smart radio able to take decisions about whether the transceiver and the main processing unit must be activated.

Part III

Conclusions and additional discussions

Chapter 6

On the Importance of Real Experimentation in Wireless Sensor Networks

Contents

6.1 Motivation and introduction

Experimental work in wireless networking, specifically in Wireless Sensor Networks, is mostly based on simulations results rather than real experimentations and implementations. Carrying out real experimentation on these kinds of distributed systems can be a highly demanding task; a proper infrastructure and a framework for debugging and data collection are needed. Also, there is a huge set of parameters (either controllable or not) that needs to be taken into account before arriving to a valid conclusion.

Additionally, when implementing a protocol in a real platform there are some design aspects and constraints that need to be faced in order to overcome hardware limitations. For instance, according to hardware constraints (*i.e.* available memory, computation speed, data width) and requirements of the implementation (*i.e.* implementation footprint, timing constraints, complexity), the designer has to decide whether an operating system has to be used and, if this is the case, a suitable operating system has to be chosen. Proper tools for compiling and debugging have to be selected and a debugging methodology and strategy has to be adopted.

During the different experiments carried out on this thesis work, we have encountered several challenges. Therefore, in this chapter we aim at summarizing a set of observations and recommendations that may ease experimentation; introducing a proper

methodology may help as well to reproduce a given experiment. Also, we briefly describe the architecture of the SensLAB platform and we talk about how to facilitate experimenting with the platform.

We go through two cases of study, which require real experimentation, and we summarize the conclusions taken from these procedures. Particularly we discuss the importance of knowing the hardware platform, how experiments depend on the different implementation layers and the difficulty of reproducing an experiment and generalizing the results.

6.2 Experimenting in Wireless Sensor Networks

Let's take al look to some of the aspects to account for when experimenting:

- *Scenario definition:* Conclusive results can only be obtained by properly defining the evaluated scenario. This practice allows fair comparison with the results from different runs of the same experiment, from previous related work, and from simulations. By the scenario we mean the different parameters that may affect the functionality of the network; examples of scenario parameters could be:
  - Number of nodes in the network, their operation (*i.e* sink, router, leaf, sniffer, interferer, etc), and their position;
  - Communication protocols being used;
  - Radio transmission information (*i.e.* modulation, data rate, transmission power);
  - Traffic pattern;
  - Environmental characteristics (*i.e.* indoor/outdoor scenario, possible interference, geometry of the room, etc).
- *Hardware platform selection:* The characteristics of the node's hardware platform greatly affect the performance of the feature being evaluated, the way results can be analyzed and the behavior of the network. To correctly set up an experiment it is vital to have knowledge about the hardware resources and limitations.
  - Available node's memory (*i.e.* MCU RAM and Flash, external memory);
  - Radio chip characteristics (*i.e.* data rate, data and configuration access mode, buffer's size, packet hardware support, possible configuration parameters and available information about the radio link, standard compliant or proprietary communication protocol, and so on);
  - MCU characteristics (*i.e.* bus and operations width (16-bits, 32-bits), low power modes, nominal operating frequency, available peripherals: GPIOs and external communication modules, timers, ADCs, sensors);
  - Debugging possibilities (*i.e.* JTAG, UART, LEDs, display).
- *Software stack selection:* By the software stack we mean the operating system, the protocols used at the different layers, the communication with the debugging tool

and the application being evaluated. The choice of such software implementation may impact the whole network performance. For instance, the way the operating system manages events and tasks, and the addition of traces for debugging have an effect on the protocols' behavior.

- *Choice of performance metrics:* A set of appropriate metrics must be selected in order to assess the performance of the protocol (or feature) being evaluated. Such metrics depend on the constraints and requirements of the evaluated scenario. These metrics could be: *energy consumption*, *latency*, *throughput*, and *scalability*.

- *Tools selection:* The choice and set-up of a proper developing, compilation, programming and debugging environment is greatly important to optimize and simplify the network deployment process. Thus, it is vital to evaluate all the possibilities and choose them according to the resources and requirements for experimentation.

There is a huge set of parameters that cause the experimenting and debugging process to be very complex. A benchmarking methodology, as the one presented by Corbett *et al.* [119], could ease the process of evaluating and comparing different features and protocols in a Wireless Sensor Network.

To facilitate experimentation, some platforms, such as SensLAB, allow accessing a set of nodes remotely. They offer as well a set of tools and software stacks. However, debugging could become a tedious task since nodes cannot be accessed locally and they cannot be monitored using simple peripherals such as LEDs. Also, as the environmental conditions cannot be monitored nor controlled it becomes impractical for some experiments. Debugging can be done only using serial communication; then, one must add traces or "printf" calls into the code. Such tracing mechanism can be invasive and affect the behavior of the system.

Testing functionality at the node level locally is an easy task, we only need an interface for debugging and we can use LEDs, serial communication and displays to observe the behavior of the node; in the same way, testing a small network is simple since not much infrastructure is needed and we can monitor all nodes in the system. As the network scale increases, local debugging becomes difficult and requires a lot of resources and a proper infrastructure. This is why most of the studies are based on simulations for large scale evaluation and, sometimes, a small network is evaluated in a real testbeb.

6.3 SensLAB: a platform for real testbed experimentation

6.3.1 Platform architecture description

SensLAB deploys a very large scale open wireless sensor network platform. It is deployed into four different sites in France (Strasbourg, Rennes, Lille and Grenoble); there are around 250 nodes in each platform. It uses the WSN430 nodes which integrate an MSP430 microcontroller with a radio chip from Texas Instruments (*i.e.* CC1101 or CC2420).

The main goal of SensLAB is offering a tool for design, development, tuning and experimentation of real large-scale sensor network applications. The SensLAB architecture offers a framework to manage available nodes and access the information they log through the serial port.

The architecture of SensLAB is illustrated in figure 6.1. Basically, the platform is composed of the *master* server and the *site* servers. The master server is used to manage the experiments defined by the user: experiment control (*i.e.* nodes' selection, start time, stop time), profiles management and access to user account information. The site severs are located on each site; these servers allow direct access to the nodes (*i.e.* flashing, resetting, serial communication access) and they provide an ssh server as well as tools for firmware compilation and the possibility to log some measurements such as temperature and voltage. Information written by the nodes through the serial port can be read through a NetCat connection. Since the only debugging mechanism in SensLAB is the serial port connection, when debugging the network, one terminal and one connection has to be opened per node of interest.

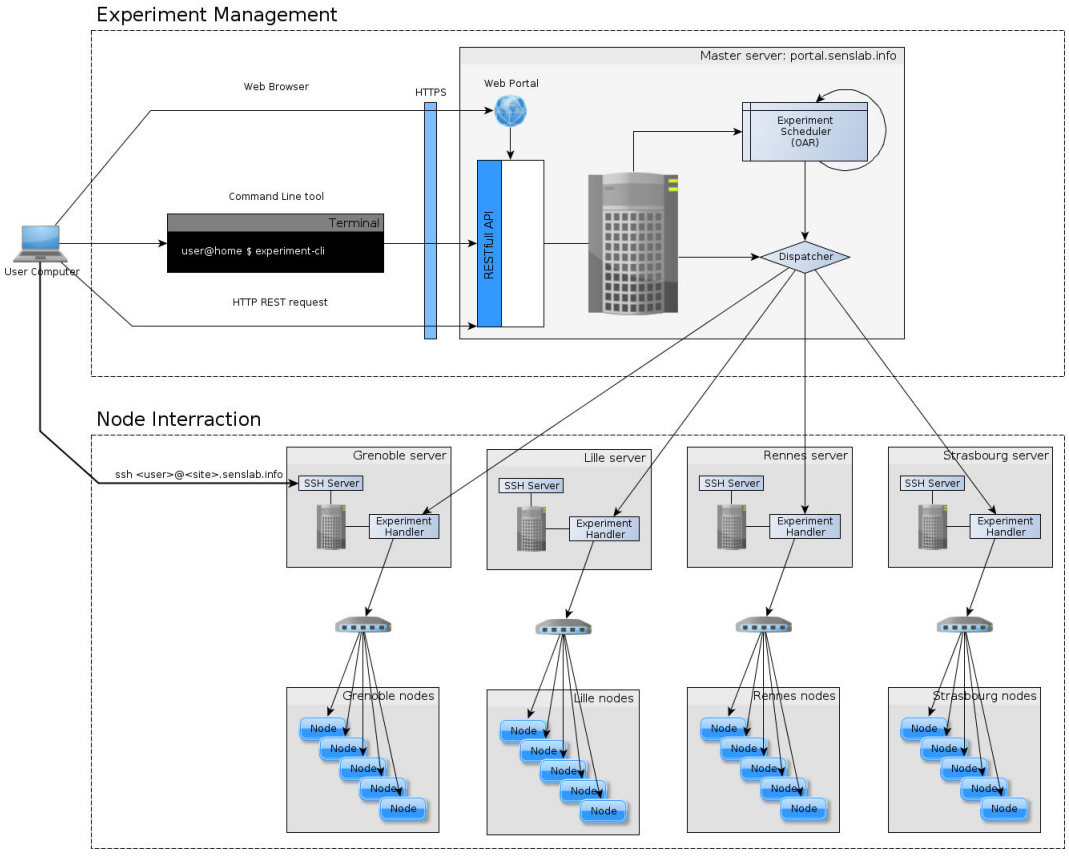


Figure 6.1: Architecture of the SensLAB platform (taken from [120])

6.3.2 Experimenting with the SensLAB platform

As the number of nodes in the network increases, manual debugging in the platform may become an annoying task: a lot of terminals have to be opened. Then, automation of the debugging process in these kind of platforms can bring several benefits: it reduces the complexity of the procedure as well as the evaluation and the final deployment time.

During this thesis work we have observed the importance of having such a debugging methodology in SensLAB; we have then developed a framework that could facilitate extensive experimentation. The framework has been written in *Python* and it is executed in the user environment where the nodes selected for experimentation can be accessed. The idea of the framework is the implementation of a set of directives for:

- *Definition and execution of the desired test-bed scenario*: i) the number of nodes and their identifiers, ii) the firmware(s) for node's flashing, iii) information for data logging, iv) the duration of the experiment, v) the events that may occur during the experimentation.

- *Automatic management of nodes*: *i.e.* flashing and resetting, and opening NetCat connections and *Automatic logging of information* sent by nodes through the serial line and that can be used for off-line analysis.

- *Execution of computational expensive tasks at the site server side*: this provides centralized control over the nodes, reduces the complexity of the firmware and overcomes the resources constraints imposed by the nodes. For example, complex functions that may take longer than serial port communication or that do not fit into the microcontroller, *i.e.* traffic generation based on a statistical function. Such process can be used to *generate events* and *analyze logged data* while the scenario is running.

- *Observing the scenario based on a global time:* Since all nodes are connected to the site server, there is a notion of global time, this could facilitate protocol performance evaluation, *i.e.* end to end packet delay, detection of events.

6.4 Cases of study

To illustrate some of the aspects we have discussed before, we will describe two real experiments, we will discuss the limitations faced while carrying out the experiments and the conclusions that may help to facilitate future procedures.

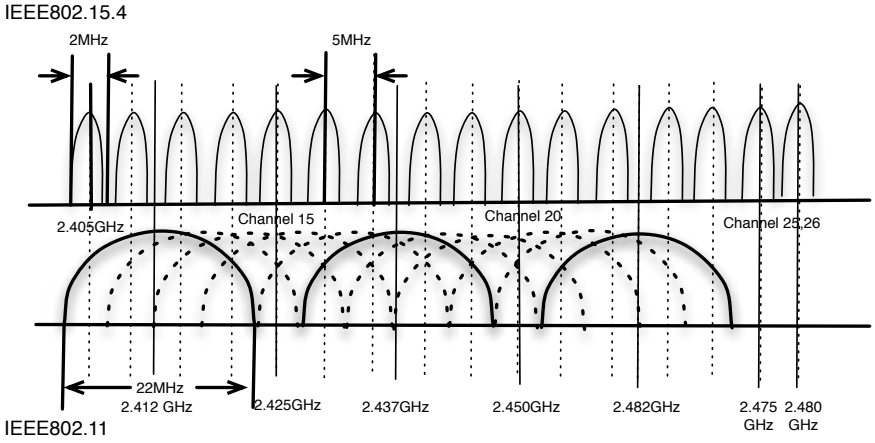
6.4.1 The problem of reproducibility: Study of the IEEE 802.11 - IEEE 802.15.4 coexistence problem

In this case of study we experiment around the coexistence of IEEE 802.15.4 and IEEE 802.11 networks. Before a proper solution to the problem can be proposed, the effects of inter-standard interference must be identified as well as the sources of errors and performance degradation. Given the huge number of parameters such study requires real experimentations.

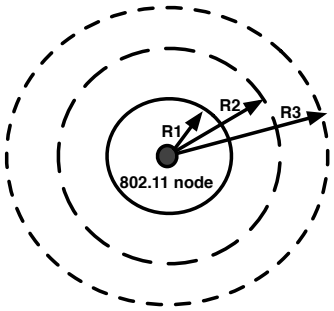
Thus, we present the related work and different experimentations carried out to determine the effects of IEEE 802.15.4/IEEE 802.11 coexistence; we summarize results and conclusions from these studies. We attempt to reproduce some of the existing work: we perform timing simulations and we experiment using different hardware platforms; then, we compare our results with those presented by the literature.

a) Background:

IEEE 802.15.4 and IEEE 802.11 may use overlapping channels for communication (*i.e.* in the 2.4 GHz frequency band), from Figure 6.2a it can be deduced that few IEEE 802.15.4 channels are completely free from IEEE 802.11 interference. The probability of having both IEEE 802.15.4 and IEEE 802.11 networks coexisting in the same medium and frequency band is very high; then, the study of the real effects of such interference becomes very important.



(a) Overlapping IEEE 802.15.4 and IEEE 802.11 channels



(b) The IEEE 802.15.4-IEEE 802.11 interference regions

Figure 6.2: IEEE 802.15.4 and IEEE 802.11 coexistence

Given the difference in transmission power and channel bandwidth between both standards, IEEE 802.15.4 side transmissions can be more affected than IEEE 802.11 transmissions. Parameters such as CCA threshold and mode, transmission power, fre-

quency offset with respect to the IEEE 802.11 network and distance have an important impact in the communication performance of IEEE 802.15.4 in presence of IEEE 802.11 transmissions.

According to the distance or the relative power between both networks three interference regions can be found (Figure 6.2b): i)*Region 1:* Nodes from both IEEE 802.15.4 and IEEE 802.11 can sense each other, hence IEEE 802.15.4 defers in presence of IEEE 802.11 transmissions and vice-versa. ii)*Region 2:* IEEE 802.15.4 nodes sense IEEE 802.11 nodes but IEEE 802.11 nodes do not sense IEEE 802.15.4 transmissions. Hence collisions may damage IEEE 802.15.4 transmissions. iii)*Region 3:* Nodes from both networks do not sense each other; interference may still exist and packets may be damaged.

b) Related work and results:

The impact on the performance of IEEE 802.15.4 communication varies according to the interference region. There are a lot of experimentations related to the coexistence problem and there is not a consensus about the results. The main issue is the huge number of parameters that may affect the communication performance and the number of variables that need to be monitored. These studies agree that the parameters that influence the results are, among others:

1. The *CCA threshold and mode*, especially at the IEEE 802.15.4 side. Several experiments have shown the influence of this value [121, 122, 123, 124, 125]. The ED (Energy Detection) mode allows detecting transmissions generated IEEE 802.11 nodes. Also, during the CSMA/CA procedure, after a given number of CCA failures (*i.e.* medium busy before starting a transmission) transmitters discard their packets. On one hand, if the CCA threshold is reduced the CSMA/CA mechanism is more sensible to IEEE 802.11 transmissions and the probability of discarding the packets is increased. On the other hand, if the CCA threshold is increased the probability of colliding with IEEE 802.11 transmissions increases.

   Then, according to the IEEE 802.11 traffic and power characteristics, a proper CCA threshold must be chosen. Yuan *et al.* [126] propose an adaptive CCA threshold mechanism. In fact, in presence of low IEEE 802.11 interference, IEEE 802.15.4 communication could still be possible; collisions may not be harmful and packets can be decoded. Such reactive mechanism requires a proper IEEE 802.11 interference estimator to take decisions regarding the CCA threshold value. Regarding the energy consumption, even if it is not studied by the authors, there is clearly a trade off between the energy consumed when several CCAs have to be performed or when transmitting a packet that could collide and be damaged.

2. The *transmission power* at both sides as well as the *distance* between IEEE 802.11 transmitters/receivers and IEEE 802.15.4 transmitter/receivers. In fact, the distance between nodes determines the power of transmissions that arrive to these nodes. Regarding the distance, authors have attempted to provide a set of recommendations to minimize the effects of IEEE 802.11 interference on IEEE 802.15.4 transmissions. These recommendations are based on real experimentations and they differ from each other. Results in terms of packet error rate according to the

distance between nodes are different for each study [127, 125, 128, 129, 130]; moreover, these experiments take into account statistics at the receiver side and not at the sender side. Petrova *et al.* [124] have shown the influence of the *orientation* of IEEE 802.11 transmissions relative to IEEE 802.15.4 transmissions.

3. The *frequency offset* of channels being used for both IEEE 802.15.4 and IEEE 802.11 transmissions. Given the frequency spectrum of transmissions, channels that do not overlap but that are adjacent can still interfere and it affects performance of the CSMA/CA mechanisms and damages transmissions. Authors recommend values to avoid interference regarding the frequency channel to be selected; from the different experiments [131, 132] values between 10 MHz and 30 MHz are recommended. Petrova *et al.* [124] have shown that IEEE 802.11n requires larger frequency offset since it uses wider channels and interferes more with IEEE 802.15.4 channels.

   Finding a channel that is completely free from interference may become difficult given the popularity of IEEE 802.11 networks. Then, multi-channel and frequency hopping solutions [133, 134, 135, 136] can be promising when coping with IEEE 802.11 interference.

4. The *packet size* influences the collision probability [124]. In fact, given the timing differences between both standards, shorter packets may have more chances to be transmitted and correctly received at the destination during IEEE 802.11 idle times.

5. The IEEE *802.11 traffic* and the IEEE 802.11 standard (b,g,n). Several studies have shown the impact of IEEE 802.11 traffic (or duty cycle) on IEEE 802.15.4 transmissions [130]. As expected, higher is the traffic higher is the loss rate at the IEEE 802.15.4 side; this could be due to inhibition at the IEEE 802.15.4 transmitter side (CCA failed) or due to collisions that damage packets.

   Most of the studies conclude that the effects of IEEE 802.11 interference depend on the standard. However, there is not a consensus here. Some authors [130] claim that IEEE 802.11g transmissions have a more significant impact than IEEE 802.11b transmissions. In fact, given the shorter idle times (back-off times, SIFS, LIFS, CCA periods) intuitively one could think that IEEE 802.11g transmissions have a higher probability to collide with IEEE 802.15.4 transmissions.

   Lets analyze the probability of finding the medium free in presence of saturated UDP IEEE 802.11 traffic (*i.e.* the IEEE 802.11 contention window value remains unchanged). The IEEE 802.15.4 station is able to access the medium and start a transmission only if it finds the medium free during the CCA sensing time; hence, IEEE 802.11 stations must be idle during this time. Then, the IEEE 802.15.4 station finds the medium free if the following condition is guaranteed:

$$DIFS + N_{BO}T_{slot} \geq CCA_{time} \tag{6.1}$$

   Where DIFS is the time the IEEE 802.11 station defers after a transmission, N_{BO} is the IEEE 802.11 back-off value randomly chosen according to the contention window, T_{slot} is the duration of a IEEE 802.11 time slot and CCA_{time} is the

duration of an IEEE 802.15.4 CCA check. For instance, for IEEE 802.11b the IEEE 802.15.4 station can find the medium free if:

$$50\mu s + (N_{BO} \times 20\mu s) \geq 128\mu s \tag{6.2}$$

Hence,

$$N_{BO} \geq 4 \tag{6.3}$$

N_{BO} is uniformly distributed in the interval [0,CW_{min}], where CW_{min} is the minimum contention window in IEEE 802.11 and equal to 31. Then, the probability of finding the medium free in presence of IEEE 802.11b traffic is given by:

$$P(CCA_{passed}) = P(N_{BO} \geq 4) = \frac{CW_{min} - 4}{CW_{min} + 1} = 0.8437 \tag{6.4}$$

Similarly, we can compute the probability of finding the medium free in presence of IEEE 802.11g. In this case N_{BO} must be at least 12 and the probability is given by:

$$P(CCA_{passed}) = P(N_{BO} \geq 12) = \frac{CW_{min} - 12}{CW_{min} + 1} = 0.1875 \tag{6.5}$$

Notice that this is not the probability of correctly transmitting a packet; indeed, the medium can be accessed but packets may not be correctly transmitted or received. Then, according to the analytical results, the probability of finding the medium free is much lower for IEEE 802.11g interference than for IEEE 802.11b interference.

However, experimental studies have shown that IEEE 802.11b transmissions have a more visible impact on IEEE 802.15.4 communication performance. Authors explain the phenomenon by the fact that IEEE 802.11b transmissions take longer than IEEE 802.11g transmissions; this explanation is true if the same traffic (*i.e.* number of packets per second) is generated at the IEEE 802.11 side. But for saturated traffic, the chance to find the medium idle for IEEE 802.15.4 transmissions is lower for IEEE 802.11g than for IEEE 802.11b. Mao *et al.* [129] have studied the effects of different IEEE 802.11 modulation schemes on IEEE 802.15.4 transmissions. They have found that DSSS based modulation and encoding methods (used in IEEE 802.11b) have a more important impact on IEEE 802.15.4 transmissions than OFDM transmissions (used in IEEE 802.11g).

Additionally, some studies have shown that the (IEEE 802.11 and IEEE 802.15.4) hardware platforms may have an influence in the experiment results [130]. Experimental studies that use frame acknowledgements and retransmissions (*i.e.* they use the ZigBee stack) [137, 131] have shown an impact on packet end-to-end delay; however they do not quantify the number of retransmissions and CCAs performed to correctly deliver a packet. Such statistics could give information about the energy consumption overhead caused by IEEE 802.11 transmissions.

Then, we consider that valid conclusions must be evaluated using different platforms. However most of the proposed solutions have been evaluated on a very specific scenario, using specific hardware/software stacks and results cannot be generalized.

c) Evaluating coexistence

We have experimented on the coexistence problem. In fact, despite the problem has been widely studied, and it has been accepted that IEEE 802.11 interference significantly reduces the performance of an IEEE 802.15.4 based Wireless Sensor Network, there is not convincing solution. One of the applications of flexible radios and the use of reconfigurable hardware can be the introduction of a hardware module able to provide coexistence between different protocols. Such a module could communicate with nodes belonging to different networks and coordinate transmissions in order to guarantee fair access to the medium.

We have looked closely to the work presented by Liang *et al.* [128] and we have based our study on the two regions defined by the authors: the *symmetric* region and the *asymmetric* region. These regions correspond respectively to the regions 1 and 2 illustrated in Figure 6.2b. We have started our study by performing some timing simulations. In the simulations we suppose an ideal channel and we study the behavior of packet transmissions based on timing values (*i.e.* standard defined intervals such as LIFS, SIFS and back-off slots, packets transmission time, and so on).

Then, we conduct experiments using two different motes, that use the same radio chip (CC2420): the *SunSpots* and the *WSN430*. During the experiments we have varied some parameters such as the traffic intensity at the IEEE 802.11 side (*i.e.* the interference level), the CCA threshold of IEEE 802.15.4 nodes and distance between the different stations.

- **Timing simulations**

  We have simulated coexistence between the unslotted CSMA/CA medium access protocol defined by the IEEE 802.15.4 standard and two IEEE 802.11 standards: IEEE 802.11b and IEEE 802.11g. In the symmetric and the asymmetric regions packet loss can be due to:

  - Collisions with other IEEE 802.15.4 transmissions;
  - Collisions with IEEE 802.11 transmissions;
  - Packets discarded due to channel access failure (*i.e.* consecutive CCA failed);
  - Other causes such as attenuation and channel conditions.

  The figure 6.3 illustrates the simulated scenario. With the purpose of determining the effect of IEEE 802.11 transmissions we have simulated only one pair of IEEE 802.15.4 nodes exchanging packets in the presence of a pair of IEEE 802.11 nodes exchanging UDP packets (*i.e.* the contention window value at the IEEE 802.11 side is kept constant). Hence, we analyze only the number of packets that are lost due to collisions with IEEE 802.11 transmissions and due to channel access failures.

  The figures 6.4 and 6.5 show the distribution of packet errors on each region according to the intensity of IEEE 802.11 traffic, simulations were carried out for both IEEE 802.11b and IEEE 802.11g. We have evaluated a saturated scenario, where the IEEE 802.11 station has always a packet to send. Then we have varied

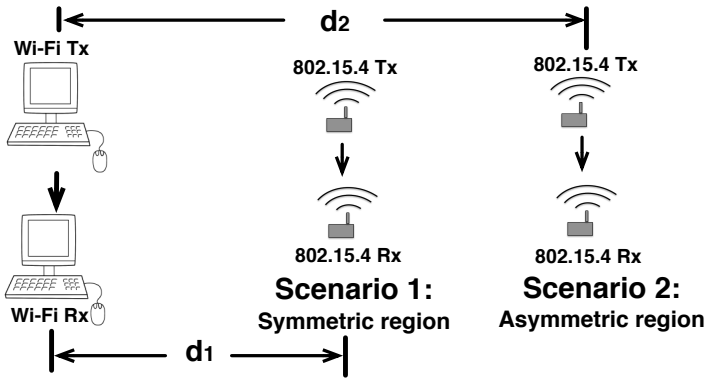
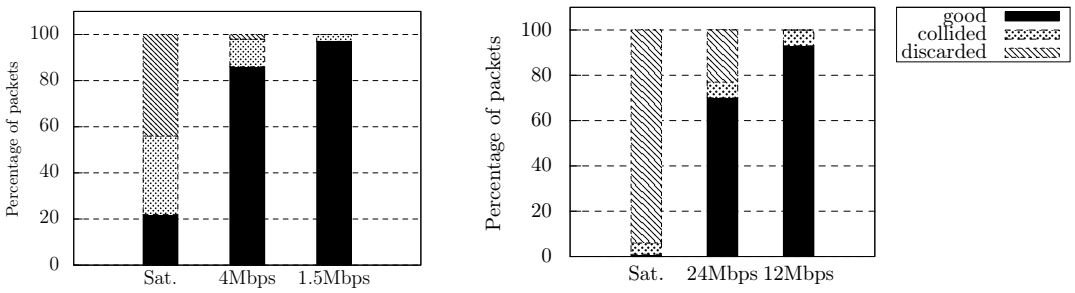


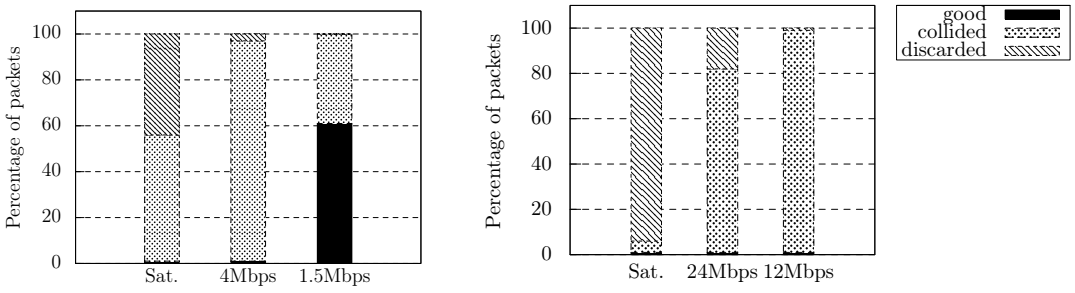
Figure 6.3: Simulated scenario for coexistence evaluation

the intensity of traffic (*i.e.* medium and low) according to the standard data rate. At the IEEE 802.15.4 side we have generated a packet every 75ms. IEEE 802.11 packets have a length of 1400 bytes and IEEE 802.15.4 have a length of 128 bytes.



(a) IEEE 802.11 b **(b)** IEEE 802.11 g

Figure 6.4: Simulations: results in the symmetric region



(a) IEEE 802.11-b **(b)** IEEE 802.11-g

Figure 6.5: Simulations: results in the asymmetric region

From these results we can conclude the following:

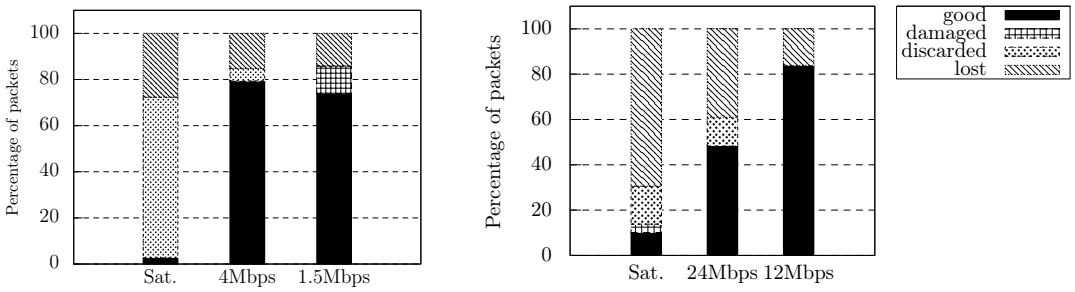
1. *For saturated traffic,* IEEE *802.11g transmission affects more* IEEE *802.15.4 transmissions than* IEEE *802.11b transmissions.* As discussed before, this is due to shorter idle times. Then, in the symmetric region, IEEE 802.15.4 nodes are very likely to find the medium busy and the probability that they discard the packet is very high. In the asymmetric region, when the IEEE 802.15.4 finds the medium free, it is very likely that its transmission is damaged by a IEEE 802.11 transmission.
2. *In both regions a significant fraction of packets is lost due to medium access failure when* IEEE *802.11 traffic is saturated.* The number of discarded packets is more important for IEEE 802.11g interference.
3. *In the asymmetric region is very likely that sent packets are damaged by* IEEE *802.11 transmissions*, since IEEE 802.15.4 data rate is much lower than IEEE 802.11 data rate and IEEE 802.15.4 transmissions may take longer than IEEE 802.11 back-off times.
4. *For moderate or low* IEEE *802.11 traffic, using same packet generation rate,* IEEE *802.11b transmissions may have a bigger impact over* IEEE *802.15.4 transmissions than* IEEE *802.11g* . This is due to the fact that IEEE 802.11g transmissions are faster, and occupy the medium during less time.

- **Experiments with the SunSPOTs**

  Now lets take a look at the experiment results using the SunsSPOTs [7] nodes. We have used two computers exchanging UDP packets using the Ipmt [138] tool, the traffic generator uses an Intel Wi-Fi Link 5100AGN wireless card. At the IEEE 802.15.4 transmitter side we have logged information about the number of transmitted packets and the number of discarded packets. At the IEEE 802.15.4 receiver side we have disabled CRC discarding functionality in order to log both correct and damaged (*i.e.* packets that failed the CRC check) received packets.

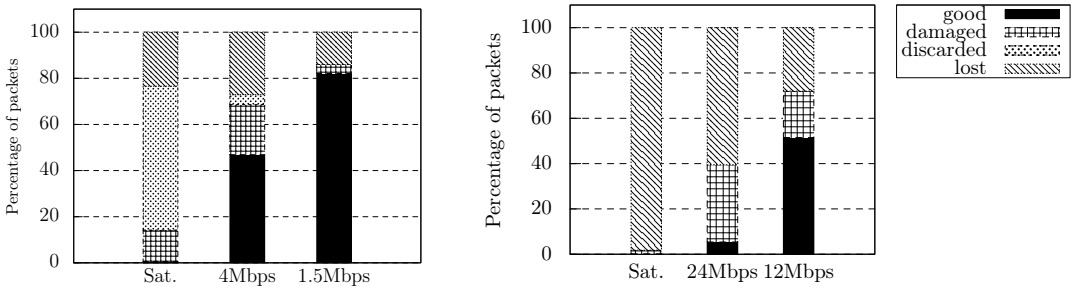
  The SunSPOTs run a lightweight java virtual machine called "Squawk" that uses standard JME packages. The SunSPOTs can be easily manipulated without having any knowledge about the device architecture. There is an implementation of the unslotted IEEE 802.15.4 CSMA/CA mechanism and we have used this implementation during our tests. The experiments were done indoor, and, for preliminary results, we have varied the distance between the IEEE 802.11 and IEEE 802.15.4 stations (*i.e.* 50 cm for the symmetric region and 10 m for the asymmetric region. We have left both CCA threshold and transmission power at their default value (*i.e.* $-77\,dBm$ and $0\,dBm$ respectively). The CCA mode is set to Energy Detection (ED) in such way that IEEE 802.11 transmissions are seen as transmissions occupying the medium. For IEEE 802.11 transmissions we have used channel 6 and for IEEE 802.15.4 transmissions we have used channel 11; in this case the IEEE 802.11 channel overlaps completely the IEEE 802.15.4 channel.

  Figures 6.6 and 6.7 summarize the results obtained for both IEEE 802.11b and IEEE 802.11g in both regions for different traffic intensities at the IEEE 802.11 side. As for the timing simulations we have tested saturated, medium and low



(a) IEEE 802.11 b **(b)** IEEE 802.11 g

Figure 6.6: Sunspots: results in the symmetric region



(a) IEEE 802.11 b **(b)** IEEE 802.11 g

Figure 6.7: Sunspots: results in the asymmetric region

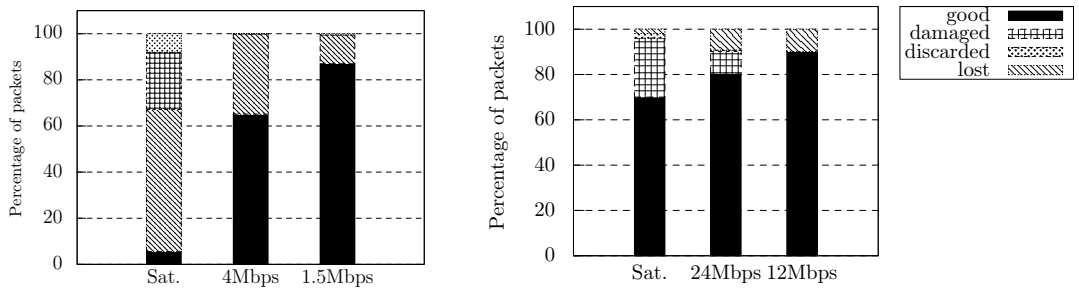
traffic at the IEEE 802.11 side and the IEEE 802.15.4 node generates a packet every 75ms. The packet lengths are the same chosen for simulations. The histograms show information about the percentage of correctly received packets, damaged (CRC-failed) packets, discarded packets and lost packets. Damaged packets and lost packets account for packets that collided with IEEE 802.11 transmissions.

It can be observed that experimental results are significantly different to the results obtained during the timing simulations. For instance, during the experiments the effect of saturated IEEE 802.11b traffic over the number of discarded packets is more visible than for IEEE 802.11g traffic. These results coincide with those presented by most of the experimental results: it is more likely that IEEE 802.15.4 nodes find the medium free when they are in presence of IEEE 802.11g transmissions. Also, under IEEE 802.11g interference, even if the packets are sent, an important percentage of these packets is lost or damaged. IEEE 802.15.4 seems not to detect IEEE 802.11g transmissions but interference has still a visible effect.

- **Experiments with the WSN430 nodes**

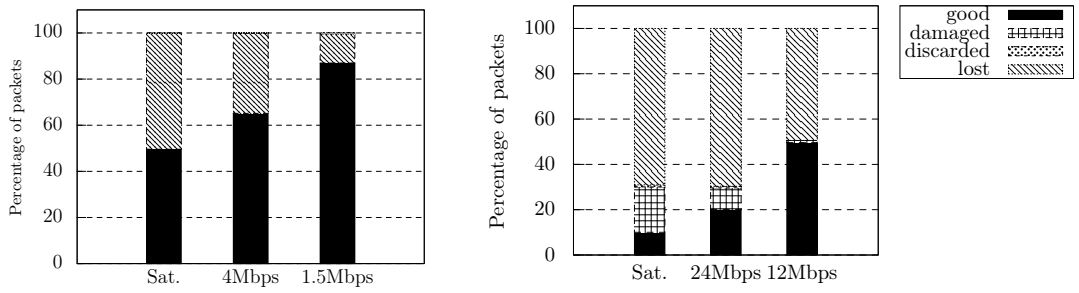
  As the SunsSPOTs, the WSN430 nodes use a CC2420 radio chip. We have used an implementation of the unslotted IEEE 802.15.4 CSMA/CA protocol available in ContikiOS.

We have set up the same scenario described above, we have used the same configuration for the CC2420 radio and we have logged the information about transmitted and received packets. The results of these experiments are summarized in figures 6.8 and 6.9.



(a) IEEE 802.11 b **(b)** IEEE 802.11 g

Figure 6.8: WSN430: results in the symmetric region



(a) IEEE 802.11 b **(b)** IEEE 802.11 g

Figure 6.9: WSN430: results in the asymmetric region

Despite that we keep the same configuration and we use the same radio chip, the results obtained with the WSN430 nodes are far from being similar to the results obtained with the SunSPOTs. Nevertheless, we can observe that IEEE 802.11 transmissions significantly affect the performance of IEEE 802.15.4 communications. As in the former case, in the symmetric region, we observe that IEEE 802.11b interference has an important effect on IEEE 802.15.4 transmissions regarding the fraction of discarded packets.

d) Conclusions and observations

Simulations and experiments with two different hardware nodes have shown the severity of the coexistence problem. However, even if we observe some common characteristics, results from timing simulations and experiments are quite different. Additionally, results differ, as well, from the results presented in the literature. For instance,

the results presented by Liang *et al.* [128] exhibit little similarities with our results. The main issue of the cited work is that it ignores the fraction of packets that are discarded due to medium access failures. Also, the authors explain the elevated packet error rate in the symmetric region by the fact that both IEEE 802.11 and IEEE 802.15.4 stations start transmitting simultaneously (*i.e.* they finish their back-off period almost at the same time); then, as soon as the IEEE 802.11 station finishes its transmission it detects the IEEE 802.15.4 transmission and defers. Authors do not further discuss this phenomenon and they do not determine its occurrence probability. In fact, as soon as the IEEE 802.15.4 station finds the medium free during the CCA check it starts sending the packet; however, there is a considerably long turnaround period (192μs) when going from the reception state to the transmission state. During this time, it is possible that the IEEE 802.11 station ends its back-off period and starts a transmission. During timing simulations we take into account this turnaround time and we see that in the symmetric region, damaged packets are due to collisions with IEEE 802.11 transmissions that start during the turnaround time. However, the number of discarded packets still accounts for a very significant fraction of the total lost packets and authors just suppose that all the lost packets have collided with a IEEE 802.11 transmission. Then, they have based their analysis and results on transmissions that have not even happened.

The significant differences in results show that reproducing an experiment in these kinds of scenarios is not straightforward. This is mainly due to the complex design space and the large number of parameters. For example, we have seen that evaluating exactly the same scenario using different hardware nodes leads to completely different results. These differences may be due to hardware characteristics (*i.e.* amplifiers, internal noise, sensibility, communication with the radio) and software characteristics (*i.e.* the operating system, the implementation of the CSMA/CA protocol). The dependency of experiment results from the characteristics of the hardware platform has already been stated by Srinivasan *et al.* [139]. Then, solutions that may work for a given hardware/software stack may be useless for other platforms and results cannot be generalized.

6.4.2 Overcoming hardware and resource limitations: The BER tests example

In this section we go through a case of study that illustrates different hardware and software limitations that have to be faced when experimenting. Overcoming such limitations requires a proper methodology and good knowledge about available resources. Additionally, we show that possible sources of errors may appear leading to incorrect conclusions.

We then cover all the process of setting up an experiment for BER estimation in the SensLAB platform. Such test could give interesting results regarding the quality of the channel and its characteristics. The Wake on Idle module could take advantage of this information to help nodes choosing “good” neighbors that are worth to be followed. Selecting the best links favors the formation of stable topologies; hence, optimal energy efficiency can be achieved since nodes remain in the stable state (*i.e.* analog tracking) most of the time.

BER tests can also be carried out to check if the PER vs BER (Equation 6.6) approximation holds for the real world time scales. In Equation 6.6, N refers to the length of the transmitted packet in bits.

$$PER = (1 - (1 - BER)^N) \tag{6.6}$$

a) Scenario and experiments description

The experiment scenario consists of 256 nodes (*i.e.* we have chosen the Grenoble site); at a given moment one of the nodes transmits a stream of bits; then, all nodes in the transmission range of the transmitter decode the stream and write the received information through the serial port. On a second time, we compare the received values with those expected and we estimate the distribution of errors. We have used the CC1101 radio chips and we have automatized the scenario execution using the framework discussed before in this chapter.

- **The test methodology**

  Given the random nature of channel errors we need to generate a stream of pseudo-random values to be transmitted by the node of interest. The transmission of a pseudo-random sequence helps to estimate a sufficiently accurate BER value. Then, during experiments each node sends a set of pseudo-random sequences; information received and logged by the receivers is then compared to the generated sequence to determine which bits were received with errors.

- **The radio chip configuration**

  Most of the radio chips are packet oriented. It means that the receiver synchronizes with the header of the packet sent by the transmitter. Then, it decodes the packet byte after byte. We have exploited an interesting feature of the CC1101 radio: the possibility to change the packet size. In fact, it is even possible to send packets of infinite size; then, the sender can transmit a stream of bytes and the receiver can decode these bytes regardless the size of the packet.

  Furthermore, the receiver can start decoding bytes present in the channel without the necessity to synchronize with a packet preamble; so, as soon as the receiver is started, it starts decoding information even if there are no real transmissions in the channel.

- **Performance limitations**

  Generation of pseudo-random stream of bits must be done at least as fast as the radio transmission rate (*i.e.* 250 kb/s for the IEEE 802.15.4 PHY layer).

  As the WSN430 nodes have a MCU running at a maximum frequency of 8 MHz we have around 31 clock cycles to generate one bit, write the sequence byte by byte in the radio transmission FIFO and send commands to the radio through the SPI to transmit data available in the FIFO. The implementation of a common pseudo-random generator such as the LFSR (Linear Feedback Shift Register) generator requires several shift operations and arithmetic and logic operations that may take several clock cycles to complete.

Efficient generation of pseudo-random sequences requires optimizing the algorithm implementation. We have implemented an LFSR generator. To generate sufficiently long sequences we have chosen a register length of 21 bits, *i.e* a sequence length (period) of $(2^{21} - 1)$ bits. The LFSR generator implementation is illustrated in Figure 6.10 and pseudo-random bit values are generated according to the polynomial expression given in Equation 6.7.

$$x^{21} + x^{19} + 1 \tag{6.7}$$

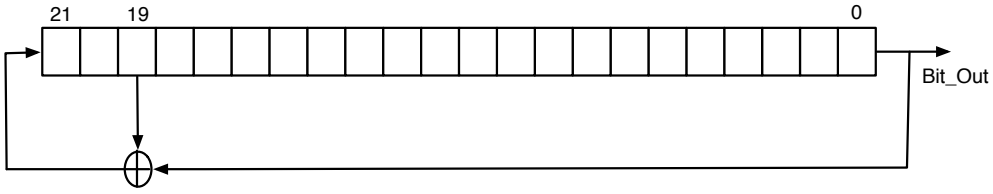


Figure 6.10: LFSR PRNG implementation

We have a 16-bits architecture and for a 21-bits register implementation we need to use 32-bit values; then, most of the operations take at least 2 CPU cycles to be completed; throughput of a straightforward implementation of the LFSR generation does not achieve the requirements for BER tests.

Thus, we have optimized the implementation by introducing a lookup table. Figure 6.11 illustrates the mechanism; pseudo-random bit values are generated based on a table predefined at compile time. Such table has been generated for a given seed value and must be changed each time a new seed is used.

The lookup operation takes much less CPU cycles to be competed than the typical implementation based on arithmetic operations. In fact, we were able to double the generation speed just by adding such mechanism. We reached generation rates of around 180 kb/s.

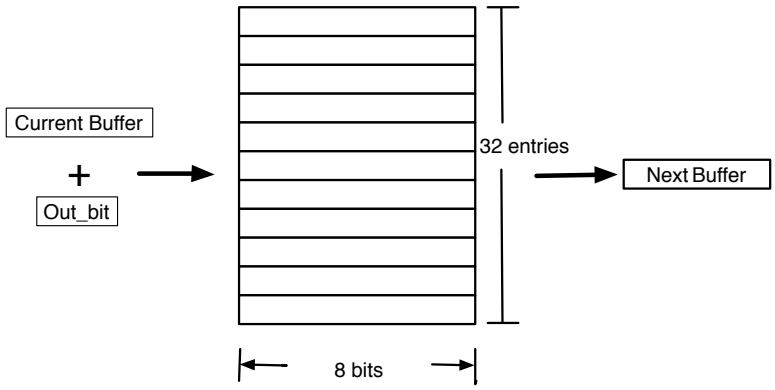


Figure 6.11: LFSR implementation using a Look up table

We have further optimized the implementation by using assembler operations very specific to the MSP430 architecture in such way that we were able to achieve the desired bit generation rate of 250 kb/s.

At the receiver side, the decoded stream must be analyzed to determine the position of the pseudo-random sequences. This process requires performing a correlation operation with the known sequence. Given the computation constraints an on-line execution of the correlation operation is not possible. Then, we have decided to log the received stream for later off-line analysis.

- **Serial link and data storage limitations**

  The serial link is used by the receiver to write the received stream; the logging speed is then limited by the serial link baud-rate. In the case of the WSN430 nodes, the maximum baud-rate is 115200 b/s; this speed is much lower than the radio data rate and even if data is correctly decoded by the receiver node it cannot be logged for off-line analysis. Then the radio communication is limited by the speed of the serial link. To allow the execution of additional operations such as radio chip control, additional CPU cycles have to be reserved. Empirically, we have observed a maximum communication data rate of 70 kb/s for correct logging of data. The data rate transmission of the CC1101 radio chip has then been set to this value.

  Regarding logging of data, the SensLAB site server quota for a user is limited to some hundreds of MBytes; however, during the tests the volume of generated data is considerable. For instance, for a test duration of 23 sequences per link we generate approximately 500GB of data. We have then used remote storage on our servers through a high speed link.

- **Data analysis**

  For the BER tests we have concatenated 23 sequences to generate the stream of bits to be transmitted. At the receiver side, positions of pseudo-random sequences are found by performing a correlation operation between the received bit-stream and known pseudo-random bit-stream. We have used the FFTW C library [140]. Given the randomness of the transmitted sequence, when a very high peak of correlation is found it is highly probable that we have found a sequence. We can determine the exact position of the sequence by using the position of the correlation peak (Figure 6.12).

  Some aspects must be taken into account for proper detection of pseudo-random sequences and data analysis:

  1. *Logging additional information*: When performing BER tests it is desirable to determine the causes of errors. The platform provides useful information that could permit to diagnose the possible causes or characteristics of bad links. Such information is the *RSSI* of the received signal, the *temperature* and other environmental variables that may affect the communication. We then decided to log such information periodically and embed it into the stream of received bytes to be extracted by the software tool at the moment of analysis.

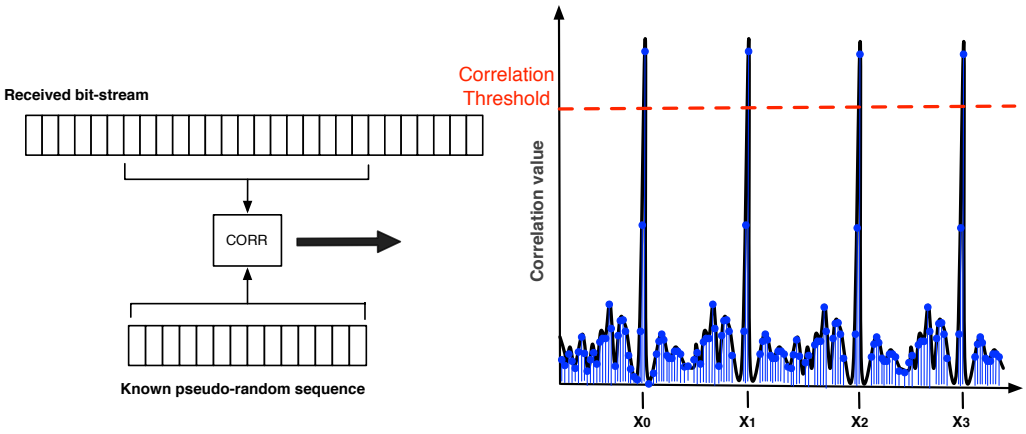


Figure 6.12: Correlation procedure to determine position of sequences

2. *Extraction of information*: When analysing the logged data we need to determine the experiment "start" time; this will allow to discover the position of additional information, to extract such information and correlate the remaining bytes to find the sequences. For that we have added a synchronization word (formed of 32 bytes) which indicates that the experiment has started.
3. *Sequence identifier*: We have added an identifier to each of the 23 transmitted sequences in order to determine whether a sequence was lost and could not be detected. Then, we have embedded a sequence identifier into the bitstream in well defined positions. In this way the analysis tool is able to tag each sequence (see figure 6.13), the sequence identifier is written several times along the sequence to allow robustness in case of bit errors.

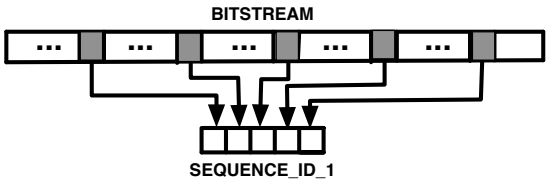


Figure 6.13: Sequence identifier: The sequence ID is embedded into the transmitted bitstream

b) Observations and conclusions

Until now we have described the methodology followed to perform BER tests in the SensLAB platform; we have also described different choices and optimizations applied during the process of setting up the experiment.

At the moment of evaluating BER we have found several links that exhibit a very high BER value and that loose a complete sequence while the remaining sequences were

correctly received with a BER of almost zero. While this situation may happen due to variable channel conditions, we do not consider it as a normal behavior. We have then investigated the phenomenon to understand the cause of errors.

The analysis tool correlates the information with the known sequence to determine the position of the 23 consecutive sequences (*i.e.* the correlation peak informs about the position of the sequence). Then, after knowing the sequences positions, comparison with the expected sequence can be done in order to find erroneous bits, as seen in Figure 6.14.

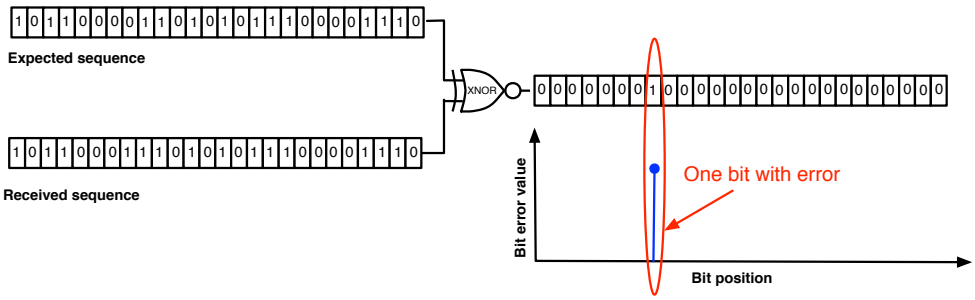


Figure 6.14: BER estimation: an XOR operation is performed

Observing the bits with error in some of the lost sequences we found a particular pattern: more than half of the sequence is correctly received and after some point it is completely wrong. This pattern suggests us that some bits were lost, avoiding proper recovery of the sequence (Figure 6.15). To verify our hypothesis we have added dummy bits to the received sequence a soon as too many consecutive errors start happening; then, we were able to recover the sequence (Figure 6.16).

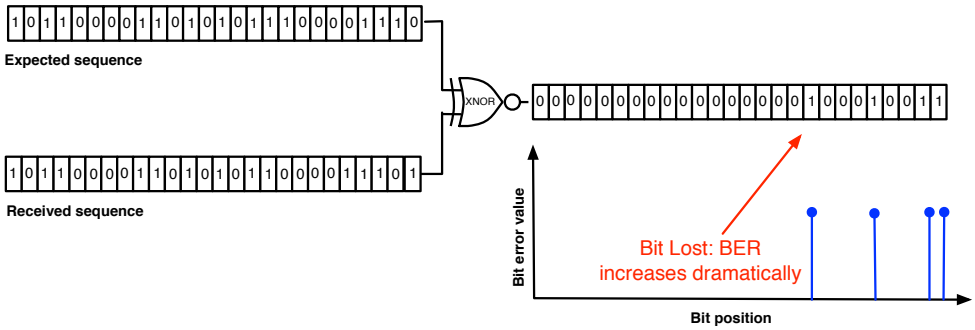


Figure 6.15: The anomaly of loosing one bit: bad BER estimation

The fact that one or several bits are lost when receiving the pseudo-random sequence significantly affects the estimated value of BER and leads to wrong conclusions regarding the quality of the link. Since only bits of information are lost we can deduce that the serial link is not the cause of these errors (*i.e.* information is written as characters of bytes). Hence, the radio chip was not able to detect and decode the lost bit and the radio chip hardware is the one introducing such error. Moreover, we have observed

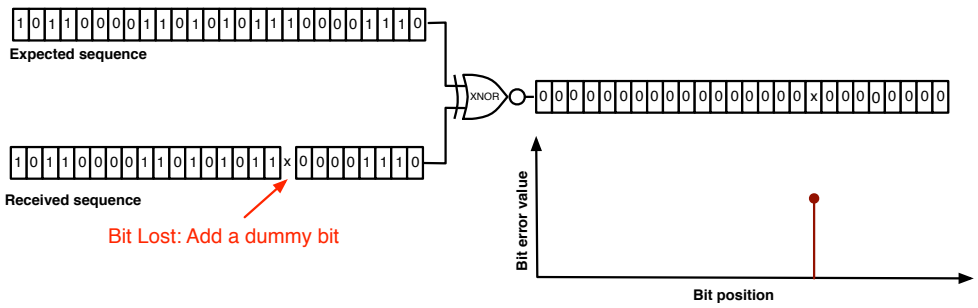


Figure 6.16: Procedure to recover a lost sequence: a dummy bit is added

that this phenomenon happens with a very low probability for perfect links (*i.e.* links with BER equal 0). Hence, channel conditions prevent the radio to decode bits; such errors must be detected and corrected to obtain a proper estimation of BER.

6.5 Summary and Conclusions

In this chapter we have discussed the importance of real experimentation in WSNs. We have described two case of studies that require real test-bed evaluation. Going through these studies illustrates the issues to face when setting up an experimental scenario; there are a lot parameters and factors that affect the behavior of the system.

Firstly, we have tried to reproduce the work presented in the literature around the IEEE 802.11/IEEE 802.15.4 coexistence problem. Our results differ significantly from those presented in related work; in fact, the huge design space makes difficult the reproduction of experiments. Even with the same scenario, using different hardware motes we obtain different results; hence conclusions are highly dependent of the hardware and the software being used.

Then we have gone through the set up process of a BER test in the SensLAB platform. We have discussed the different challenges to be faced during the procedure. We have also observed that hardware errors may avoid the correct estimation of the BER value and special care has to be taken when these kinds of errors arrive.

Chapter 7

Conclusions and future directions

Contents

The huge number of applications for Wireless Sensor Networks leads to a set of contradictory constraints and requirements. We believe that in these kinds of applications flexibility must be traded off for energy efficiency; the use of specialized hardware units (analog and digital coprocessors) may help reaching the desired performance while energy consumption is minimized. We have shown that adding some intelligence into the radio chip helps to optimize energy consumption.

Sensing parameters and events detection can be done efficiently using analog hardware; then, the main processing unit and digital treatment of information can be activated as soon as an event is detected. The energy level or RSSI value on a given frequency band can also be seen as a variable that can be sensed: when the energy level exceeds a given threshold, an event indicating that there is an ongoing transmission can be triggered.

7.1 Conclusions and summary of results

This dissertation aims at proposing and evaluating protocols that leverage the introduction of "smart" radios while optimizing energy consumption. We exploit low-level analog channel sensing to provide a set of services to higher layers in the protocol stack and to the main processing unit.

First, we have described the Wake on Idle protocol and hardware module in Chapter 4. Wake on Idle allows, through the use of analog channel sensing, efficient wake up and very low duty cycles to account for low power consumption. Efficient wake up and medium access are provided on top of a neighborhood maintenance mechanism. The working principle of Wake on Idle is the generation of well defined instants of time for busy tones exchange based on pseudo-random values. Through a proof of concept implementation we have demonstrated that we are able to synchronize and assess the presence of neighbors based only on analog channel sensing. Additionally we provide a mechanism that allows efficient wake up of nodes and medium access to exchange data packets. Evaluation and analysis show that the proposed protocol achieves very low duty cycles and good communication performance: it is robust to external interference, it copes with contention and packets' collision probability is low. To allow proper synchronization, Wake on Idle calls in a repetitive way a set of routines: pseudo

random number generation, filtering and timer programming; these tasks can be easily implemented in hardware while no intervention from the main processing unit is needed. We have presented a possible implementation of the hardware module that can be integrated with the radio chip; through the use of a set of configuration registers different system parameters can be changed. Additionally, an optional multi-channel extension can be used to increase the robustness when there is external interference and contention.

Secondly, we have proposed Sleep on Idle as another application of analog sensing. The idea of Sleep on Idle is to optimize duty cycle by exchanging information thanks to analog signaling. We have demonstrated that by introducing Sleep on Idle, IEEE 802.15.4 idle listening can be significantly reduced; the protocol can reach duty cycles comparable to asynchronous medium access methods such as ContikiMAC while keeping the advantages of having the nodes synchronized. The Sleep on Idle notification functionality can also be introduced within the smart radio.

Finally, we conclude the document by highlighting the importance of real experimentation when evaluating WSNs applications. We summarize conclusions taken from observations of real experiments; we have shown how results may be highly dependant of the stack (hardware and software) being used and we have discussed the huge set of parameters that may affect results and conclusions. We have also seen that hardware can introduce errors and proper results cannot be obtained. It is then very important to introduce a proper methodology for experimentation, data analysis and debugging. Such methodology implies facing several limitations.

7.2 Future directions

The work presented on this thesis opens a lot of opportunities for further research and work. We have shown the feasibility and preliminary results of the proposed protocols. However, several open issues have to be covered to obtain optimal energy consumption of the protocols. Some of the aspects that have to be covered in future work are listed below.

Efficient neighborhood discovery

In Chapter 4 we have discussed the problem of neighborhood discovery in Wireless Sensor Networks; we have presented a possible solution and have left its evaluation and implementation as an open issue.

Lets discuss some aspects to take into account when implementing the "hybrid" approach being proposed as a preliminary solution to the problem:

- *Time to discover nodes*

  If we suppose all nodes enter the random-access (discovery) state at almost the same time, they start sending and listening for (broadcast) advertisements or "hello" frames. A soon as a new node is discovered it is added in the neighbor's list. The questions are: how long should the nodes remain in the discovery state? when should they pass to the table state (*i.e.* activation of Wake on Idle)? If the nodes have an approximation of the possible number of nodes on its transmission

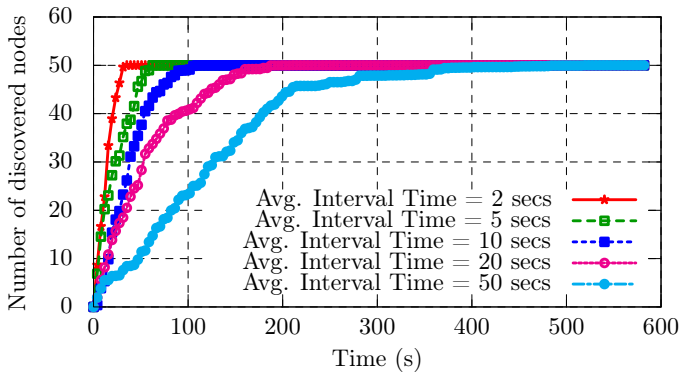


Figure 7.1: ContikiMAC: Number of discovered nodes vs time(s)

range, they can stop the discovery procedure when a certain percentage of these nodes is discovered. The discovery process can be very expensive in terms of energy consumption (*i.e.* broadcast frames are continuously transmitted) and it should be stopped as soon as possible.

We have carried out some preliminary experiments where nodes using ContikiMAC send (and listen) broadcast notifications, based on randomly generated time intervals. We have set-up a testbed of 50 nodes, the sampling frequency of ContikiMAC is 4 Hz and the transmission power 0 dBm; distance between nodes is of some centimeters and all nodes are able to see each other's transmissions. We have varied the average time between broadcast transmissions, then we have observed the time needed by the nodes to discover nodes present on their transmission range.

The number of discovered nodes vs time is plotted in Figure 7.1. Every curve corresponds to a different average time between beacons (hello frames). In most of the cases, nodes take some seconds to discover more than half of their neighbors. The proper time between announcements needs to be found: short intervals cause high contention which leads to collisions and energy waste; on the other side, long intervals may lead to very long discovery periods. This value must be chosen according to the density of the network and the parameters for duty cycling (*i.e.* sampling frequency).

- *Choosing potential neighbors*

  After nodes present on its transmission range have been discovered, the node must decide whether to associate to a given node. To take this decision, a metric must be chosen. For example the RSSI (and the LQI) values of received beacon frames can be used as an indicator of the link quality with a given node. The idea is to choose stable links to obtain a stable topology and efficiently perform neighborhood maintenance using Wake on Idle.

- *Determining loss of connectivity*

  Another important point is how nodes detect that connectivity has been lost and the discovery process must be re-executed in order to form a new topology. Using Wake on Idle nodes can detect whether a neighbor has been disconnected but a soon as a node is completely isolated from the network it cannot inform other nodes this fact and it cannot join the network. Every node can decide whether it needs to enter the discovery state; then, it wakes up the whole network and the discovery process is started.

  The mechanism introduced to wake up the network is another aspect to take into account. We can take advantage of synchronization and the possibility to wake up (one-hop) neighbors by omitting the transmission of analog busy tones.

- *Topology formation*

  After suitable nodes are chosen, nodes must decide whether they are follower or initiator with respect to a given neighbor. If, for example, a cluster tree topology is to be formed, a parent could be the initiator for its children, taking into account that a node can keep associations with more than one parent. We must study some metrics to automatically determine the relationship parent/child or initiator/follower.

Hardware module implementation

We are planning to implement the hardware module (WoI/SoI) using a Hardware Description Language (HDL) for synthesis and further evaluation. Novel FPGA technologies that account for very low power consumption can be used. Optimal energy consumption of the protocols can be evaluated thanks to the actual implementation of this module.

Security extension

Security is becoming very important in Wireless Sensor Networks. In Chapter 4 we have stated that using Wake on Idle along with cryptographic approaches for pseudo-random number generation and exchange of seeds provides secure communication, specially when frequency-hopping is added.

Other future work

Regarding Wake on Idle there are some other aspects that require further analysis and study, in particular: i) Support for burst transmissions and ii) Support for multicast and broadcast transmissions.

Another interesting approach is the possibility to adapt different parameters and characteristics of the protocols. The smart radio can adapt to a given condition and it can reconfigure itself. We want to study how to adapt the protocols according to a set of measured parameters.

We are also planning to exploit the data obtained during the experiments performed around the IEEE 802.15.4/IEEE 802.11 coexistence. In fact, a solution to the communication degradation problem can be the introduction of a "smart radio" able to coordinate transmissions from both standards.

Bibliography

[1] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An Operating System for Sensor Networks Ambient Intelligence. In *Ambient Intelligence*. 2005. 8, 14

[2] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, 2004. 8, 14, 93

[3] J. Polastre, R.t Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005. 13

[4] MICAz Datasheet. `http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf`. 13

[5] WSN430 Datasheet and developers guide. `http://perso.ens-lyon.fr/eric.fleury/Upload/wsn430-docbook`. 13, 53

[6] Ez430-RF2500 Datasheet. `http://www.ti.com/tool/ez430-rf2500`. 13, 53

[7] Sun microsystems. SunSpots website. `http://www.sunspotworld.com`. 13, 118

[8] ST microelectronics. STM32W108xx: High-performance, IEEE 802.15.4 wireless system-on-chip. Doc ID 16252 Rev 13. 13

[9] J. Ko, K. Klues, C. Richter, W. Hofer, B. Kusy, M. Bruening, T. Schmid, Q. Wang, P. Dutta, and A. Terzis. Low Power or High Performance: A Tradeoff Whose Time Has Come (and Nearly Gone). In *Proceedings of EWSN: European Conference on Wireless Sensor Networks*, 2012. 14

[10] J. Gilbert and F. Balouchi. Comparison of energy harvesting systems for Wireless Sensor Networks. *International Journal of Automation and Computing*, 2008. 14

[11] M.O. Farooq and T. Kunz. Operating Systems for Wireless Sensor Networks: A Survey. *Sensors*, 2011. 14

[12] J. Mitola. The software radio architecture. *Communications Magazine, IEEE*, 1995. 15

[13] GNURadio Official Website. `http://gnuradio.org/redmine/projects/gnuradio/wiki`. 15

[14] Q. Zhang, G.J.M. Smit, L.T. Smit, A. Kokkeler, F.W. Hoeksema, and M. Heskamp. A reconfigurable platform for cognitive radio. In *IEE mobility conference 2005*, 2005. 15

[15] G. J. Minden, J. B. Evans, L. Searl, D. DePardo, V. R. Petty, R. Rajbanshi, T. Newman, Q. Chen, F. Weidling, J. Guffey, D. Datla, B. Barker, M. Peck, B. Cordill, A. M. Wyglinski, and A. Agah. KUAR: A Flexible Software-Defined Radio Development Platform. In *New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, 2007. 15

[16] L. Belanger R. Sathappan, M. Dumas and M. Uhm. New architecture for development platform targeted to portable applications. In *SDR Forum Technical Conference (SDR)*, 2006. 15

[17] A. Khattab, J. Camp, C. Hunter, P. Murphy, A. Sabharwal, and E.W. Knightly. WARP: a flexible platform for clean-slate wireless medium access protocol design. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2008. 15

[18] I. Miro-Panades, F. Clermidy, P. Vivet, and A. Greiner. Physical Implementation of the DSPIN Network-on-Chip in the FAUST Architecture. In *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '08, 2008. 15

[19] G. Lu, D. De, M.Xu, W.Z. Song, and J. Cao. TelosW: Enabling ultra-low power wake-on sensor network. In *Proceedings of the 7th International Conference on Networked Sensing Systems*, 2010. 16, 19

[20] Texas Instruments. CC1101 Low-Power Sub-1 GHz RF Transceiver. Data Sheet SWRS061G. 16

[21] G. Asada, T. Dong, F. Lin, G. Pottie, W. Kaiser, and H. Marcy. Wireless Integrated Network Sensors: Low power systems on a chip. In *ESSCIR Proceeding of Solid-State Circuits Conference*, 1998. 16

[22] L. Gu and J.A. Stankovic. Radio-Triggered Wake-Up for Wireless Sensor Networks. *Real-Time Systems*, 2005. 17

[23] B. Doorn, W. Kavelaars, and K. Langendoen. A prototype low cost wakeup radio for the 868 MHz band. *International Journal in Sensor Networks*, 2009. 17

[24] B. Rumberg, D.W. Graham, V. Kulathumani, and R. Fernandez. Hibernets: Energy-efficient sensor networks using analog signal processing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2011. 17

[25] J. Ansari, D. Pankin, and P. Mähönen. Radio-triggered wake-ups with addressing capabilities for extremely low power sensor network applications. *IJWIN International Journal of Wireless Information Networks*, 2009. 17

[26] V. Rosello Gomez-Lobo, J. Portilla Berrueco, and T. Riesgo Alcaide. Ultra Low Power FPGA-Based Architecture for Wake-up Radio in Wireless Sensor Networks. In *Proceedings of the 37th Annual Conference of the IEEE Industrial Electronics Society*, 2011. 17

[27] Actel. IGLOO Low Power Flash FPGAs with Flash Freeze Technology. IGLOO DS. Rev 23. 17

[28] B. A. Warneke and K. S. J. Pister. An ultra-low energy microcontroller for Smart Dust Wireless Sensor Networks. In *Solid-State Circuits Conference. Digest of Technical Papers*, 2004. 18

[29] S. Mysore, B. Agrawal, F.T. Chong, and T. Sherwood. Exploring the Processor and ISA Design for Wireless Sensor Network Applications. In *Proceedings of the 21st International Conference on VLSI Design*, 2008. 18

[30] L. Nazh, M. Minuth, B. Zhai, J. Olson, T. Austin, and D. Blaauw. A second-generation sensor network processor with application-driven memory optimizations and out-of-order execution. In *ACM/IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, 2005. 18

[31] V. Ekanayake, C. Kelly, IV, and R. Manohar. An ultra low-power processor for sensor networks. In *Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*, 2004. 18

[32] V. Ekanayake, C. Kelly, IV, and R. Manohar. BitSNAP: Dynamic Significance Compression for a Low-Energy Sensor Network Asynchronous Processor. In *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*, 2005. 18

[33] S. Namtvedt. AN047: CC1100/CC2500 - Wake-On-Radio. Technical report, Texas Instruments, 2009. 19

[34] M. Sheets, F. Burghardt, T. Karalar, J. Ammer, Y. H. Chee, J. Rabaey, and A. Functionality. A power-managed protocol processor for wireless sensor networks. In *in Proc. IEEE Symp. VLSI Circuits*, 2006. 19

[35] M. Hempstead, N. Tripathi, P. Mauro, G. Wei, and D. Brooks. An ultra low power system architecture for sensor network applications. In *SIGARCH Computer Architectures News*, 2005. 19

[36] M.A. Pasha, S. Derrien, and O. Sentieys. A Novel Approach for Ultra Low-Power WSN Node Generation. In *ISSC IET Irish Signals and Systems Conference*, 2010. 20

[37] M.A. Pasha, S. Derrien, and O. Sentieys. A complete design-flow for the generation of ultra low-power WSN node architectures based on micro-tasking. In *Proceedings of the 47th Design Automation Conference*, 2010. 20

[38] M.A. Pasha, S. Derrien, and O. Sentieys. System-level synthesis for wireless sensor node controllers: A complete design flow. *TODAES ACM Transactions on Design Automation of Electronic Systems*, 2012. 20

[39] T. Muck, A. Frohlich, M. Gernoth, and W. Schroder-Preikschat. Implementing OS components in hardware using AOP. *SIGOPS Operating Sysems Review*, 2012. 20

[40] P.R. Panda. SystemC: A modeling platform supporting multiple design abstractions, 2001. 20

[41] F. Liu, Q. Tan, X. Song, and N. Abbasi. AOP-based high-level power estimation in SystemC. In *GLSVLSI Proceedings of the 20th symposium on Great lakes symposium on VLSI*, 2010. 20

[42] Y. Endoh. ASystemC: an AOP extension for hardware description language. In *AOSD Proceedings of the tenth international conference on Aspect-oriented software development companion*, 2011. 20

[43] M. Amiri. Evaluation of Lifetime Bounds of Wireless Sensor Networks. *Computing Research Repository*, 2010. 20

[44] P. Völgyesi, J. Sallai, S. Szilvási, P. Dutta, and Á. Lédeczi. Marmot: A Novel Low-Power Platform for WSNs. In *Networked Digital Technologies*, 2010. 20

[45] R. Min, M. Bhardwaj, S. Cho, E. Shih, A. Sinha, A. Wang, and A. Chandrakasan. Low-Power Wireless Sensor Networks. In *VLSI Design*, 2001. 20

[46] Rajesh M. Vikrant V. FPGA Based Kalman Filter for Wireless Sensor Networks. In *International Journal of Computer Technology and Applications*, 2011. 20

[47] T. Kwok and Y. Kwok. Computation and Energy Efficient Image Processing in Wireless Sensor Networks Based on Reconfigurable Computing. In *Proceedings of the 2006 International Conference Workshops on Parallel Processing*, 2006. 20

[48] V. Raghunathan, C. Schurgers, S. Park, M. Srivastava, and B. Shaw. Energy-aware wireless microsensor networks. In *IEEE Signal Processing Magazine*, 2002. 20

[49] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings 21st International Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002. 24, 28

[50] IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, 2007. 24

[51] M. N. Halgamuge, M. Zukerman, K. Ramamohanarao, and H. L. Vu. An estimation of sensor energy consumption. *Electromagnetics Research B*, 2009. 24

[52] M. Al Ameen, S. M. R. Islam, and K. Kwak. Energy Saving Mechanisms for MAC Protocols in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, 2010. 24

[53] R. Ahmad, E. Dutkiewicz, and X. Huang. A Survey of Low Duty Cycle MAC Protocols in Wireless Sensor Networks. 2010. 24

[54] A Bachir, M Dohler, and T Watteyne. MAC essentials for Wireless Sensor Networks. *Surveys & Tutorials*, 2010. 24

[55] S.M. Lasassmeh and J.M. Conrad. Time synchronization in Wireless Sensor Networks: A survey. In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, 2010. 26

[56] S. Yoon, C. Veerarittiphan, and M.L. Sichitiu. Tiny-sync: Tight time synchronization for Wireless Sensor Networks. *ACM Transactions in Sensor Networks*, 2007. 26, 48, 54

[57] A. Duda, G. Harrus, Y. Haddad, and G. Bernard. Estimating global time in distributed systems. In *Proceedings of the International Conference on Distributed Computing Systems*, 1987. 26, 54

[58] S. Ganeriwal, R. Kumar, and M.B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003. 26, 54

[59] F. Kuhn, C. Lenzen, T.s Locher, and R. Oshman. Optimal gradient clock synchronization in dynamic networks. In *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 2010. 26

[60] L. Ma, H. Zhu, and G. Nallamothu. Impact of linear regression on time synchronization accuracy and energy consumption for Wireless Sensor Networks. *Military Communications Conference MILCOM*, 2008. 27, 55

[61] Z. Zhong, P. Chen, and T. He. On-demand time synchronization with predictable accuracy. In *Proceedings of IEEE INFOCOM*, 2011. 27

[62] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *OSDI Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, 2002. 27, 48

[63] T. Schmid, J. Friedman, Z. Charbiwala, Y.H. Cho, and M.B. Srivastava. Low-power high-accuracy timing systems for efficient duty cycling. In *Proceedings of the 13th international symposium on Low power electronics and design*, 2008. 27

[64] S Ganeriwal, I Tsigkogiannis, H Shim, V Tsiatsis, M B Srivastava, and D Ganesan. Estimating Clock Uncertainty for Efficient Duty-Cycling in Sensor Networks. *IEEE/ACM Transactions on Networking*, 2009. 27

[65] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, 2000. 28

[66] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves. Energy-efficient, collision-free Medium Access Control for Wireless Sensor Networks. *Wireless Networking*, 2006. 28

[67] T. Van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for Wireless Ssensor Networks. In *SenSys Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003. 28

[68] J. Kim, J. On, and S. Kim. Performance evaluation of synchronous and asynchronous MAC protocols for wireless sensor networks. *Sensor Technologies*, 2008. 28

[69] Y. Sun, S. Du, O. Gurewitz, and D.B. Johnson. DW-MAC: a low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks. In *MobiHoc Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, 2008. 28, 33

[70] W. Ye, F. Silva, and J. Heidemann. Ultra-Low Duty Cycle MAC with Scheduled Channel Polling. In *SenSys Proceedings of the Fourth International Conference On Embedded Networked Sensor Systems*, 2006. 28

[71] 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 (WPANs). *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, 2006. 28, 34

[72] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for Wireless Sensor Networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04. ACM, 2004. 29

[73] Texas Instruments. CC2420 2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver. Data Sheet SWRS041B. 30

[74] M. Buettner, G.V. Yee, E. Anderson, and R. Han. X-MAC: a short preamble MAC protocol for duty-cycled Wireless Sensor Networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM, 2006. 31

[75] D. Moss. Box-macs: Exploiting physical and link layer boundaries in low-power networking. *Stanford Information Networks Group Technical Report*, 2008. 31

[76] A. El-Hoiydi and J.-D. Decotignie. WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks. *IEEE Symposium on Computers and Communications (ISCC)*, 2004. 31

[77] W. Pak, K.T. Cho, J. Lee, and S. Bahk. W-MAC: Supporting Ultra Low Duty Cycle in Wireless Sensor Networks. In *GLOBECOM Proceedings of the Global Communications Conference*, 2008. 31

[78] A. Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical report, Swedish Institute of Computer Science, 2011. 32, 96

[79] H.i Cao, K.W. Parker, and A. Arora. O-MAC: A Receiver Centric Power Management Protocol. In *Proceedings of ICNP'06*, 2006. 33

[80] L. Tang, Y. Sun, O. Gurewitz, and D.B. Johnson. PW-MAC: An energy-efficient Predictive-Wakeup MAC protocol for Wireless Sensor Networks. In *Proceedings of INFOCOM*, 2011. 33, 75, 76, 77

[81] L. Tang, Y. Sun, O.r Gurewitz, and D. Johnson. EM-MAC: A Dynamic Multi-channel Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of MobiHoc'11*, 2011. 33, 79, 86

[82] J. Na, S. Lim, and C. Kim. Dual Wake-up low power listening for duty cycled Wireless Sensor Networks. *EURASIP Journal on Wireless Communication Networks*, 2008. 33

[83] P. Dutta, S. Dawson-Haggerty, Y. Chen, C. M. Liang, and A. Terzis. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. In *Proceedings of SenSys'10*, 2010. 33, 91

[84] B. Jang, J.B. Lim, and M.L. Sichitiu. AS-MAC: An asynchronous scheduled MAC protocol for wireless sensor networks. In *MASS*, 2008. 33

[85] J. Degesys, I. Rose, A. Patel, and R. Nagpal. Desync: Self-organizing desynchronization and TDMA in wireless sensor networks. In *In Proc. Conference on Information Processing in Sensor Networks*, 2007. 34

[86] 802.15.4e-2012 - IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Std 802.15.4e-2012*, 2012. 37

[87] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626, 2003. 45

[88] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. *Internet RFCs*, 2003. 45

[89] P. Levis, E. Brewer, D. Culler, D. Gay, S. Madden, N.l Patel, J. Polastre, S. Shenker, R.t Szewczyk, and A. Woo. The emergence of a networking primitive in Wireless Sensor Networks, Communication of the ACM, 2008. 47

[90] A. Pandey and R.C. Tripathi. A Survey on Wireless Sensor Networks Security. *International Journal of Computer Applications*, 2010. 49

[91] J. Sen. A Survey on Wireless Sensor Network Security. *International Journal of Communication Networks and Informatics*, 2009. 49

[92] Senslab. Drivers for WSN430 motes. `http://www.senslab.info/?p=328`. 53

[93] G. Marsaglia. Yet another RNG, 1994. `http://groups.google.com/group/sci.stat.math`. 61

[94] P. Martin. An analysis of random number generators for a hardware implementation of genetic programming using FPGAs and Handel-C. In *Proceedings of GECCO'02*, 2002. 61

[95] S. Kulkarni, A. Iyer, and C. Rosenberg. An address-light, integrated MAC and routing protocol for Wireless Sensor Networks. *IEEE/ACM Trans. Netw.*, 2006. 71

[96] X. Wang, X. Zhang, Q. Zhang, and G. Chen. An energy-efficient integrated MAC and routing protocol for Wireless Sensor Networks. In *Proceedings of the 2009 IEEE international conference on Communications*, ICC'09, 2009. 71

[97] W. Zeng, A. Arora, and N. Shroff. Maximizing energy efficiency for convergecast via joint duty cycle and route optimization. In *INFOCOM Proceedings of the 29th conference on Information communications*, 2010. 71

[98] R. Venkatesh, J. Garcia-Luna-Aveces, and K. Obraczka. Energy-Efficient application-aware Medium Access for Sensor Networks. In *2nd IEEE Conf. on Mobile Ad-hoc and Sensor Systems*, 2005. 71, 87

[99] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-Low Power Data Gathering in Sensor Networks. In *Proceedings of IPSN*, 2007. 71

[100] T. Watteyne, A. Mehta, and K. Pister. Reliability through frequency diversity: why channel hopping makes sense. In *Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*. ACM, 2009. 78

[101] K. Jones, A. Wadaa, S. Olariu, L. Wilson, and M. Eltoweissy. Towards a new paradigm for securing Wireless Sensor Networks. In *Proceedings of the 2003 workshop on New security paradigms*, NSPW '03, 2003. 78

[102] M. J. Mcglynn. Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2001. 85

[103] S. A. Borbash, A. Ephremides, and M. J. McGlynn. An asynchronous neighbor discovery algorithm for Wireless Sensor Networks. *Ad Hoc Networks*, 2007. 85

[104] Y. Tseng, C. Hsu, and T. Hsieh. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. *Joint Conference of the IEEE*, 2002. 85

[105] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers (Oxford Science Publications)*. Oxford University Press, USA, 1980. 85

[106] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *SenSys Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008. 85, 103

[107] A. Kandhalu, K. Lakshmanan, and R. Rajkumar. U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010. 85

[108] Specification of the Bluetooth System, Volume 1,2,3: Core, v4.0. Bluetooth SIG, 2010. 86

[109] D. Bohman, M. Frank, P. Martini, and C. Scholz. Performance of symmetric neighbor discovery in bluetooth ad hoc networks. *GI-Jahrestagung*, 2004. 86

[110] C. Drula, C. Amza, F. Rousseau, and A.j Duda. Adaptive energy conserving algorithms for neighbor discovery in opportunistic Bluetooth networks. *Selected Areas in Communications, IEEE Journal on*, 2007. 86

[111] C. Cordeiro and K. Challapali. C-MAC: A cognitive MAC protocol for multi-channel wireless networks. *New Frontiers in Dynamic Spectrum Access Networks*, 2007. 86

[112] N. Karowski, A. Carneiro Viana, and A. Wolisz. Optimized Asynchronous Multi-channel Neighbor Discovery. In *INFOCOM IEEE Conference on Computer Communications*, 2011. 86

[113] G. Chalhoub, A. Guitton, F. Jacquet, A. Freitas, and M. Misson. Medium Access Control for a tree-based Wireless Sensor Network: Synchronization management. In *Wireless Days. WD '08. 1st IFIP*, 2008. 87

[114] M. Neugebauer, J. Plönnigs, and K. Kabitzsch. A New Beacon Order Adaptation Algorithm for IEEE 802.15.4 Networks, 2005. 91

[115] J. Jeon, J. Lee, J. Yeao Ha, and W. Hyun Kwon. DCA: Duty-Cycle Adaptation Algorithm for IEEE 802.15.4 Beacon-Enabled Networks. In *VTC Spring Vehicular Technology Conference*, 2007. 91

[116] R. de Paz Alberola and D. Pesch. Duty Cycle Learning Algorithm (DCLA) for IEEE 802.15.4 Beacon-Enabled Wireless Sensor Networks. *Ad Hoc Netw.*, 2012. 91

[117] R. de Paz Alberola and D. Pesch. Joint Duty Cycle and Link Adaptation for IEEE 802.15.4 Beacon-Enabled Networks. In *Proceedings of the 6th Workshop on Hot Topics in Embedded Networked Sensors*, 2010. 91

[118] WSNet. WSNet Simulator Homepage. `http://wsnet.gforge.inria.fr`. 96

[119] D.J. Corbett and A.G Ruzzelli. A Procedure for Benchmarking MAC Protocols Used in Wireless Sensor Networks. Technical Report No. 593, 2006. 109

[120] SensLAB. Illustration of the SensLAB architecture. `http://wiki.senslab.info/documentation/platform/fit-eco_architecture`. 110

[121] A. Sikora and V. F. Groza. Coexistence of IEEE802.15.4 with other Systems in the 2.4 GHz-ISM-Band. 2005. 113

[122] M. Bertocco, G. Gamba, and A. Sona. Is CSMA/CA really efficient against interference in a wireless control system? An experimental answer. In *ETFA Emerging Technologies and Factory Automation. IEEE International Conference on*, 2008. 113

[123] M. Zeghdoud, P. Cordier, and M. Terre. Impact of Clear Channel Assessment Mode on the Performance of ZigBee Operating in a WiFi Environment. In *Operator-Assisted (Wireless Mesh) Community Networks, 1st Workshop on*, 2006. 113

[124] M. Petrova, L. Wu, P. Mahonen, and J. Riihijarvi. Interference measurements on performance degradation between colocated ieee 802.11g/n and ieee 802.15.4 networks. *International Conference on Networking*, 2007. 113, 114

[125] W. Yuan, Xi. Wang, J. M. G. Linnartz, and I. Niemegeers. Experimental Validation of a Coexistence Model of IEEE 802.15.4 and IEEE 802.11b/g Networks. *IJDSN*, 2010. 113, 114

[126] W. Yuan, J.P.. Linnartz, and I.G. Niemegeers. Adaptive CCA for IEEE 802.15.4 Wireless Sensor Networks to Mitigate Interference. In *WCNC*, 2010. 113

[127] S.Y. Shin, S. Choi, H.S. Park, and W.H. Kwon. Packet Error Rate Analysis of IEEE 802.15.4 Under IEEE 802.11b Interference. In *WWIC*, 2005. 114

[128] C.J. M. Liang, N. Priyantha, J. Liu, and A. Terzis. Surviving Wi-Fi interference in low power ZigBee Networks, booktitle = Sensys Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, year = 2010,. 114, 116, 121

[129] Y. Mao, Z. Zhao, and X. Jia. Understanding the indoor interference between IEEE 802.15.4 and IEEE 802.11b/g via measurements. In *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, 2011. 114, 115

[130] Z-Wave Alilanza. WLAN Interference and IEEE 802.15.4, 2007. 114, 115

[131] Schneider. ZigBee - WiFi Interference: White Paper and Test Note, 2008. 114, 115

[132] Freescale. MC1319x Coexistence in the 2.4GHz ISM Band. AN2935, 2010. 114

[133] C. Won, Jong-Hoon Youn, Hesham Ali, Hamid Sharif, and Jitender Deogun. Adaptive radio channel allocation for supporting coexistence of 802.15.4 and 802.11b. In *IEEE Vehicular Technology Conference*, 2005. 114

[134] S. Pollin, M. Ergen, A. Dejonghe, L. Van Der Perre, F. Catthoor, I. Moerman, and A. Bahai. Distributed Cognitive Coexistence of 802.15.4 with 802.11. In *Proc. IEEE Int'l Conf. Cognitive Radio Oriented Wireless Networks and Comm. (CROWNCOM '06)*, 2006. 114

[135] Min S. Kang, Jo W. Chong, Hyesun Hyun, Su M. Kim, Byoung H. Jung, and Dan K. Sung. Adaptive Interference-Aware Multi-Channel Clustering Algorithm in a ZigBee Network in the Presence of WLAN Interference. In *Proceedings of the 2nd International Symposium on Wireless Pervasive Computing (ISWPC 2007)*, 2007. 114

[136] B. Yun, J.and Lee, J. Li, and K. Han. A Channel Switching Scheme for Avoiding Interference of between IEEE 802.15.4 and Other Networks. In *Proceedings of the 2008 International Multi-symposiums on Computer and Computational Sciences*, 2008. 114

[137] Jennic. Co-existence of IEEE 802.15.4 at 2.4 GHz. JN-AN-1079. 115

[138] LIG laboratory. Ipmt tools. `http://ipmt.forge.imag.fr/`. 118

[139] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. Understanding the causes of packet delivery success and failure in dense Wireless Sensor Networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, 2006. 121

[140] M. Frigo and S.G. Johnson. The FFTW package. `http://www.fftw.org`. 124