IPP Hurray!

www.hurray.isep.ipp.pt

Technical Report

On the use of IEEE 802.15.4/ZigBee as federating communication protocols for Wireless Sensor Networks

André Cunha

HURRAY-TR-070902

Version: 1.0

Date: 17-09-2007

On the use of IEEE 802.15.4/ZigBee as federating communication protocols for Wireless Sensor Networks

André Cunha

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: arec@isep.ipp.pt

http://www.hurray.isep.ipp.pt

Abstract

We are witnessing the dawn of a new paradigm in Information and Communication Technologies (ICT), where large-scale embedded computing systems will interact with the physical environment in a ubiquitous and pervasive fashion. The landscape of potential new applications is tremendous; homeland security, environmental monitoring, health care, domotics, or factory automation are just a few elucidative examples of how these emerging embedded technologies will impact our daily life and society at large.

In order for these applications to become a reality, it is mandatory to find adequate Wireless Sensor Network (WSN) infrastructures that comply with complex requirements. In this context, IEEE 802.15.4/ZigBee seems potentially interesting to act as federating communication protocols for WSNs. Nevertheless, several issues in the standard specifications are still open.

The main research objectives of this Thesis are to evaluate the adequateness of the IEEE 802.15.4/ZigBee protocols for supporting WSN applications, the identification of open issues in the standard specifications and the proposal of solutions to effectively solve some of these problems and to enhance some native mechanisms.

For this purpose, an implementation of the IEEE 802.15.4/ZigBee protocol stack under the TinyOS operating system was conducted. This protocol stack has been used to leverage our research work, in the sense that it has been enabling the test, validation and demonstration of our scientific findings through experimentation. This work has also been driven by the need for an open-source implementation of the IEEE 802.15.4/ZigBee protocols, filling a gap between some newly released complex *C* implementations and black-box implementations from different manufacturers.

The problem of building synchronized cluster-tree networks, which are quite suitable for ensuring real-time and energy-efficient communications, is also addressed by this Thesis, demonstrating the feasibility of the Time Division Beacon Scheduling approach.

Reliability is also an important aspect in WSN applications, particularly those with critical requirements. ZigBee Cluster-Tree networks are prone to the single point of failure problem in the ZigBee Routers (cluster-heads). This Thesis presents a mechanism that improves on the ZigBee default behaviour by reducing or even eliminating network inaccessibility times in case of ZigBee Router failure.

Additionally, timeliness is an important feature of the IEEE 802.15.4 protocol, turning it quite appealing for applications under timing constraints. The native (explicit) Guaranteed Time Slot (GTS) allocation mechanism may be inefficient concerning bandwidth utilization. Thus, an implicit allocation mechanism (i-GAME) was implemented and proved to overcome that problem, by allowing several nodes to share the same GTS.

Finally, this Thesis presents a performance evaluation of the slotted CSMA/CA mechanism, comparing experimental and simulation results.

Universidade do Porto

Faculdade de Engenharia

FEUP

On the use of IEEE 802.15.4/ZigBee as federating communication protocols for Wireless Sensor Networks

By

André Ribeiro e Cunha

A dissertation submitted in partial fulfilment of the requirements for the degree of
Master in Communication Networks and Services

July 2007

Thesis Supervision:
Dr. Mário Ferreira Alves
ISEP

Thesis Co-Supervision:
Dr. Manuel Ricardo
FEUP

On the use of IEEE 802.15.4/ZigBee as federating communication protocols for Wireless Sensor Networks

Abstract

We are witnessing the dawn of a new paradigm in Information and Communication Technologies (ICT), where large-scale embedded computing systems will interact with the physical environment in a ubiquitous and pervasive fashion. The landscape of potential new applications is tremendous; homeland security, environmental monitoring, health care, domotics, or factory automation are just a few elucidative examples of how these emerging embedded technologies will impact our daily life and society at large.

In order for these applications to become a reality, it is mandatory to find adequate Wireless Sensor Network (WSN) infrastructures that comply with complex requirements. In this context, IEEE 802.15.4/ZigBee seems potentially interesting to act as federating communication protocols for WSNs. Nevertheless, several issues in the standard specifications are still open.

The main research objectives of this Thesis are to evaluate the adequateness of the IEEE 802.15.4/ZigBee protocols for supporting WSN applications, the identification of open issues in the standard specifications and the proposal of solutions to effectively solve some of these problems and to enhance some native mechanisms.

For this purpose, an implementation of the IEEE 802.15.4/ZigBee protocol stack under the TinyOS operating system was conducted. This protocol stack has been used to leverage our research work, in the sense that it has been enabling the test, validation and demonstration of our scientific findings through experimentation. This work has also been driven by the need for an open-source implementation of the IEEE 802.15.4/ZigBee protocols, filling a gap between some newly released complex *C* implementations and black-box implementations from different manufacturers.

The problem of building synchronized cluster-tree networks, which are quite suitable for ensuring real-time and energy-efficient communications, is also addressed by this Thesis, demonstrating the feasibility of the Time Division Beacon Scheduling approach.

Reliability is also an important aspect in WSN applications, particularly those with critical requirements. ZigBee Cluster-Tree networks are prone to the single point of failure problem in the ZigBee Routers (cluster-heads). This Thesis presents a mechanism that improves on the ZigBee default behaviour by reducing or even eliminating network inaccessibility times in case of ZigBee Router failure.

Additionally, timeliness is an important feature of the IEEE 802.15.4 protocol, turning it quite appealing for applications under timing constraints. The native (explicit) Guaranteed Time Slot (GTS) allocation mechanism may be inefficient concerning bandwidth utilization. Thus, an implicit allocation mechanism (i-GAME) was implemented and proved to overcome that problem, by allowing several nodes to share the same GTS.

Finally, this Thesis presents a performance evaluation of the slotted CSMA/CA mechanism, comparing experimental and simulation results.

Keywords: Wireless Sensor Networks; IEEE 802.15.4; ZigBee; TinyOS; ZigBee Cluster-Tree Topologies; Real-Time Communications; Fault-Tolerance.

Utilização de IEEE 802.15.4/ZigBee como protocolos de comunicação federadores para Redes de Sensores sem Fios

Resumo

Estamos a testemunhar o nascimento de um novo paradigma nas Tecnologias de Informação e Comunicação (ICTs), em que sistemas computacionais embebidos de grande escala irão interagir com o ambiente físico de uma forma ubíqua. O panorama de potenciais novas aplicações é extremamente vasto; segurança, monitorização ambiental, cuidados de saúde, domótica ou automação industrial são apenas alguns exemplos elucidativos do impacto destas tecnologias embebidas emergentes na nossa vida diária e na sociedade em geral.

Para que estas aplicações se tornem uma realidade, é fundamental encontrar infra-estruturas de redes de sensores sem fios (WSNs) que satisfaçam os requisitos complexos de escalabilidade, mobilidade, fiabilidade, temporalidade, custo e eficiência energética, entre outros. Neste contexto, os protocolos IEEE 802.15.4/ZigBee parecem potencialmente interessantes para serem utilizados em WSNs. Contudo, existem ainda muitos aspectos ambíguos e em aberto nas respectivas normas.

Os principais objectivos desta Tese são avaliar a adequabilidade dos protocolos IEEE 802.15.4/ZigBee para suportar aplicações baseadas em WSNs, identificar os pontos em aberto nas normas e propôr soluções para estes problemas e melhorar alguns dos seus mecanismos.

Neste contexto, foi implementada a pilha protocolar IEEE 802.15.4/ZigBee em TinyOS, que tem servido para testar, validar e demonstrar as nossas propostas científicas através da experimentação. Este trabalho foi também motivado pela necessidade de existir uma implementação em código-aberto facilmente perceptível e completa, o que não existia até aqui.

É também endereçada a construção de redes ZigBee Cluster-Tree sincronizadas, adequadas para suportar comunicações em tempo-real e eficiência energética, demonstrando a execuibilidade de um mecanismo de escalonamento do envio de beacons baseado na divisão do tempo (TDBS).

A fiabilidade é também um aspecto relevante nas aplicações WSN, particularmente aquelas com requisitos críticos. As redes ZigBee Cluster-Tree estão sujeitas ao problema de uma falha num ZigBee Router isolar duas partes da rede. Esta Tese apresenta um mecanismo (proactivo) que melhora o comportamento tradicional do ZigBee, reduzindo ou mesmo eliminando os tempos de inacessibilidade à rede.

O comportamento temporal é uma característica importante do protocolo IEEE 802.15.4, tornando-o atractivo para sistemas de tempo-real. O mecanismo de alocação explícita de “slots” (GTS) existente pode tornar-se ineficiente em termos de utilização da largura de banda. Portanto, foi implementado um mecanismo de alocação implícita (i-GAME), que permite que vários nós partilhem o mesmo GTS, provando-se a sua exequibilidade e eficiência.

Finalmente, esta Tese apresenta a avaliação da performance do mecanismo Slotted CSMA/CA, comparando resultados experimentais e de simulação.

Palavras-chave: Redes de Sensores sem Fios; IEEE 802.15.4; ZigBee; TinyOS; Topologias ZigBee Cluster-Tree; Comunicações tempo-real; Tolerância a falhas.

Acknowledgments

First of all I would like to express my gratitude to my supervisor, Mário Alves, for his outstanding supervision, counsel, advice, support, inspiration, patience, and for always being available during the course of this work. I would also like to thank my co-supervisor, Manuel Ricardo, for his interest in my work and guidance thought the development process of this Thesis.

I want to thank all the people in IPP-HURRAY! Research Group, at the School of Engineering of the Polytechnic Institute of Porto for their support and enthusiasm that makes working in IPP-HURRAY! very stimulating and challenging. It is impressive how so few can do so much.

A special thanks goes out to the ART-WiSe team, namely, Eduardo Tovar as the head of IPP-HURRAY! Research Group, Mário Alves, Anis Koubâa, an exceptional researcher, Ricardo Severino, Emmanuel Lomba, Skender Ben Attia, Petr Jurcik, Melek Attia, Anneleen Van Nieuwenhuyse and João Leal. Also, I would like to thank all my colleagues at Head-On lab.

I also must acknowledge all my teachers and colleges of my Master programme at the Faculty of Engineering in University of Porto, that made the last 2 years very fruitful and interesting.

I would also like to thank my family for the support they provided me through my entire life and to my friends for their encouragement.

Last but not least, a very special thanks to Ana Marta, without whose love, encouragement and support, I would not have finished this Thesis.

Contents

List of Figures

List of Tables

List of Acronyms

ACL	Access Control List
AODV	Ad hoc On Demand Distance Vector
APL	Application Layer
APS	Application Support Sublayer
APSDE-SAP	APS Data Entity Service Access Point
APSME-SAP	APS Management Entity Service Access Point
BE	Backoff Exponent
BI	Beacon Interval
BO	Beacon Order
BPSK	Binary Phase-Shift Keying
BSN	Beacon Sequence Number
CAP	Contention Access Period
CCA	Clear Channel Assessment
CFP	Contention Free Period
CID	Cluster Identifier
CLH	Cluster Head
COTS	Commercial-off-the-self
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CW	Contention Window (length)
DSN	Data Sequence Number
DSSS	Direct Sequence Spread Spectrum
ED	Energy Detection
FCS	Frame Check Sequence
FFD	Full Function Device
FH	Frequency Hopping
FHSS	Frequency Hopping Spread Spectrum
GTS	Guaranteed Time Slot
IFS	Interframe Spacing
LAN	Local Area Network
LIFS	Long Interframe Spacing
LLC	logical link control
LPDU	LLC protocol data unit
LQI	link quality indication
LR-WPAN	Low Rate-Wireless Personal Area Network
LSB	Least Significant Bit
M2M	Machine-to-Machine
MAC	Medium Access Control
MCPS	MAC Common Part Sublayer
MCPS-SAP	MAC Common Part Sublayer-Service Access Point
MFR	MAC Footer
MHR	MAC Header
MLME	MAC Sublayer management entity
MLME-SAP	MAC Sublayer Management Entity-Service Access Point

MPDU	MAC Protocol Data Unit
MSB	Most Significant Bit
MSDU	MAC Service Data Wnit
NB	Number of Backoff (periods)
NLDE-SAP	Network Layer Data Entity Service Access Point
NWK	ZigBee Network layer
NWK	Network Layer
OSI	Open Systems Interconnection
PAN	Personal Area Network
PD-SAP	PHY data service access point
PDU	Protocol Data Unit
PHR	Physical Header
PHY	Physical Layer
PIB	Personal Area Network Information base
PLME	Physical Layer Management Entity
PLME-SAP	Physical Layer Management Entity-Service Access Point
POS	Personal Operating Space
PPDU	PHY Protocol Data Wnit
PSDU	PHY Dervice Data Unit
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
RFD	Reduced Function Device
RSSI	Received Signal Strength Indication
RX	Receive or Receiver
SAP	Service Access Point
SD	Superframe Duration
SDU	Dervice Data Unit
SFD	Start-of-Frame Delimiter
SIFS	Short Interframe Spacing
SO	Superframe Order
TDBS	Time Division Beacon Scheduling
TRX	Transceiver
TX	Transmit or Transmitter
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
ZC	ZigBee Coordinator
ZDO	ZigBee device Objects
ZED	ZigBee End Device
ZG	ZigBee Gateway
ZR	ZigBee Router

Chapter 1

Overview

This Thesis addresses the use of IEEE 802.15.4/ZigBee as federating communication protocols for Wireless Sensor Networks. It particularly addresses their timeliness and reliability features in a way that large-scale embedded computing applications with critical Quality of Service (QoS) requirements can be engineered. This chapter overviews the research context and objectives and also outlines the major contributions of this work.

1.1 Introduction

An increasing trend for ubiquitous computing leads to the need of monitoring and controlling everything, everywhere, in a pervasive fashion. The dawn of the next generation of Internet services (e.g. Google Maps [1], Sensor Maps [2]) and the evolution of mobile services along with the gigantic advancement of information and communication technologies (namely on memories, batteries, energy scavenging techniques and hardware designs), and the necessity of large-scale communication infrastructures triggered the birth of the Wireless Sensor Network (WSN) paradigm.

In the upcoming years, wireless communications will be part of everyday objects. The possibilities of wireless communications are unlimited as most entities will intercommunicate and interact. Wireless will be embedded in all objects, from small items such as clothes, mobile phones, gadgets, toys, home appliances, food carts to cars, bridges, roads, farm lands, buildings, animals and people. This computing ubiquity will help improving the quality of life and will change people's habits. While this reality is still at its early days, technologies and applications are evolving at dramatic speeds. General consensus is that there is technological potential for these new applications; the big issue is to make this economically viable as information technology has nothing to lose but its cables. The integration of a wireless module is not just enabling a way to communicate but it is a mean to make objects smarter and granting them new abilities [3].

Many new problems and challenges must be overcome in Wireless Sensor Networks as their paradigm differs from traditional wireless networks. There is the need for low cost devices enabling large-scale applications (as there can be hundreds or thousands of nodes scattered in large areas) and energy requirements that impose low communication rates and ranges and low duty cycles. Some of the challenges in WSNs are energy-efficiency, scalability, routing, mobility, reliability, timeliness, security, clustering, localization and synchronization strategies.

Wireless Sensor Networks enable a wide range of new applications and usages like building automation (e.g. security, HVAC, lighting control, access control), consumer electronics (e.g. TV/VCR/DVD/CD remote control), industrial automation (e.g. asset management, process control, environmental control, energy management) and personal heath care (e.g. heart rate monitoring).

There are a wide range of wireless communication protocol standards (Figure 1) for a wide range of applications (e.g. voice, video and general data communications), everyone of them setting a compromise between bit rate and radio coverage, according to their target application scenarios (personal, local, metropolitan and wide). However there is a need for communication protocols that meet the needs of WSN applications. In general, WSNs do not impose stringent requirements in terms of bandwidth, but they require low energy consumption so that network/nodes lifetime is prolonged as long as possible. In fact, meeting energy requirements are most often the main goal of WSNs protocols and technologies.

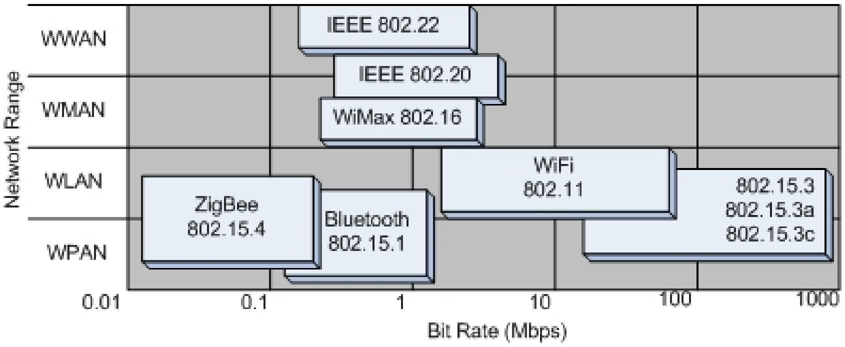


Figure 1 - Wireless protocols overview

Currently, there are a wide range of wireless systems manufacturers that provide a vast number of solutions for WSNs. However, most of these systems are based on proprietary communication protocols, disabling interoperability between technologies from different manufacturers.

The joint efforts of the IEEE 802.15.4 Task Group [4,5] and the ZigBee Alliance [6] have ended up with the specification of a standard protocol stack for Low-Rate Wireless Personal Area Networks (LR-WPANs), an enabling technology for Wireless Sensor Networks (WSNs) [7-8].

The IEEE 802.15.4 protocol [4] specifies the Medium Access Control (MAC) sub-layer and the Physical Layer. Although this protocol was not specifically designed for WSNs, it provides enough flexibility for fitting different requirements of WSN applications by adequately tuning its parameters. The ZigBee specification [5] relies on the IEEE 802.15.4 Physical and Data Link Layers, building up the Network and Application Layers, thus defining a full protocol stack for Low Rate Wireless Personal Area Networks (LR-WPANs). Figure 2 shows the layered architecture of the IEEE 802.15.4/ZigBee protocol stack.

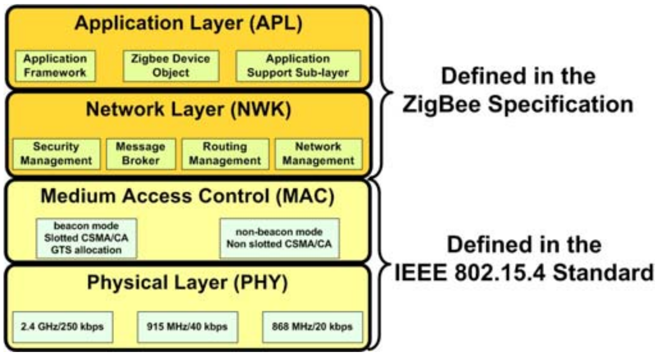


Figure 2 - The IEEE 802.15.4/ZigBee protocol stack architecture

ZigBee [6] is gaining an exponentially increasing interest from industry and academia, considered as a potential solution for low-cost low-power wirelessly connected monitoring and control devices [7-11]. This interest is mainly driven by a large number of emerging applications, including domotics (as the current principal commercial target of the ZigBee Alliance), health care monitoring, industrial automation, environmental monitoring and surveillance. The reputation of ZigBee, even though not widely commercially available yet, is closely related to the objectives for which it was designed and to its flexibility to fit different network and application requirements. While it was designed for low-cost and LR-WPANs, ZigBee is able to provide low energy consumption and real-time guarantees, which creates an eagerness for its application to WSNs. This was the major motivation for this Thesis.

1.2 Research Context

This work has been developed within the context of the ART-WiSE (Architecture for Real-Time communications in Wireless Sensor networks) research framework [12-14] aiming at the specification of a scalable two-tiered communication architecture for improving the timing and reliability behaviour of WSNs. One of the major goals is relying as far as possible on existing standard communication protocols and commercial-off-the-shelf (COTS) technologies – IEEE 802.15.4/ZigBee for Tier 1 and IEEE 802.11 for Tier 2. The reason for using standard technologies pushed forward by commercial manufacturers is that they can speed-up a wider utilization of WSNs.

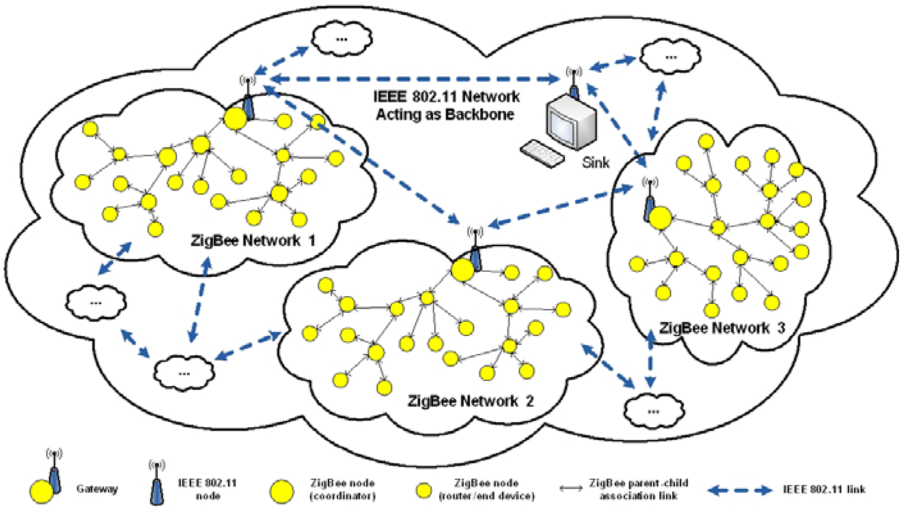


Figure 3 - Example of the ART-WiSe two-tiered architecture

The ART-WiSe architecture is based on a two-tiered network structure (Figure 3) where a wireless network in Tier 2 serves as a backbone for a WSN in Tier 1.

- Tier-2 is an IEEE 802.11 [15] compliant network acting as a backbone for the underlying sensor network. It is composed of a scalable set of special nodes called ZigBee Gateways (ZG), which act as interfaces between the two tiers. Each ZG must include a ZigBee Coordinator (ZC) that will manage the underlying ZigBee Network.
- Tier-1 is an IEEE 802.15.4 compliant WSN interacting with the physical environment (e.g. to collect sensory data). This WSN is partitioned into several independent ZigBee Networks, each of them managed by one ZG (including a ZC). Each ZigBee Network may still be structured into multiple clusters, whenever the density/location of the ZGs does not provide direct radio coverage for the WSN nodes.

The ZigBee Gateway (ZG) [16], as well as the communication protocol stacks in the two tiers must be capable of fulfilling some pre-defined requirements. We believe that both protocol stacks, the IEEE 802.15.4/ZigBee in Tier 1 and the IEEE 802.11/WiFi with the IEEE 802.11e [17] extension in Tier 2 are good candidates to meet the ART-WiSe requirements [12], namely:

- *Real-Rime Performance*, the additional cost in terms of hardware, development, deployment and maintenance must be vindicated by guaranteeing improved timing performance and real-time communications for time-critical messages;
- *Reliability*, since the Tier 1 nodes are less power-constrained, support higher data rate, feature more memory, processing and radio coverage, they are less error-prone than simple sensor nodes (Tier 1);
- *Scalability,* all the network architecture must scale, both at Tier 1 and Tier 2 levels;

- *Mobility*, the architecture must support mobility of sensor nodes, groups of nodes, clusters and ZGs;
- *Cost-Effectiveness*, the two-tiered architecture must be cost-effective as the additional cost of hardware, deployment, development and maintenance, must be as low as possible and be compensated by the increase in QoS.
- *Energy-efficiency*, nodes and network lifetime must be appropriate for the target application, important trade-offs must be found, e.g. reliability, timeliness, scalability and mobility.

This must be archived in a transparent way and in order that the application requirements are always respected.

1.3 Research Objectives

The main objective of this Thesis is to *evaluate the adequateness of IEEE 802.15.4/ZigBee standard protocols for Wireless Sensor Networks*. The hypothesis is that these protocols are appropriate and their mechanisms can in fact support large-scale WSNs in spite of some open issues and ambiguities in their standard specifications.

For this purpose, the implementation of these protocols is an important step allowing a better understanding of the protocol behaviours and opening way for testing, evaluating and improving the standard protocols by proposing adequate solutions that overcome some of the referred problems.

1.4 Research Contributions

The main research contributions of this work are:

1. The implementation [18,19] of the IEEE 802.15.4 standard protocol and the Network Layer of the ZigBee standard for nesC/TinyOS [20,21], supporting the Crossbow MICAz [22] and TelosB [23] motes. Importantly, this open-source implementation has been and will continue to potentiate research works on the IEEE 802.15.4/ZigBee protocols and WSNs in general, allowing their demonstration and validation through experimentation. In this context, the Open-ZB website [24] offers an open-source toolset for these protocols, namely the referred protocol stack implementation and an OPNET [25] simulation model [26].

   The most relevant issues concerning the software architecture were published in [27] and are presented in Chapter 3.
2. Collaboration in the design, implementation and experimental validation of a collision-free beacon scheduling mechanism for ZigBee Cluster-Tree network topologies - the Time Division Beacon Scheduling (TDBS). The TDBS approach enables the engineer of synchronized cluster-tree WSNs, where each cluster may operate at different duty-cycles, thus prolonging network lifetime. [28,29].

   This issue was addressed in [28] and is presented in Chapter 4.

3. Collaboration in the design of fault-tolerance mechanisms tackling ZigBee Router failures in ZigBee Cluster-Tree networks. The proposed mechanism improve on the default behaviour by reducing or even eliminating network inaccessibility times and is backward compatible with the IEEE 802.15.4/ZigBee standards.
   The proposal was outlined in [30] and further elaborated in Chapter 5.
4. Collaboration in the design, implementation and experimental validation of an implicit Guaranteed Time Slot (GTS) allocation mechanism (i-GAME) for the IEEE 802.15.4 protocol that enables an improved bandwidth utilization [31,32].
   This work was published at [31] and is described in Chapter 6.
5. Collaboration in the performance evaluation of the IEEE 802.15.4 Slotted CSMA/CA, comparing experimental results with the ones obtained from the IEEE 802.15.4 simulation model.
   This work has been partially presented in [27] and is described in Chapter 7.
6. Contribution to the specification of the ART-WiSe architecture, as already outlined in this chapter and in [14,16].

1.5 Structure of this Thesis

This Thesis is structured as follows. Chapter 2 provides an overview of the most relevant aspects of the IEEE 802.15.4 and ZigBee protocols. Chapter 3 presents the technological context and the development tools used in the protocol stack implementation, including issues such as the hardware platforms, programming language, operating system and network analysers. This chapter also addresses the software architecture of the protocol stack giving an insight on the implementation strategies that were adopted.

ZigBee Cluster-Tree networks are addressed in Chapter 4. This chapter outlines existing problems and open issues in synchronized cluster-tree topologies and presents a collision-free beacon superframe scheduling mechanism.

Chapter 5 focuses on fault-tolerance mechanisms for ZigBee Cluster-Tree networks, as they are prone to the single point of failure problem. This chapter analyzes common problems associated to ZigBee Routers blackout or link degradation, describing a proactive fault-tolerance mechanism and presenting some implementation guidelines for integrating these add-ons in ZigBee.

An implicit Guaranteed Time Slot allocation mechanism (i-GAME) is described in Chapter 6. This chapter highlights how i-GAME overcomes the limitations of GTS bandwidth utilization and focuses on its implementation, experimental evaluation and validation.

Chapter 7 addresses the evaluation of the IEEE 802.15.4 CSMA/CA mechanism, comparing simulation and experimental results. This chapter enables the validation of the implemented mechanism.

The Thesis concludes with Chapter 8, which summarizes the presented contributions and identifies topics for future research.

Chapter 2

Federating Communication Protocols

This chapter overviews the most relevant aspects of the ZigBee and IEEE 802.15.4 protocol standard. The highlighted features of the ZigBee standard are in the Network Layer as it is the only layer implemented (as described in Chapter 3). This chapter presents the most important features of the IEEE 802.15.4 protocol and ZigBee protocols. It particularly focuses on the IEEE 802.15.4 Data Link and ZigBee Network Layers, which are the most relevant in the context of this Thesis.

2.1 ZigBee Protocol Standard

2.1.1 Overview

ZigBee defines two layers of the OSI (Open Systems Interconnection) model: the Application Layer (APL) and the Network Layer (NWL), as depicted in Figure 4. Each layer performs a specific set of services for the layer above. The different layers communicate through Service Access Points (SAP's). These SAPs enclose two types of entities: (1) a data entity (NLDE-SAP) to provide data transmission service and (2) a management entity (NLME-SAP) providing all the management services between layers.

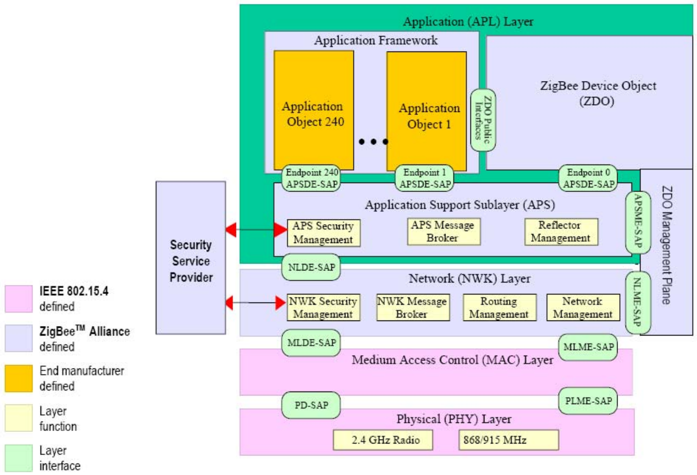


Figure 4 - ZigBee architecture [6]

The application layer (APL) is the top layer and includes the Application Support Sub-layer (APS), the ZigBee Device Objects (ZDO) and the Application Framework (AF), containing application specific implemented objects. The interactions between the APL and the APS are accomplished via *EndPoints*, each of them including a set of interfaces (APSDE-SAP) to support data transmissions between layers. The Application Support SubLayer (APS) provides two sets of services: (1) the APS Management Entity Service Access Point (APSME-SAP) used by the ZDO of Coordinators to retrieve information from the APS layer and to implement security, and (2) the APS Data Entity Service Access Point (APSDE-SAP) used by the AF and the ZDO to exchange data with the APS. The ZigBee Device Object (ZDO) provides and interfaces to the AF used for discovering other devices and the application objects provided services or EndPoints. The ZDO is also responsible for communicating information about itself and its provided services. The ZDO is located in *EndPoint 0*. The Application Objects are the manufacturer's applications running on top of the ZigBee protocol stack. These objects, located between Endpoints 1 to 240, adhere to a given profile approved by the ZigBee Alliance. The address of the device and the *EndPoints* available provide a uniform way of addressing individual application objects in the ZigBee network. The set of ZDOs, their configuration and functionalities form a ZigBee profile. The ZigBee profiles intent to be an uniform representation of common application scenarios. Currently, the ZigBee available profiles include the Network Specific (stack identifier 0); Home Controls (stack identifier 1); Building Automation (stack identifier 2) and Plant Control (stack identifier 3).

The ZigBee Network Layer (NWK) is responsible for Network management procedures (e.g. nodes joining and leaving the network), security and routing. It also encloses the neighbour tables and the storage of related information. The NWK Layer provides one set of interfaces, the Network Layer Data Entity Service Access Point (NLDE-SAP) used to exchange data with the APS.

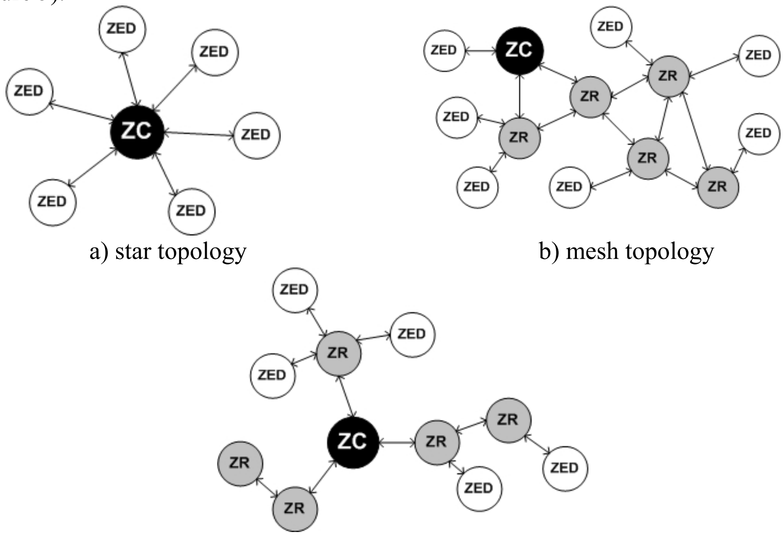
IEEE 802.15.4/ZigBee devices can be classified according to their functionalities: *Full Function Devices* (FFD) implement the full IEEE 802.15.4/ZigBee protocol stack; *Reduced Function Devices* (RFD) implement a subset of the protocol stack.

Regarding the devices role in the network, ZigBee defines 3 types of devices:

- *ZigBee Coordinator* (ZC): One for each ZigBee Network; Initiates and configures Network formation; Acts as an IEEE 802.15.4 Personal Area Network (PAN) Coordinator; Acts as ZigBee Router (ZR) once the network is formed; Is a Full Functional Device (FFD) – implements the full protocol stack; If the network is operating in beacon-enabled mode, the ZC will send periodic beacon frames that will serve to synchronize the rest of the nodes. In a Cluster-Tree network all ZR will receive beacon from their parents and send their own beacons to synchronize nodes belonging to their clusters
- *ZigBee Router* (ZR): Participates in multi-hop routing of messages in mesh and Cluster-Tree networks; Associates with ZC or with previously associated ZR in Cluster-Tree topologies; Acts as an IEEE 802.15.4 PAN Coordinator; Is a Full Functional Device (FFD) – implements the full protocol stack.
- *ZigBee End Device* (ZED): Does not allow other devices to associate with it; Does not participate in routing; It is just a sensor/actuator node; Can be a Reduced Function Device (RFD) – implementing a reduced subset of the protocol stack.

Throughout this Thesis the names of the devices and their acronyms are used interchangeably.

ZigBee/IEEE 802.15.4 enable three network topologies – star, mesh and cluster-tree (Figure 5).



a) star topology

b) mesh topology

c) cluster-tree topology

Figure 5 - ZigBee network topologies

In the star topology (Figure 5 a), a unique node operates as a ZC. The ZC chooses a PAN identifier, which must not be used by any other ZigBee network in the vicinity. The communication paradigm of the star topology is centralized, i.e. each device (FFD or RFD) joining the network and willing to communicate with other devices must send its data to the ZC, which dispatches it to the adequate destination. The star topology may not be adequate for traditional Wireless Sensor Networks for two reasons. First, the sensor node selected as a ZC will get its battery resources rapidly ruined. Second, the coverage of an IEEE 802.15.4/ZigBee cluster is very limited while addressing a large-scale WSN, leading to a scalability problem.

The mesh topology (Figure 5 b) also includes a ZC that identifies the entire network. However, the communication paradigm in this topology is decentralized, i.e. each node can directly communicate with any other node within its radio range. The mesh topology enables enhanced networking flexibility, but it induces additional complexity for providing end-to-end connectivity between all nodes in the network. Basically, the mesh topology operates in an ad-hoc fashion and allows multiple hops to route data from any node to any other node. In contrast with the star topology, the mesh topology may be more power-efficient and the battery resource usage is fairer, since the communication process does not rely on one particular node.

The cluster-tree network topology (Figure 5 c) is a special case of a mesh network where there is a single routing path between any pair of nodes and there is a distributed synchronization mechanism (IEEE 802.15.4 beacon-enabled mode). There is only one

ZC which identifies the entire network and one ZR per cluster. Any of the FFD can act as a ZR providing synchronization services to other devices and ZRs.

The IEEE 802.15.4 frame formats are attached in Annex A and the ZigBee frame formats in Annex B.

2.1.2 ZigBee Network Layer

The ZigBee Network Layer is responsible for network management (e.g. association/disassociation, starting the network, addressing, device configuration and the maintenance of the NIB - NWK Information Base) and formation, message routing and security-related services.

The ZigBee Network Layer supports two service entities. The Network Layer Data Entity (NLDE) provides a data service, allowing the transmission of data frames and topology specific routing. Figure 6 depicts the Network Layer reference model.

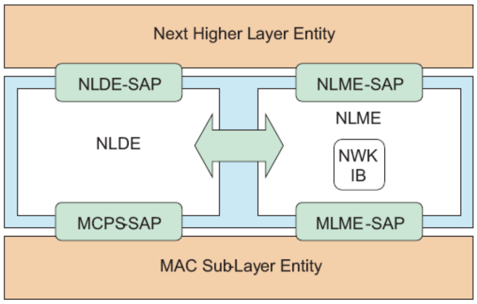


Figure 6 - Network Layer reference model [6]

Joining and leaving a network must be supported by all ZigBee Devices. Both ZigBee Coordinators and Routers must support the following additional functionalities:

- Permit devices to join the network using the following:
    - Association indications from the MAC sub-layer;
    - Explicit join requests from the application.
- Permit devices to leave the network using the following:
    - Network Leave command frames;
    - Explicit leave requests from the application.
- Participate in assignment of logical network addresses.
- Maintain a list of neighbouring devices.

The ZigBee Coordinator also defines some important additional network parameters. It determines the maximum number of children (C_m) any device is allowed to have. From this set of children, a maximum number (R_m) of devices can be router-capable devices. The remaining are ZEDs. Every device has an associated depth, representing the number of hops a transmitted frame must travel, using only a parent-child links, to reach the ZigBee Coordinator. The ZC has a depth of 0, while its children have a depth of 1. The ZC also determines the maximum depth (L_m) of the network. The maximum number of children, routers and network depth are used for calculating the addresses of the devices in the network, in a distributed address scheme.

2.1.3 Short Address Assignment

A parent device uses the C_m, R_m, and L_m values to compute a *Cskip* function defining the size of the address sub-block that is distributed by each parent depending on its depth (*d*) in the network. For a given network depth *d*, *Cskip(d)* is calculated as follows:

$$Cskip(d) = \begin{cases} 1 + Cm \cdot (Lm - d - 1), & if\ Rm = 1 \\ \dfrac{1 + Cm - Rm - Cm \cdot Rm^{Lm-d-1}}{1 - Rm}, & Otherwise \end{cases} \tag{2.1}$$

A parent device that has a *Cskip(d)* value of zero is not capable of accepting children and must be treated as an end device. A parent device that has a *Cskip(d)* value greater that zero must accept devices and assigns addresses if possible. A parent device assigns an address that is one greater than its own to the first router that associated. The next associated router receives an address that is separated according to the return value of the *Cskip(parent depth)* function. The maximum number of associated routers is defined in the network parameter *nwkMaxRouters* (R_m).

Considering a parent node with a depth *d* and an address of A_{parent}, the number of child devices *n* is between 1 and C_m-R_m.

$$1 \le n \le (C_m - R_m) \tag{2.2}$$

The A_{child} address of the n^th^ child router is calculated according to Eq. 2.3(*n* is the number of child routers):

$$\begin{aligned} A_{child} &= A_{parent} + (n-1) \times Cskip(d) + 1, n = 1 \\ A_{child} &= A_{parent} + (n-1) \times Cskip(d), n > 1 \end{aligned} \tag{2.3}$$

The A_{child} address of the n^th^ child end device is calculated according to Eq. 2.4 (*n* is the number of child end devices):

$$A_{child} = A_{parent} + Rm \times Cskip(d) + n \tag{2.4}$$

Figure 7 depicts an example of an address assignment scheme. The parameters used in the address assignment are the following: maximum depth (L_m) = 3, maximum children (C_m) = 6 and maximum routers (R_m) = 4.

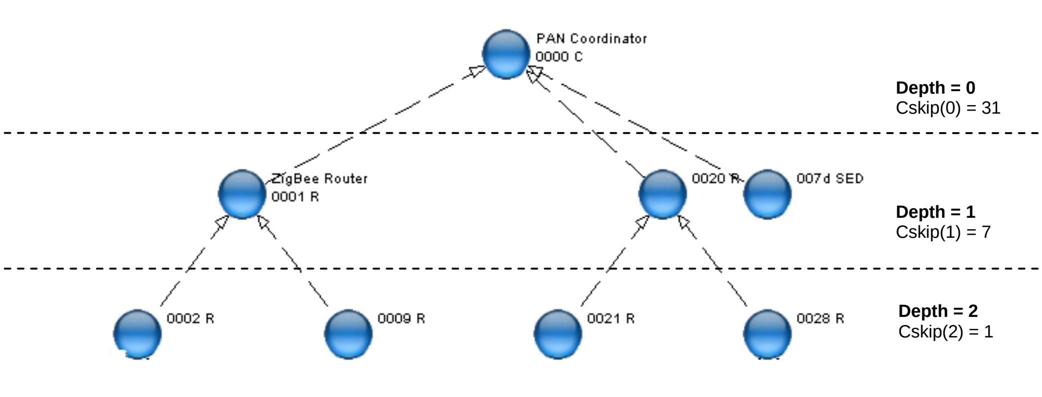
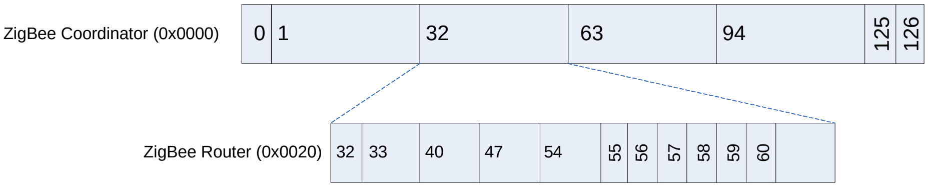


Figure 7 - Address assignment scheme example

Figure 8 shows the ZigBee Coordinator (0x0000) available addressing scheme. Considering the above network parameters, the ZigBee Coordinator is allowed to associate up to A4 routers and 2 end devices in its available address pool. On the other hand, the ZR (0x0020) is allowed to associate up to 4 ZRs and 6 ZEDs.



ZigBee Coordinator (0x0000)	0	1	32	63	94	125	126

ZigBee Router (0x0020)	32	33	40	47	54	55	56	57	58	59	60	

Figure 8 - ZigBee Coordinator addressing scheme (decimal values)

2.1.4 ZigBee Routing

ZigBee Coordinators and Routers must provide the following functionalities:

- Relay data frames on behalf of higher layers;
- Relay data frames on behalf of other ZR;
- Participate in route discovery in order to establish routes for subsequent data frames;
- Participate in route discovery on behalf of end devices;
- Participate in end-to-end route repair;
- Participate in local route repair;
- Employ the ZigBee path cost metric as specified in route discovery and route repair.

Additionally, ZigBee Coordinators and Routers may provide the following functionalities:

- Maintain routing tables in order to remember best available routes;
- Initiate route discovery on behalf of higher layers;
- Initiate route discovery on behalf of other ZR;
- Initiate end-to-end route repair;
- Initiate local route repair on behalf of other ZR.

2.1.5 Routing Schemes

ZigBee Coordinators and Routers support three types of routing:

- *Neighbour Routing* – based on a neighbour tables that contains the information of all the devices within radio coverage. If the target device is physically in range the message can be sent directly. Note that ZEDs cannot do this.
- *Table Routing* - Ad-hoc On Demand Distance Vector (AODV) [33], based on routing and route discovery tables with the path cost metrics;
- *Tree-Routing* - based on the address assignment schemes; messages are hierarchically routed upstream/downstream the tree.

Neighbour Routing

This type of routing uses the neighbour tables. If the target device is physically in range it is possible to send messages directly to the destination. Physically in range means that the source ZC or ZR has a neighbour table entry for the destination. This routing mechanism is mostly used as addition to other routing mechanisms and for the ZigBee Routers to route messages to its children devices, when they are the destination.

Table Routing - Ad-hoc On-Demand Distance Vector (AODV)

ZigBee Table Routing is based on the AODV routing algorithms. Each ZigBee Coordinator and Router that supports this Table Routing must maintain two tables: (1) the routing table, a long-lived and persistent table with the information of routes, and (2) a route discovery table with the information of the route discovery procedures where each entry only lasts the duration of the discovery.

The Ad-hoc On Demand Distance Vector (AODV) [33] routing protocol was designed for ad hoc mobile networks. AODV is capable of both unicast and multicast routing. AODV allows mobile nodes to obtain routes quickly for new destinations, and does not require nodes to maintain routes to destinations that are not in active communication. AODV allows mobile nodes to respond to link breakages and changes in network topology in a timely manner. The operation of AODV is loop-free, and by avoiding the Bellman-Ford "counting to infinity" problem offers quick convergence when the ad-hoc network topology changes (typically, when a node moves in the network). When the link breaks, AODV causes the affected set of nodes to be notified so that they are able to invalidate the routes using the lost link. It is an on demand algorithm, meaning that it builds routes between nodes only if requested by source nodes. It maintains these routes as long as they are needed by the sources. Additionally, AODV can form trees, connecting multicast groups, composed of the group members and the nodes needed to connect. AODV uses sequence numbers to ensure the freshness of routes. It is loop-free, self-starting, and scales to larger numbers of nodes.

In ZigBee Networks, the routing management is done by the means of NWK command frames (refer to Figure B.3 in Annex B for the NWK command frame format). The available commands are the following:

- *Route request* – Command send to search for a route to a specified device, can also be used to repair a route
- *Route reply* – Command send in response of a route request, also used to request state information
- *Route Error* – notification of a source device of the data frame about the failure in forwarding the frame:
- *Leave* – notification of a device leaving the network
- *Route Record* – notification of a list of nodes used in relaying a data frame
- *Rejoin request* – notification of a device rejoining the network
- *Rejoin response* – rejoin response of a rejoin request

The route choice for a communication flow is based on the total link cost represented by *C*, meaning that the path with the lowest cost is chosen. The total link cost is the sum of individual point-to-point link cost.

The calculation of *C* is as follows: for a defined path *P* where *L* defines the length of a set of devices $[D_1, D_2, \ldots D_L]$ and a link $[D_i, D_{i+1}]$ the path cost *C* is defined as:

$$C\{P\} = \sum_{i=1}^{L-1} C\{[D_1, D_{i+1}]\} \tag{2.5}$$

each $C\{[D_1, D_{i+1}]\}$ is the individual point-to-point link cost, calculated by the following formulation:

$$C\{l\} = \begin{cases} 7, \\ \min\left(7, round\left(\frac{1}{p_l^4}\right)\right) \end{cases} \tag{2.6}$$

where p_l is defined as the probability of packet delivery through link *l.*

The link probability estimation factors are implementation specific, but generally it they are based on the counting of the received beacons and data frames in order to detect packet loss and in the estimation of the Link Quality Indicator (LQI).

Tree-Routing

This routing mechanism is based on the short addressing scheme and was initially proposed by MOTOROLA [34]. Each device, upon the reception of a data frame, reads the routing information fields (refer to Figure B.1 in Annex B) and checks the destination address. If the destination is a child of the device (neighbour table check), the device relays the packet to the appropriate address. If the destination address is not a child, the device must check if the address is a descendent using the condition in 2.7, where *A* is device network address, *D* the destination address and *d* the device depth in the network.

$$A < D < A + Cskip(d - 1) \tag{2.7}$$

The next hop (*N*) address when routing down is given by:

$$N = A + 1 + \left\lfloor \frac{D - (A+1)}{Cskip(d)} \right\rfloor \times Cskip(d) \tag{2.8}$$

If the destination address is not a descendant, the device relays the packet to its parent.

Consider the network scenario illustrated in Figure 7 and the following network parameters: $L_m = 3$; $C_m = 6$; $R_m = 4$. The *Cskip* values are presented in Table 1.

Table 1 - Cskip example values

Depth	Cskip(Depth)
0	31
1	7
2	1

If ZR 0x0002 transmits a message to ZR 0x0028, the tree-routing protocol behaves as follows:

1. ZR 0x0002 builds the data frame and sends it to its parent (0x0001). The most relevant fields of this data frame are outlined next:
    - MAC destination address – 0x0001;
    - MAC source address – 0x0002;
    - Network Layer Routing Destination Address – 0x0028;
    - Network Layer Routing Source Address – 0x0002;

2. ZR 0x0001 receives the data frame, realizes that the message in not for him and has to be relayed. The device checks its neighbour table for the routing destination address, trying to find if the destination is one of its child devices. Then, the device checks if the routing destination address is a descendant by verifying condition in 2.7 that results in:

0x0001 < 0x0028 < 0x0001 + 7

Note that ZR 0x0001 is a depth 1 device in the network. After verifying that the destination is not a descendant, ZR 0x0001 routes the data frame to its parent, ZC 0x0000. The most relevant fields of this data frame are outlined next:

- MAC destination address – 0x0000;
- MAC source address – 0x0001;
- Network Layer Routing Destination Address – 0x0028;
- Network Layer Routing Source Address – 0x0002;

3. ZC 0x0000 receives the data frame and verifies if the routing destination address exists in its neighbour table. After realizing that the destination device is not its neighbour, since the ZC is the root of the tree and cannot route up, the next hop address is calculated as follows:

$$N = 0x0000 + 1 + \left\lfloor \frac{0x0028 - (0x0000 + 1)}{31} \right\rfloor \times 31$$

The next hop address results in N = 32 (decimal) = 0x0020. The most relevant fields of this data frame are outlined next:

- MAC destination address – 0x0020;
- MAC source address – 0x0000;
- Network Layer Routing Destination Address – 0x0028;
- Network Layer Routing Source Address – 0x0002;

4. ZR 0x0020 receives the data frame and checks its neighbour table for the routing destination address. After verifying that the address is its neighbour, the message is routed to it. The next hop is assigned with the short address present in the respective neighbour table entry. The most relevant fields of this data frame are outlined next:

- MAC destination address – 0x0028;
- MAC source address – 0x0020;
- Network Layer Routing Destination Address – 0x0028;
- Network Layer Routing Source Address – 0x0002;

2.2 IEEE 802.15.4 Protocol Standard

The IEEE 802.15.4 Full Function Devices (FFD) have three different operation modes:

- The *Personal Area Network (PAN) Coordinator*: the principal controller of the PAN. This device identifies its own network as well as its configurations, to which other devices may be associated. In ZigBee, this device is referred to as the ZigBee Coordinator (ZC).

- The *Coordinator*: provides synchronization services through the transmission of beacons. This device should be associated to a PAN Coordinator and does not create its own network. In ZigBee, this device is referred to as the ZigBee Router (ZR).
- The *End Device*: a device which does not implement the previous functionalities and should associate with a ZC or ZR before interacting with the network. In ZigBee, this device is referred to as the ZigBee End Device (ZED).

The Reduced Function Device (RFD) is an end device operating with the minimal implementation of the IEEE 802.15.4. An RFD is intended for applications that are extremely simple, such as a light switch or a passive infrared sensor; they do not have the need to send large amounts of data and may only associate with a single FFD at a time.

Throughout this Thesis the IEEE 802.14.5 operational modes and the ZigBee device names are used interchangeably (e.g. PAN Coordinator = ZigBee Coordinator, Coordinator = ZigBee Router and End Device = ZigBee End Device). The designation of Coordinator represents both ZC and ZRs.

2.2.1 Physical Layer

The IEEE 802.15.4 physical layer is responsible for data transmission and reception using a certain radio channel and according to a specific modulation and spreading technique.

The IEEE 802.15.4 offers three operational frequency bands: 2.4 GHz, 915 MHz and 868 MHz. There is a single channel between 868 and 868.6 MHz, 10 channels between 902 and 928 MHz, and 16 channels between 2.4 and 2.4835 GHz (see Figure 9). The protocol also allows dynamic channel selection, a channel scan function in search of a beacon, receiver energy detection, link quality indication and channel switching.

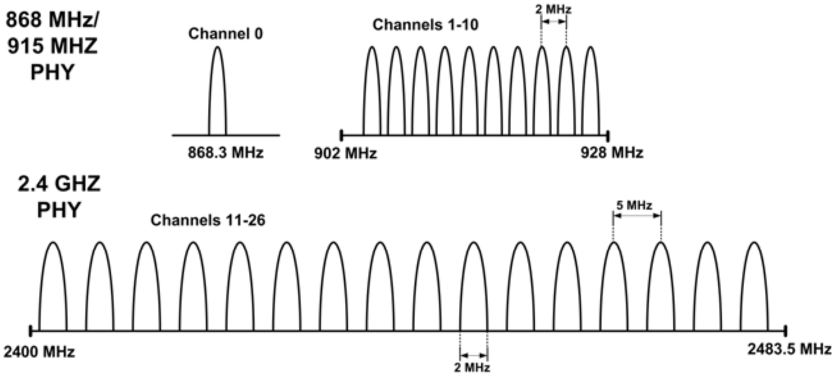
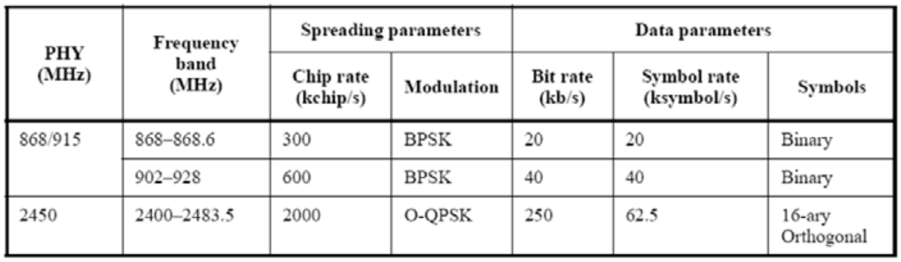


Figure 9 - Operating frequencies and bands [4]

All of these frequency bands are based on the Direct Sequence Spread Spectrum (DSSS) spreading technique. The features of each frequency band (e.g. modulation, chip rate, bit rate) are summarized in Table 2.

Table 2 - IEEE 802.15.4 frequency bands and data rates [4]



PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
868/915	868–868.6	300	BPSK	20	20	Binary
	902–928	600	BPSK	40	40	Binary
2450	2400–2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

The physical layer of IEEE 802.15.4 is in charge of the following tasks:

- *Activation and deactivation of the radio transceiver*: The radio transceiver may operate in one of three states: transmitting, receiving or sleeping. Upon request of the MAC sub-layer, the radio is turned *ON* or *OFF*. The turnaround time from transmitting to receiving and vice versa should be no more than 12 symbol periods, according to the standard (each symbol corresponds to 4 bits).
- *Energy Detection* (ED): Estimation of the received signal power within the bandwidth of an IEEE 802.15.4 channel. This task does not make any signal identification or decoding on the channel. The energy detection time should be equal to 8 symbol periods. This measurement is typically used by the Network Layer as a part of channel selection algorithm or for the purpose of Clear Channel Assessment (CCA), to determine if the channel is busy or idle.
- *Link Quality Indication* (LQI): Measurement of the Strength/Quality of a received packet. It measures the quality of a received signal. This measurement may be implemented using receiver ED, a signal to noise estimation or a combination of both techniques.
- *Clear Channel Assessment* (CCA): Evaluation of the medium activity state: busy or idle. The CCA is performed in three operational modes: (1) Energy Detection mode: the CCA reports a busy medium if the detected energy is above the ED threshold. (2) Carrier Sense mode: the CCA reports a busy medium only is it detects a signal with the modulation and the spreading characteristics of IEEE 802.15.4 and which may be higher or lower than the ED threshold. (3) Carrier Sense with Energy Detection mode: this is a combination of the aforementioned techniques. The CCA reports that the medium is busy only if it detects a signal with the modulation and the spreading characteristics of IEEE 802.15.4 and with energy above the ED threshold.
- *Channel Frequency Selection*: The IEEE 802.15.4 defines 27 different wireless channels. Each network can support only part of the channel set. Hence, the physical layer should be able to tune its transceiver into a specific channel when requested by a higher layer.

2.2.2 Medium Access Control (MAC) Sub-layer

The MAC protocol supports two operational modes (Figure 10):

- **The non beacon-enabled mode**. When the ZC selects the non-beacon enabled mode, there are neither beacons nor superframes. Medium access is ruled by an unslotted CSMA/CA mechanism (refer to Section 2.2.6).
- **The beacon-enabled mode**. In this mode, beacons are periodically sent by the ZC or ZR to synchronize nodes that are associated with it, and to identify the PAN. A beacon frame delimits the beginning of a superframe (refer to Section 2.2.3) defining a time interval during which frames are exchanged between different nodes in the PAN. Medium access is basically ruled by Slotted CSMA/CA. However, the beacon-enabled mode also enables the allocation of contention free time slots, called Guaranteed Time Slots (GTSs) for nodes requiring guaranteed bandwidth.

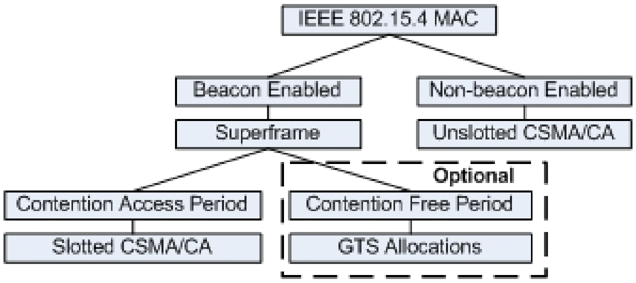


Figure 10 - IEEE 802.15.4 Operational Modes

2.2.3 Superframe Structure

The superframe is defined between two beacon frames (Figure A.2 in Annex A.2 depicts the beacon frame format), and has an active period and an inactive period. Figure 11 depicts the IEEE 802.15.4 superframe structure.

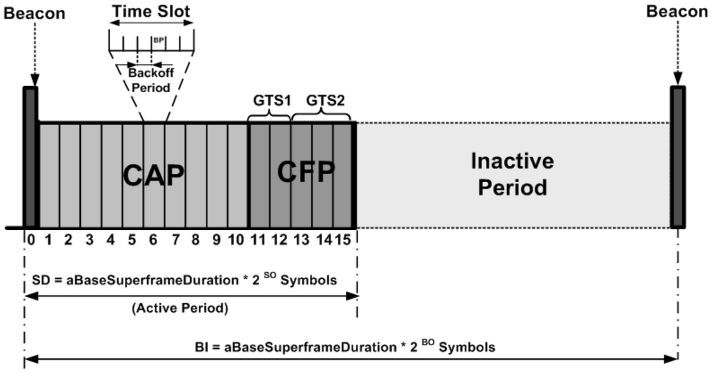


Figure 11 - IEEE 802.15.4 Superframe Structure [4]

The active portion of the superframe structure is composed of three parts, the *Beacon*, the *Contention Access Period* (CAP) and the *Contention Free Period* (CFP):

- *Beacon*: the beacon frame is transmitted at the start of slot 0. It contains the information on the addressing fields, the superframe specification, the GTS fields, the pending address fields and other PAN related.
- *Contention Access Period (CAP):* the CAP starts immediately after the beacon frame and ends before the beginning of the CFP, if it exists. Otherwise, the CAP ends at the end of the active part of the superframe. The minimum length of the CAP is fixed at *aMinCAPLength = 440 symbols*. This minimum length ensures that MAC commands can still be transmitted when GTSs are being used. A temporary violation of this minimum may be allowed if additional space is needed to temporarily accommodate an increase in the beacon frame length, needed to perform GTS management. All transmissions during the CAP are made using the Slotted CSMA/CA mechanism. However, the acknowledgement frames and any data that immediately follows the acknowledgement of a data request command are transmitted without contention. If a transmission cannot be completed before the end of the CAP, it must be deferred until the next superframe.
- *Contention Free Period (CFP):* The CFP starts immediately after the end of the CAP and must complete before the start of the next beacon frame (if *BO* equals *SO*) or the end of the superframe. Transmissions are contention-free since they use reserved time slots (GTS) that must be previously allocated by the ZC or ZR of each cluster. All the GTSs that may be allocated by the Coordinator are located in the CFP and must occupy contiguous slots. The CFP may therefore grow or shrink depending on the total length of all GTSs.

In beacon-enabled mode, each Coordinator defines a superframe structure Figure 11 which is constructed based on:

- The *Beacon Interval* (*BI*), which defines the time between two consecutive beacon frames;
- The *Superframe Duration* (*SD*), which defines the active portion in the *BI*, and is divided into 16 equally-sized time slots, during which frame transmissions are allowed.

Optionally, an inactive period is defined if *BI* > *SD*. During the inactive period (if it exists), all nodes may enter in a sleep mode (to save energy). *BI* and *SD* are determined by two parameters, the Beacon Order (*BO*) and the Superframe Order (*SO*), respectively, as follows:

$$\left.\begin{array}{l} BI = aBaseSuperframeDuration \times 2^{BO} \\ SD = aBaseSuperframeDuration \times 2^{SO} \end{array}\right\} for\ 0 \leq SO \leq BO \leq 14 \qquad (2.9)$$

aBaseSuperframeDuration = 15.36 ms (assuming 250 kbps in the 2.4 GHz frequency band) denotes the minimum duration of the superframe, corresponding to *SO=0*.

As depicted in Figure 11, low duty cycles can be configured by setting small values of the *SO* as compared to *BO*, resulting in greater sleep (inactive) periods. In ZigBee Cluster-Tree networks, each cluster can have different and dynamically adaptable duty-cycles. This feature is particularly interesting for WSN applications, where energy consumption and network lifetime are main concerns. Additionally, the Guaranteed Time Slot (GTS) mechanism, is quite attractive for time-sensitive WSNs, since it is possible to guarantee end-to-end message delay bounds both in Star [73] and Cluster-Tree [74] topologies.

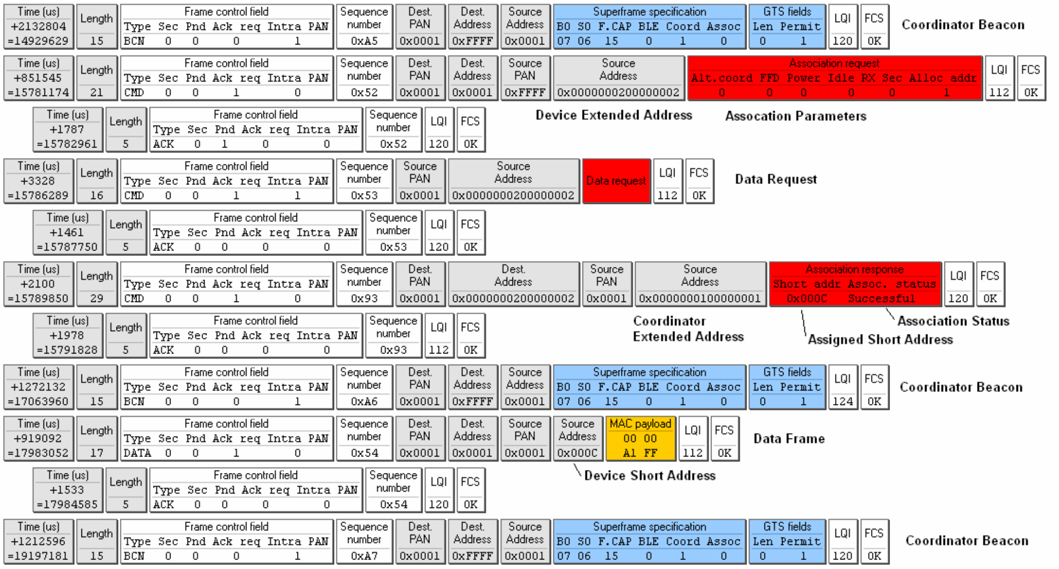
2.2.4 Association and Channel Scan Mechanisms

The association procedure takes place when a device wants to associate with a Coordinator. This mechanism can be divided into three separate phases: (1) channel scan procedure; (2) selection of a possible parent; (3) association with the parent.

IEEE 802.15.4 enables four types of channel scan procedures: (1) the *energy detection* scan, where the device obtains a measure of the peak energy in each channel; (2) the *active scan*, where the device locates all Coordinators transmitting beacon frames; this scan is performed on each channel by first transmitting a beacon request command; (3) the *passive scan*, where similarly to the active scan, the device locates all Coordinator transmitting beacon frames with the difference that the scan is performed only in a receive mode, without transmitting beacon requests; and (4) the *orphan scan*, used to locate the Coordinator with which the scanning device had previously associated.

After the channel scan procedure is completed, the NWK layer receives a list of all detected PAN descriptors (containing information about the potential parents). Based on the information collected during the scan, the device can choose the most suitable parent (that permits associations). The IEEE 802.15.4 protocol standard leaves the way to take the association decision to the system designer. Nevertheless one of the most relevant parameters to be considered is the Link Quality Indicator (LQI).

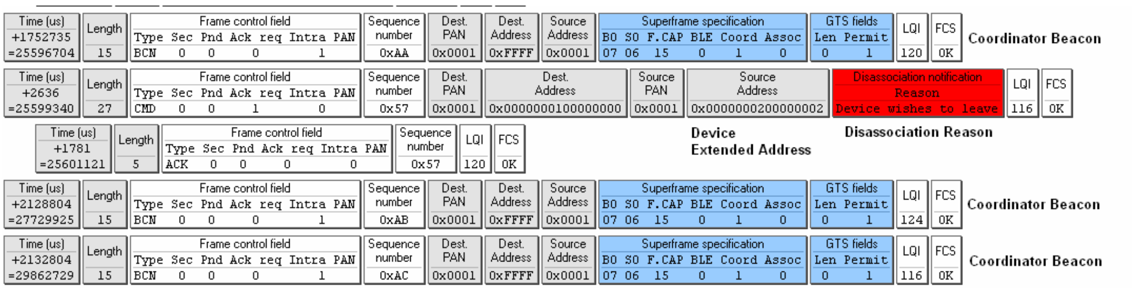
For a device to associate to a Coordinator, it must send an association command frame (Figure A.7 in Annex A). Then, if the Coordinator accepts the device, it adds it to its neighbour table as its child. An association response command frame (as depicted in Figure A.8 in Annex A) is, in the case of a successful association, sent to the device (via an indirect transmission, refer to Section 2.2.8), embedding its short address. Otherwise, in the case of an unsuccessful association, the association response embeds the problem status information. The Coordinator replies to the association command frame with an acknowledgment embedding the pending data control flag active, meaning that it has data ready to be transmitted to the device. The association procedure is completed when the device sends a data request command frame (Figure A.10 in Annex A) to the Coordinator requesting the pending data (the association response command). After a successful association, the device stores all the information about the new PAN by updating its MAC PAN Information Base (MAC PIB) and can start transmissions. Figure 12 exemplifies the sequence of messages for a successful association request, followed by a data transmission.



```
Time (us)  Length  Frame control field              Sequence  Dest.   Dest.    Source   Superframe specification       GTS fields   LQI  FCS
+2132804           Type Sec Pnd Ack req Intra PAN   number    PAN     Address  Address  BO SO F.CAP BLE Coord Assoc    Len Permit
=14929629  15      BCN  0   0   0       1           0xA5      0x0001  0xFFFF   0x0001   07 06 15    0   1     0        0   1        120  OK   Coordinator Beacon

Time (us)  Length  Frame control field              Sequence  Dest.   Dest.    Source   Source              Association request                            LQI  FCS
+851545            Type Sec Pnd Ack req Intra PAN   number    PAN     Address  PAN      Address             Alt.coord FFD Power Idle RX Sec Alloc addr
=15781174  21      CMD  0   0   1       0           0x52      0x0001  0x0001   0xFFFF   0x0000000200000002  0         0   0          0   0        1     112  OK
                                                                                        Device Extended Address   Assocation Parameters

   Time (us)  Length  Frame control field              Sequence  LQI  FCS
   +1787              Type Sec Pnd Ack req Intra PAN   number
   =15782961  5       ACK  0   1   0       0           0x52      120  OK

Time (us)  Length  Frame control field              Sequence  Source  Source              Data request  LQI  FCS
+3328              Type Sec Pnd Ack req Intra PAN   number    PAN     Address
=15786289  16      CMD  0   0   1       1           0x53      0x0001  0x0000000200000002                112  OK   Data Request

   Time (us)  Length  Frame control field              Sequence  LQI  FCS
   +1461              Type Sec Pnd Ack req Intra PAN   number
   =15787750  5       ACK  0   0   0       0           0x53      120  OK

Time (us)  Length  Frame control field              Sequence  Dest.   Dest.               Source  Source              Association response       LQI  FCS
+2100              Type Sec Pnd Ack req Intra PAN   number    PAN     Address             PAN     Address             Short addr Assoc. status
=15789850  29      CMD  0   0   1       0           0x93      0x0001  0x0000000200000002  0x0001  0x0000000100000001  0x000C     Successful     120  OK
                                                                                                  Coordinator         Assigned Short Address
                                                                                                  Extended Address    Association Status

   Time (us)  Length  Frame control field              Sequence  LQI  FCS
   +1978              Type Sec Pnd Ack req Intra PAN   number
   =15791828  5       ACK  0   0   0       0           0x93      112  OK

Time (us)  Length  Frame control field              Sequence  Dest.   Dest.    Source   Superframe specification       GTS fields   LQI  FCS
+1272132           Type Sec Pnd Ack req Intra PAN   number    PAN     Address  Address  BO SO F.CAP BLE Coord Assoc    Len Permit
=17063960  15      BCN  0   0   0       1           0xA6      0x0001  0xFFFF   0x0001   07 06 15    0   1     0        0   1        124  OK   Coordinator Beacon

Time (us)  Length  Frame control field              Sequence  Dest.   Dest.    Source   Source   MAC payload  LQI  FCS
+919092            Type Sec Pnd Ack req Intra PAN   number    PAN     Address  PAN      Address  00 00
=17983052  17      DATA 0   0   1       0           0x54      0x0001  0x0001   0x0001   0x000C   A1 FF        112  OK   Data Frame
                                                                                        Device Short Address

   Time (us)  Length  Frame control field              Sequence  LQI  FCS
   +1533              Type Sec Pnd Ack req Intra PAN   number
   =17984585  5       ACK  0   0   0       0           0x54      120  OK

Time (us)  Length  Frame control field              Sequence  Dest.   Dest.    Source   Superframe specification       GTS fields   LQI  FCS
+1212596           Type Sec Pnd Ack req Intra PAN   number    PAN     Address  Address  BO SO F.CAP BLE Coord Assoc    Len Permit
=19197181  15      BCN  0   0   0       1           0xA7      0x0001  0xFFFF   0x0001   07 06 15    0   1     0        0   1        120  OK   Coordinator Beacon
```

Figure 12 - Association mechanism example.

The disassociation from a Coordinator is done via a disassociation request command (Figure A.9 in Annex A). The disassociation can be initiated either by the device or by the Coordinator. After the disassociation procedure, the device loses its short address and is not able to communicate. The Coordinator updates the list of associated devices, but it can still keep the device information for a future re-association. Figure 13 shows a transmission sequence of a disassociation request initiated by a device.



```
Time (us)  Length  Frame control field              Sequence  Dest.   Dest.    Source   Superframe specification       GTS fields   LQI  FCS
+1752735           Type Sec Pnd Ack req Intra PAN   number    PAN     Address  Address  BO SO F.CAP BLE Coord Assoc    Len Permit
=25596704  15      BCN  0   0   0       1           0xAA      0x0001  0xFFFF   0x0001   07 06 15    0   1     0        0   1        120  OK   Coordinator Beacon

Time (us)  Length  Frame control field              Sequence  Dest.   Dest.               Source  Source              Disassociation notification  LQI  FCS
+2636              Type Sec Pnd Ack req Intra PAN   number    PAN     Address             PAN     Address             Reason
=25599340  27      CMD  0   0   1       0           0x57      0x0001  0x0000000100000000  0x0001  0x0000000200000002  Device wishes to leave       116  OK
                                                                                                  Device              Disassociation Reason
                                                                                                  Extended Address

   Time (us)  Length  Frame control field              Sequence  LQI  FCS
   +1781              Type Sec Pnd Ack req Intra PAN   number
   =25601121  5       ACK  0   0   0       0           0x57      120  OK

Time (us)  Length  Frame control field              Sequence  Dest.   Dest.    Source   Superframe specification       GTS fields   LQI  FCS
+2128804           Type Sec Pnd Ack req Intra PAN   number    PAN     Address  Address  BO SO F.CAP BLE Coord Assoc    Len Permit
=27729925  15      BCN  0   0   0       1           0xAB      0x0001  0xFFFF   0x0001   07 06 15    0   1     0        0   1        124  OK   Coordinator Beacon

Time (us)  Length  Frame control field              Sequence  Dest.   Dest.    Source   Superframe specification       GTS fields   LQI  FCS
+2132804           Type Sec Pnd Ack req Intra PAN   number    PAN     Address  Address  BO SO F.CAP BLE Coord Assoc    Len Permit
=29862729  15      BCN  0   0   0       1           0xAC      0x0001  0xFFFF   0x0001   07 06 15    0   1     0        0   1        116  OK   Coordinator Beacon
```

Figure 13 - Dissassociation mechanism example

2.2.5 Guaranteed Time Slot (GTS) mechanism

The GTS mechanism allows devices to access the medium without contention, in the CFP. Figure A.13 in Annex A depicts the GTS request command frame format. GTSs are allocated by the Coordinator and are used only for communications between the Coordinator and a device. Each GTS may contain one or more time slots. The

Coordinator may allocate up to seven GTSs in the same superframe, provided that there is sufficient capacity in the superframe. Each GTS has only one direction: from the device to the Coordinator (transmit) or from the Coordinator to the device (receive). Figure 14 illustrates message sequence diagram for a GTS allocation.

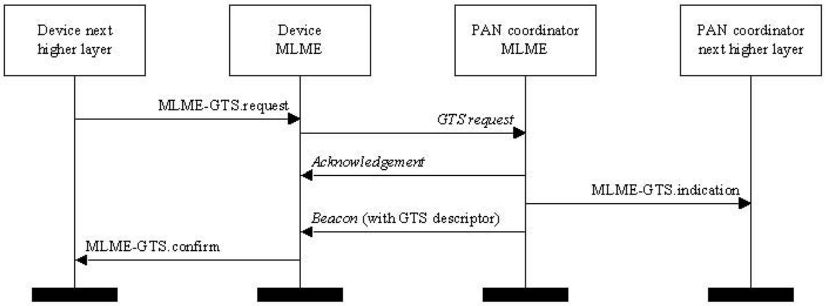


Figure 14 - GTS allocation message sequence diagram [4]

The GTS can be deallocated at any time at the discretion of the Coordinator or the device that originally requested the GTS allocation. A device to which a GTS has been allocated can also transmit during the CAP. The Coordinator is the responsible for performing the GTS management; for each GTS, it stores the starting slot, length, direction, and associated device address. All these parameters are embedded in the GTS request command. Only one transmit and/or one receive GTS are allowed for each device. Upon the reception of the deallocation request the Coordinator updates the GTS descriptor list by removing the previous allocated slot and rearranging the remaining allocation starting slots. The arrangement of the CFP consists in shifting right the allocated GTS descriptors with starting slot before the recent deallocated GTS descriptor and consequently the final CAP slot variable is updated. Figure 15 illustrates an example of this procedure.

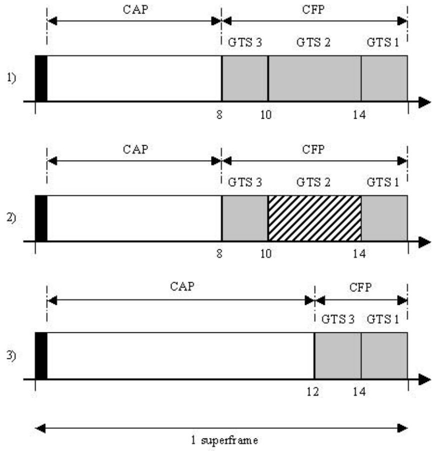


Figure 15 - CFP defragmentation upon a GTS deallocations [4]

In the Figure 15, the 1st timeline represents the three allocated GTS. The 2nd timeline shows the deallocation of GTS 2 that starts on the 10th time slot and has duration of 4 time slots. The final timeline show GTS 3 shifted right by 4 time slots. The first CTF time slot shifted right from slot 8 (in timeline 1) to slot 12 (in timeline 3).

The Coordinators monitor GTS activity and if there are no transmissions during a defined number of time slots the GTS allocation expires. The expiration occurs if no data or no acknowledgement frames are received by the device or by the Coordinator, on every *2*n* superframes, where *n* is defined as:

$$\begin{cases} n = 2^{(8-macBeaconOrder)}, if\ 0 \le macBeaconOrder \le 8 \\ n = 1, if\ 9 \le macBeaconOrder \le 14 \end{cases} \tag{2.10}$$

2.2.6 CSMA/CA Mechanism

In IEEE 802.15.4, contention-based MAC (Medium Access Control) can be either slotted or unslotted CSMA/CA, depending on the network operation behaviour: beacon-enabled or non beacon-enabled modes, respectively.

The CSMA/CA mechanism is based on backoff periods (with the duration of 20 symbols). Three variables are used to schedule medium access:

- *Number of Backoffs* (*NB*), representing the number of failed attempts to access the medium;
- *Contention Window* (*CW*), representing the number of backoff periods that must be clear before starting transmission;
- *Backoff Exponent* (*BE*), enabling the computation of the number of wait backoffs before attempting to access the medium again.

Figure 16 depicts a flowchart describing the slotted version of the CSMA/CA mechanism. It can be summarized in five steps:

1. initialization of the algorithm variables: *NB* equal to 0; *CW* equals to 2 and *BE* is set to the minimum value between 2 and a MAC sub-layer constant (*macMinBE*);
2. after locating a backoff boundary, the algorithm waits for a random defined number of backoff periods before attempting to access the medium;
3. Clear Channel Assessment (CCA) to verify if the medium is idle or not.
4. The CCA returned a busy channel, thus *NB* is incremented by 1 and the algorithm must start again in Step 2;
5. The CCA returned an idle channel, *CW* is decremented by 1 and when it reaches 0 the message is transmitted, otherwise the algorithm jumps to Step 3.

In the slotted CSMA/CA, when the battery life extension is set to 0, the CSMA/CA must ensure that, after the random backoff (step 2), the remaining operations can be undertaken and the frame can be transmitted before the end of the CAP. If the number of backoff periods is greater than the remaining in the CAP, the MAC sub-layer pause the backoff countdown at the end of the CAP and defers it to the start of the next superframe. If the number of backoff periods is less or equal than the remaining number of backoff periods in the CAP, the MAC sub-layer applies the backoff delay and re-evaluate whether it can proceed with the frame transmission. If the MAC sub-layer do not have enough time, it defers until the start of the next superframe, continuing with the two CCA evaluations (step 3). If the battery life extension set to 1, the backoff countdown must only occur during the first six full backoff periods, after the reception of the beacon, as the frame transmission must start in one of these backoff periods.

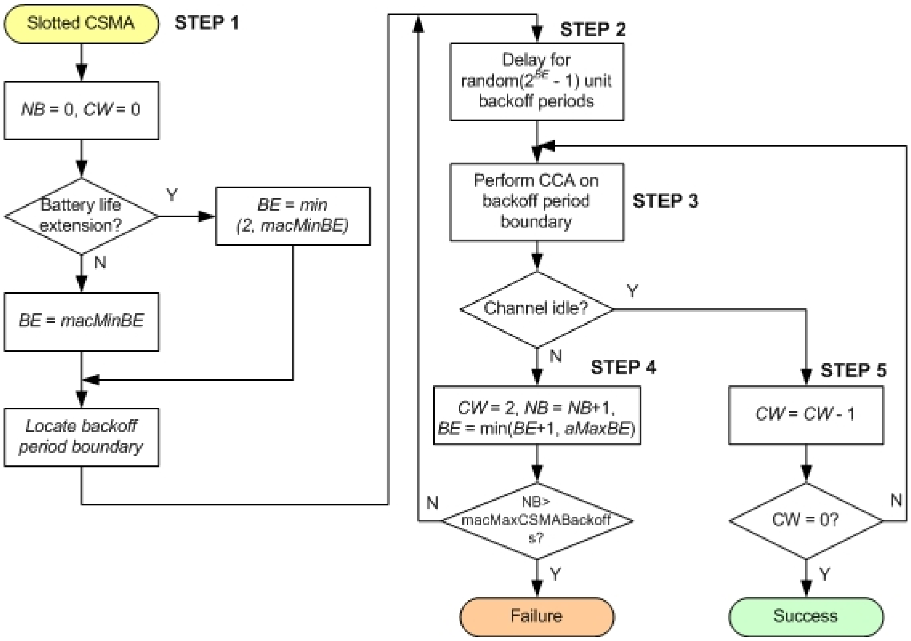


Figure 16 - The Slotted CSMA/CA Mechanism

The non slotted mode of the CSMA/CA (Figure 17) is very similar to the slotted version except the algorithm does not need to rerun (*CW* number of times) when the channel is idle.

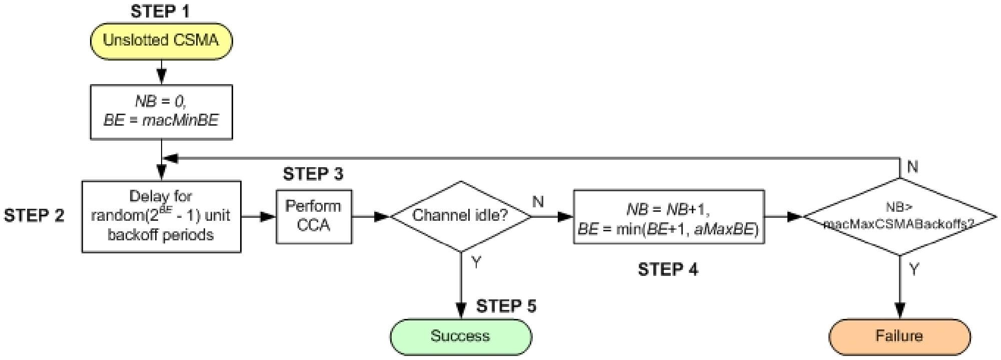


Figure 17 - The Un-slotted CSMA/CA mechanism

2.2.7 Inter-Frame Spacing (IFS)

The inter-frame spacing (IFS) is an idle communication period that is needed for supporting the MAC sub-layer needs to process data received by the physical layer. To allow this, all transmitted frames are followed by an IFS period. If the transmission requires an acknowledgment, the IFS will follow the acknowledgement frame. The length of the IFS period depends on the size of the transmitted frame: a long inter-frame spacing (LIFS) or short inter-frame spacing (SIFS). The selection of the IFS is based on the IEEE 802.15.4 *aMaxSIFSFrameSize* parameter, defining the maximum allowed frame size to use the SIFS. The CSMA/CA algorithm takes the IFS value into account for transmissions in the CAP. These concepts are illustrated in Figure 18.

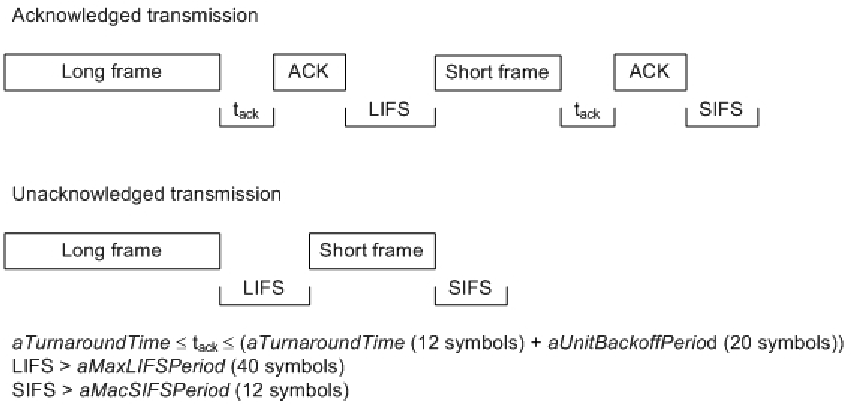

Acknowledged transmission
Long frame
ACK
Short frame
ACK
t_{ack}
LIFS
t_{ack}
SIFS
Unacknowledged transmission
Long frame
Short frame
LIFS
SIFS
aTurnaroundTime ≤ t_{ack} ≤ (*aTurnaroundTime* (12 symbols) + *aUnitBackoffPeriod* (20 symbols))
LIFS > *aMaxLIFSPeriod* (40 symbols)
SIFS > *aMacSIFSPeriod* (12 symbols)

Figure 18 - Inter-frame spacing

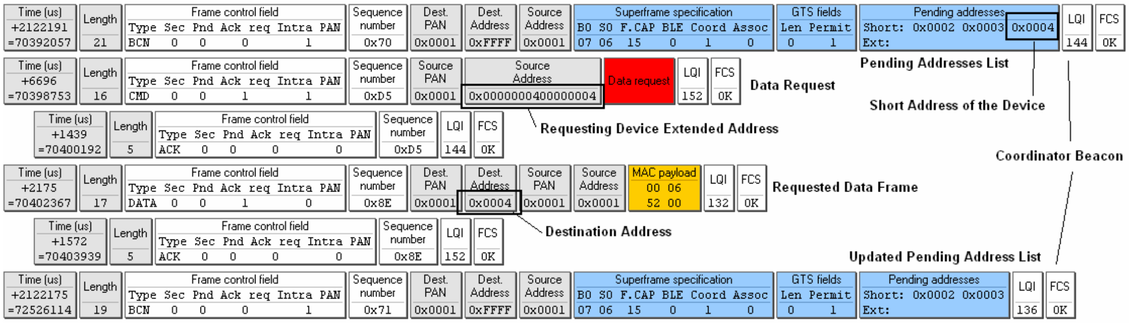
2.2.8 Transmission scenarios and reception conditions

The IEEE 802.15.4 protocol standard enables three different types of transmissions:

1. *Direct transmissions* – the frames are transmitted to the medium without any channel assessment i.e. the beacon frames, the acknowledgment frames and the frames in the GTS time slots;
2. *Indirect transmissions* – the frames are stored in the Coordinator to which the destination device is associated. Then, the information about the stored frames (or pending transmissions) is included in the pending addresses descriptors fields of the beacon frame. If a device has pending data in the Coordinator it can request it by sending a data request command frame. An example of this mechanism is depicted in Figure 19 where the Coordinator beacon contains the short address 0x0004 in the pending address list. In the Coordinator neighbour table, the short address 0x0004 is associated to the extended address 0x0000000400000004. Then, the device 0x0004 requests the data with a data request message embedding its extended address. The Coordinator searches in

its neighbour tables for the short address corresponding to the extended address received in the command frame and transmit the corresponding pending data. In the next Coordinator beacon the pending address list is updated.

3. *Normal transmissions* – the frames are transmitted to the medium with contention, by applying the CSMA/CA algorithm i.e. data frames and command frames transmitted during the CAP. Depending of the operation mode (beacon-enabled or non beacon-enabled) the CSMA/CA algorithm has two versions, the slotted or the unslotted respectively.



```
Time (us)  | Length | Frame control field               | Sequence | Dest.  | Dest.   | Source  | Superframe specification      | GTS fields | Pending addresses               | LQI | FCS
+2122191   |        | Type Sec Pnd Ack req Intra PAN    | number   | PAN    | Address | Address | BO SO F.CAP BLE Coord Assoc   | Len Permit | Short: 0x0002 0x0003 0x0004     |     |
=70392057  | 21     | BCN   0   0      0          1     | 0x70     | 0x0001 | 0xFFFF  | 0x0001  | 07 06  15    0    1      0    | 0     1    | Ext:                            | 144 | OK

Time (us)  | Length | Frame control field               | Sequence | Source | Source             | Data request | LQI | FCS
+6696      |        | Type Sec Pnd Ack req Intra PAN    | number   | PAN    | Address            |              |     |
=70398753  | 16     | CMD   0   0      1          1     | 0xD5     | 0x0001 | 0x0000000400000004 |              | 152 | OK

Time (us)  | Length | Frame control field               | Sequence | LQI | FCS
+1439      |        | Type Sec Pnd Ack req Intra PAN    | number   |     |
=70400192  | 5      | ACK   0   0      0          0     | 0xD5     | 144 | OK

Time (us)  | Length | Frame control field               | Sequence | Dest.  | Dest.   | Source | Source  | MAC payload | LQI | FCS
+2175      |        | Type Sec Pnd Ack req Intra PAN    | number   | PAN    | Address | PAN    | Address | 00 06       |     |
=70402367  | 17     | DATA  0   0      1          0     | 0x8E     | 0x0001 | 0x0004  | 0x0001 | 0x0001  | 52 00       | 132 | OK

Time (us)  | Length | Frame control field               | Sequence | LQI | FCS
+1572      |        | Type Sec Pnd Ack req Intra PAN    | number   |     |
=70403939  | 5      | ACK   0   0      0          0     | 0x8E     | 152 | OK

Time (us)  | Length | Frame control field               | Sequence | Dest.  | Dest.   | Source  | Superframe specification      | GTS fields | Pending addresses        | LQI | FCS
+2122175   |        | Type Sec Pnd Ack req Intra PAN    | number   | PAN    | Address | Address | BO SO F.CAP BLE Coord Assoc   | Len Permit | Short: 0x0002 0x0003     |     |
=72526114  | 19     | BCN   0   0      0          1     | 0x71     | 0x0001 | 0xFFFF  | 0x0001  | 07 06  15    0    1      0    | 0     1    | Ext:                     | 136 | OK
```

Figure 19 - Indirect transmission example.

The IEEE 802.15.4 protocol standard identifies three different transmissions scenarios during the CAP:

- *Successful data transmission*– the sender successfully transmits the frame to the intended recipient. The recipient receives the frame and sends an acknowledgment if required. If it is an acknowledged request, the sender starts a timer that expires after *macAckWaitDuration* symbols. Upon the reception of the acknowledge frame (before the timer expires), the sender disables and reset the timer. The data transfer is completed successfully.
- *Loss of frame* – the sender successfully transmits the frame to the medium but it never reaches the destination, so that an acknowledgement frame is not transmitted. The sender timer expires (after *macAckWaitDuration*) and the sender retransmits the frame again. This procedure is repeated up to a maximum of *aMaxFrameRetries* times after which the transmission aborts.
- *Loss of acknowledgment* - the sender successfully transmits the frame to the intended recipient that upon reception replies with an acknowledgement frame. The sender never receives the acknowledgment and retries the transmission.

The MAC sub-layer will only accept frames from the Phy layer if it satisfies the following requirements:

- The frame type subfield of the frame control field does not contain an illegal frame type;
- If the frame type indicates that the frame is a beacon frame, the source PAN identifier must match *macPANId,* unless *macPANId* is equal to *0xffff*, in which case the beacon frame must be accepted regardless of the source PAN identifier;

- If a destination PAN identifier is included in the frame, it must match *macPANId* or the broadcast PAN identifier (*0xffff*);
- If a short destination address is included in the frame, it must match either *macShortAddress* or the broadcast address (*0xffff*). Otherwise, if an extended destination address is included in the frame, it must match *aExtendedAddress*;
- If only source addressing fields are included in a data or MAC command frame, the frame is accepted only if the device is a Coordinator and the source PAN identifier matches *macPANId*.

Chapter 3

IEEE 802.15.4/ZigBee Protocol Stack Implementation

This chapter details the implementation of the IEEE 802.15.4/ZigBee protocol stack in TinyOS. This chapter also focuses on the hardware technology used and overviews the TinyOS operating system including the nesC programming language and the network analysers used for debug and analysis during the implementation. The implementation is available as open-source in Open-ZB [24]. This chapter concerns the software architecture and the implementation of the most relevant features of the protocols. This chapter will also address the implementation differences between the MICAz and TelosB mote platforms. This chapter ends with some concluding remarks on the implementation efforts.

3.1 Introduction

The Open-ZB [24] development efforts include the implementation of the IEEE 802.15.4 Data Link Layer and a part of the ZigBee Network Layer. This protocol stack implementation is transversal to all experiments described in this Thesis, namely on Chapters 4, 6 and 7. The future objectives of the Open-ZB are to implement the full IEEE 802.15.4 protocol stack and the full functionalities of the ZigBee Network Layer.

The first version of the IEEE 802.15.4 implementation only supported the MICAz motes [22] and it was conditioned to that hardware platform. The latest version also supports the TelosB [23] hardware platform. TelosB architecture is slightly different from the MICAz, especially due the 16 bits MSP430 microcontroller [35] compared with MICAz 8 bit Atmega128 microcontroller [36]. This triggers the need for a selection of the hardware files (or drivers) already provided in TinyOS and to an adaptation of the previous version of the implementation to support the 16 bits memory block of the MSP430.

The Open-ZB protocol stack implementation has three main blocks: (1) the development of the hardware abstraction layer, including the IEEE 802.15.4 physical layer and the timer module supporting both MICAz and TelosB mote platforms; (2) the IEEE 802.15.4 MAC sub-layer; and (3) the ZigBee Network Layer. The implemented features of the IEEE 802.15.4 include the slotted version of CSMA/CA algorithm, allowing the testing and parameterization of its variables, the different types of transmission scenarios (e.g. direct, indirect and GTS transmissions), association of the devices, channel scans (e.g. energy detection and passive scan), beacon management and other mechanisms. Other IEEE 802.15.4 features were left out of this implementation version because they are not needed for the experimentations described in Chapters 4, 6, and 7. Features that are not currently supported include the unslotted version of the

CSMA/CA, the active and orphan channel scan, the use of extended addressing fields in normal data transmissions.

In the ZigBee Network Layer, the currently supported features comprise the data transfer between the Network Layer and the MAC sub-layer, the association mechanisms and the network topology management (e.g. cluster-tree support by the ZigBee Addressing schemes) and routing (e.g. neighbour routing and tree-routing). Security is not supported yet.

3.2 Technological Context and Development Tools

3.2.1 TinyOS and nesC

TinyOS [21] is an operating system for embedded systems with an event-driven execution model. TinyOS is developed in nesC [20], a language for programming structured component-based applications. nesC has a C-like syntax and is designed to express the structuring concepts of TinyOS. This includes the concurrency model, mechanisms for structuring, naming and linking together software components into embedded system applications. The component-based application structure provides flexibility to the application design and development. nesC applications are built out of components and interfaces.

The components define two target areas:

- *the specification*, a code block that declares the functions it provides (implements) and the functions that it uses (calls);
- *the implementation* of the functions provided.

The interfaces are bidirectional collections of functions provided or used by a component. The interfaces *commands* are implemented by the providing component and the interface *events* are implemented by the component using it. The components are "wired" together by means of interfaces, forming an application.

TinyOS defines a concurrency model based on tasks and hardware events handlers/interrupts. TinyOS tasks are synchronous functions that run without preemption until completion and their execution is postponed until they can execute. Hardware events are asynchronous events that are executed in response to a hardware interrupt and also run to completion.

TinyOS directory structure is the following:

- tinyos-1.x
    - apps – Standard TinyOS application and test programs;
    - contrib – Users contribution (generally the tinyos-1.x directory structure is replicated in each contribution);
    - doc – Documentation and On-line Tutorial;
    - tools – Development utilities and programs;
    - tos – TinyOS modules and interfaces.
- tos
    - interfaces – Interfaces for TinyOS component;
    - lib – Libraries;

- platform – Drivers for mote hardware;
- sensorboards – Drivers for sensor boards;
- system – Drivers for the mote system – EEPROM,UART;
- types – Special type definition.

Figure 20 depicts the possible interactions between the components and interfaces.

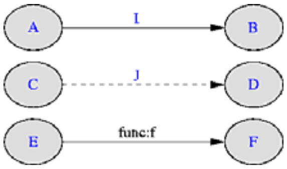


Figure 20 - Graphic arrangement of the components and their wiring

The graphical arrangements have the following meaning:

- **A** requires interface **I**, **B** provides **I**, and **A** and **B** are wired together.
- **C** and **D** both require or both provide **J**. The direction of the arrow indicates that the original wiring is "**C** = **D**".
- **E** requires function **f**, and **F** provides function **f**.

TinyOS also provides a program called *nesdoc* that provides a graphical arrangement of all the components used by an application. This tool is very useful to understand how TinyOS binds all the components.

3.2.2 MOTES – MICAz and TelosB

The IEEE 802.15.4/ZigBee implementation is supported by two hardware platforms, the MICAz [22] and the TelosB [23] motes. The MICAz mote (Figure 21 left) has the following features:

- ATMEL ATmega128L 8-bit microcontroller [36];
- CC2420 RF transceiver [37];
- 128 KB of Program memory (in-system reprogrammable flash);
- 4 KB of EEPROM;
- Supports several sensor boards;
- UART communication port.

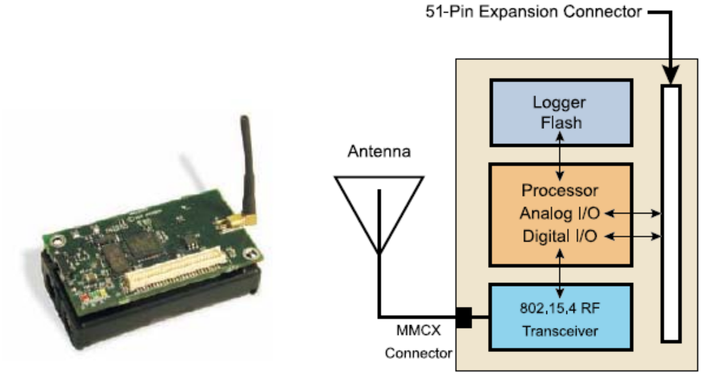


Figure 21 - Crossbow Micaz mote and the block diagram

The TelosB mote (Figure 22 left) has the following characteristics:

- TI MSP430 16-bit microcontroller [35];
- CC2420 RF transceiver [37];
- 48 KB of Program memory (in-system reprogrammable flash);
- 10 KB of EEPROM;
- Includes a temperature and light sensor;
- UART communication port (USB converter).

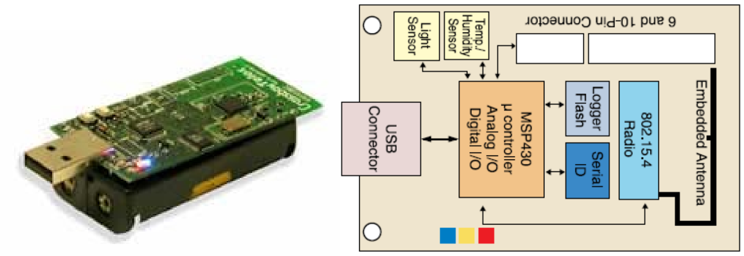
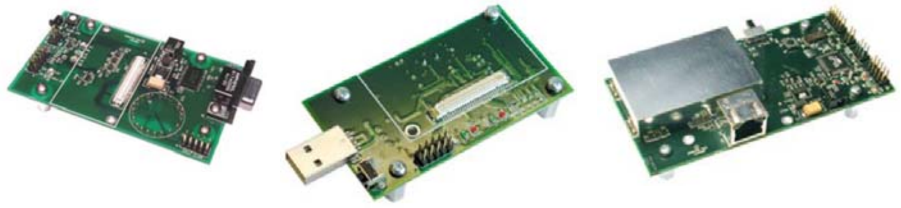


Figure 22 - Crossbow TelosB mote and the block diagram

3.2.3 Interface Boards

The TelosB motes do not need any programmer interface because they already have an USB port that can be used to upload programs as well as interfacing the mote with other equipments.

The MICAz mote needs to be programmed using an interface board such as the MIB510 (Figure 23.A) [38], the MIB520 (Figure 23.B) [39], and the MIB600 (Figure 23.C) [40]. The interface boards MIB510 and MIB520 are very similar except the fact the MIB510 has a serial RS-232 interface and the MIB520 has an USB interface. The MIB600 has an RJ-45 Ethernet interface with an implementation of the full TCP/IP protocol. These three interface boards allow the use of a JTAG adapter for debugging and can be used as base stations interfacing the wireless sensor network with a PC.

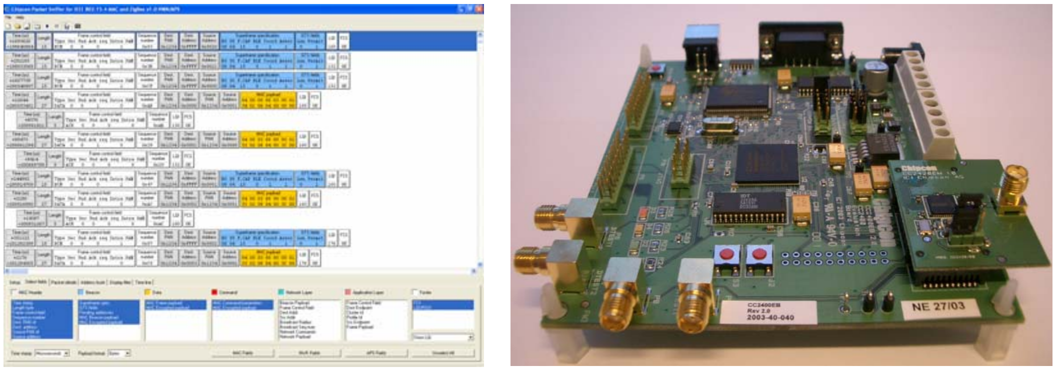


A) MIB510 **B) MIB520** **C) MIB600**

Figure 23 - Interface Boards - MIB510, MIB520 and MIB600

3.2.4 IEEE 802.15.4/ZigBee Protocol Analysers

The implementation of the IEEE 802.15.4/ZigBee has been supported by two network protocol analysers (packet sniffers): the Chipcon CC2420 Packet Sniffer for IEEE 802.15.4 v1.0 [41] and the Daintree IEEE 802.15.4/ZigBee Network Analyser [42]. These analysers interpret the IEEE 802.15.4 and ZigBee frames, allowing to debug and to validate the implementation of the IEEE 802.15.4/ZigBee protocols.



a) Snapshot of the sniffer application **b) CC2420 EB with a CC2420EM**

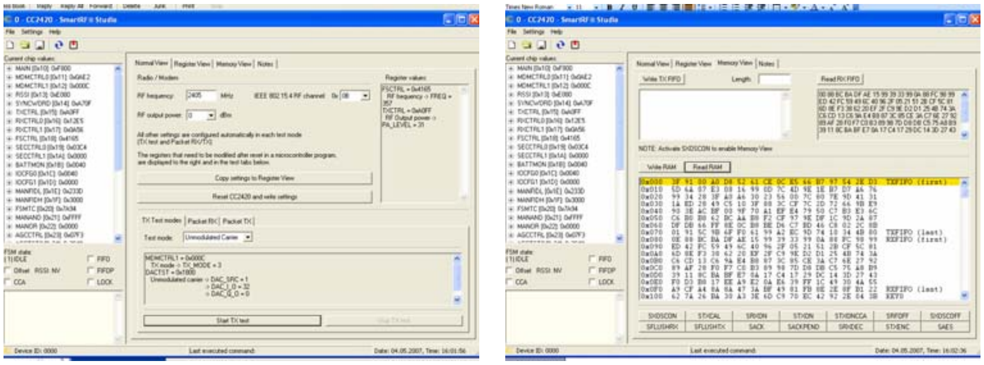
Figure 24 - Overview of the Chipcon IEEE802.15.4/ZigBee Packet Sniffer

The packet sniffer provided by Chipcon (Figure 24), the CC2420 Packet Sniffer for IEEE 802.15.4 v1.0 provides a raw list of the packets transmitted. This application works in conjunction with a CC2400EB board (Figure 24.b) and a CC2420EM module (equipped with a CC2420 radio transceiver). Figure 24.a depicts a snapshot of the sniffer application which provides the following features:

- Raw list of the received packets with timestamp information;
- Interpretation of the packets information, highlighting the different packet fields;
- Packet fields filtering;
- Device list.

Chipcon also provides a tool used to test the transceivers, the SmartRF Studio [43]. This application interacts with the CC2420EB/CC2420EM evaluation board and allows viewing and interacting with the CC2420 transceiver memory registries. With this tool is possible to test different configurations on the transceiver and test its behaviour with simple send/receive functions. This tool was very useful during the protocol stack implementation enabling a better understanding of the physical layer implementation and the functionalities of the transceiver. Figure 25 depicts an overview example of the Smart RF application interfaces, which provides the following features:

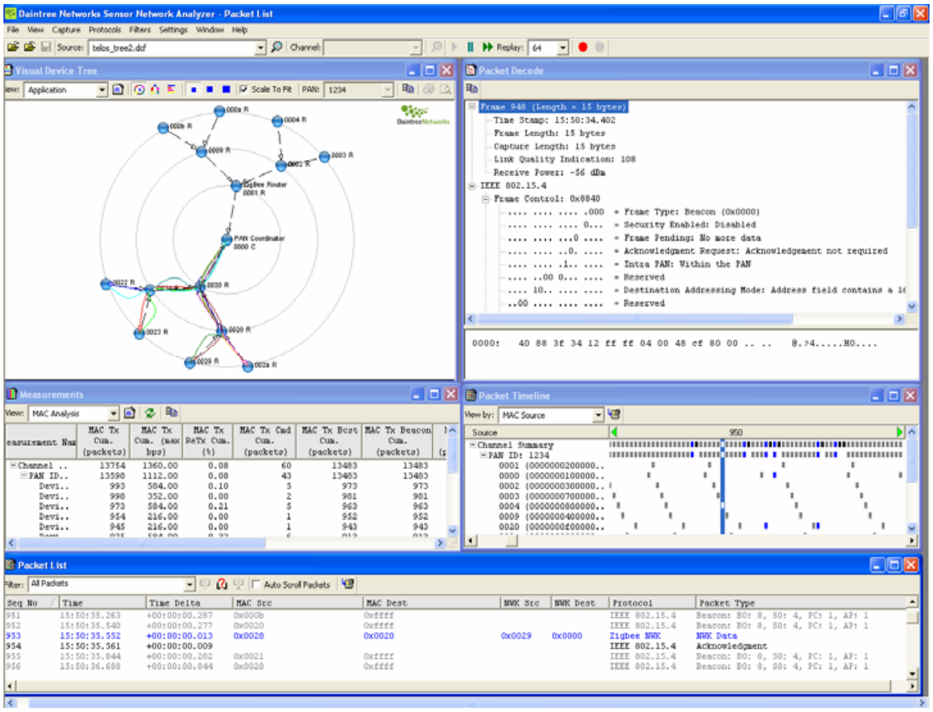
- Read/Write from/to the CC2420 transceiver memory registries (Figure 25.a);
- Execute functions of the transceiver (e.g. TR ON, TX OFF, etc.)
- Test transmissions, IEEE 802.15.4 compatible packets or an unmodulated carrier;
- Memory views (Figure 25.b) of the buffers (receive and transmit).



a) Registry view **b) Memory view**

Figure 25 - Overview Chipcon SmartRF Studio

The Daintree Network Analyser provides more functionalities than the Chipcon sniffer. Besides the received packets list and their field highlighting, it also constructs a graphic view of the network topology, including the visualization of routing paths, message flows, device states and link quality of the messages, as depicted in Figure 26.



```
Frame 940 (Length = 15 bytes)
  Time Stamp: 15:50:34.402
  Frame Length: 15 bytes
  Capture Length: 15 bytes
  Link Quality Indication: 108
  Receive Power: -56 dBm
IEEE 802.15.4
  Frame Control: 0x8840
    .... .... .... .000  = Frame Type: Beacon (0x0000)
    .... .... .... 0...  = Security Enabled: Disabled
    .... .... ...0 ....  = Frame Pending: No more data
    .... .... ..0. ....  = Acknowledgment Request: Acknowledgement not required
    .... .... .1.. ....  = Intra PAN: Within the PAN
    .... ..00 0... ....  = Reserved
    .... 10.. .... ....  = Destination Addressing Mode: Address field contains a 16
    ..00 .... .... ....  = Reserved

0000:   40 88 3f 34 12 ff ff 04 00 48 ef 80 00 .. ..   @.?4.....HO....
```

easurement Nam	MAC Tx Cum. (packets)	MAC Tx Cum. (max bps)	MAC Tx ReTx Cum. (%)	MAC Tx Cmd Cum. (packets)	MAC Tx Bcst Cum. (packets)	MAC Tx Beacon Cum. (packets)
Channel ..	13754	1360.00	0.08	60	13483	13483
PAN ID..	13598	1112.00	0.08	43	13483	13483
Devi..	993	584.00	0.10	5	973	973
Devi..	998	352.00	0.00	2	981	981
Devi..	973	584.00	0.21	5	963	963
Devi..	954	216.00	0.00	1	952	952
Devi..	945	216.00	0.00	1	943	943

```
Channel Summary
  PAN ID: 1234
    0001 (0000000200000..
    0000 (0000000100000..
    0002 (0000000300000..
    0003 (0000000700000..
    0004 (0000000000000..
    0009 (0000000400000..
    0020 (0000000f00000..
```

Seq No	Time	Time Delta	MAC Src	MAC Dest	NWK Src	NWK Dest	Protocol	Packet Type
951	15:50:35.263	+00:00:00.287	0x000b	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
952	15:50:35.540	+00:00:00.277	0x0020	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
953	15:50:35.552	+00:00:00.013	0x0028	0x0020	0x0029	0x0000	Zigbee NWK	NWK Data
954	15:50:35.561	+00:00:00.009					IEEE 802.15.4	Acknowledgment
955	15:50:35.844	+00:00:00.282	0x0021	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
956	15:50:36.688	+00:00:00.844	0x0028	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1

Figure 26 - Overview of Daintree Network Analyser

Another interesting feature, is the network status of the devices by analysing the messages transmitted, messages received, loss message ration, bandwidth usage, average link quality indicator among others. This application also distinguishes the analysis parameters depending on the selected protocol layers. The Daintree Analyser enables the import of a plant layout (office floor, factory floor) and overlay the network topological view over it. This feature allows dragging and dropping nodes, assigning labels to each node and it can be very useful for monitoring the network.

The hardware used in conjunction with this network analyser is the 2400 Sensor Network Adapter [44]. This adapter includes an Ethernet interface and can be used for a multiple and synchronized node sniffing, meaning that several 2400 can be scattered (connected to an Ethernet network) in a certain geographical area in a way that IEEE 802.15.4/ZigBee traffic can be collected at different locations of a large-scale network into a single application.

3.2.5 Related implementations and hardware

There are several implementations of the IEEE 802.15.4/ZigBee protocols supported by different hardware platforms [45-55]. These were developed in C language and programmed directly in the microcontroller without any supporting operating system (like TinyOS). Also, in some implementations, the source code is not open, enabling just the implementation of top level applications using a pre-defined interface set. In addition, these implementations can only be used in the provided hardware platform. Additionally these implementations only support the non-beacon enabled mode, therefore allowing the construction of ZigBee standard mesh networks (refer to Section 2.1.1), but not of beacon-enabled Star and Cluster-Tree networks.

Ember [45] EmberZNet, compliant with the 2006 ZigBee specification. This solution works with the EM250 System on Chip and EM260 ZigBee co-processor [46,47]. Freescale Semiconductor [48] also provides a commercial implementation compliant with the 2006 ZigBee specification, the BeeStack. The software stack supports several Freescale chip platforms, such as the MC13192 [49] and the MC13201 [50].

The IA USB Dongle [51], developed by Integration Associates [52] provides an USB hardware with device drivers that implement a 2006 ZigBee compliant stack. The provided drivers allow the integration of the dongle with different operating systems. The source code is not provided.

Texas Instruments developed the Z-Stack [53] that is compliant with the ZigBee 2006 specification and supports multiple platforms including the CC2431 System-on-Chip [54], the CC2420 [37] and MSP430 platforms [35]. The Z-Stack is a free implementation developed in C language. The ATMEL AVR Z-Link [55] is another IEEE 802.15.4 compliant platform that includes a free stack implementation in C with available source code.

Besides the above mentioned companies there are several others with ZigBee solutions. Nevertheless, only the mesh network topologies are supported and the software implementations are limited. Most of these companies are semiconductor companies dedicated to hardware development.

Refer to [56] for a full list of ZigBee compliant platforms.

3.3 Software Architecture

The Open-ZB implementation has three main TinyOS components: the *Phy*, the *Mac* and the *NWL* (Figure 27). The *Mac* and the *NWL* are shared by the two platforms (MICAz and the TelosB) and there are two different Phy components, one for each platform. At compilation time, the *Phy* component is selected according to the envisaged platform. The need of two different *Phy* components is due to the fact that the TinyOS hardware specific modules are different for each platform. Also, the two platform differ in the hardware timers they provide, leading to two different timer modules (the *TimerAsync*) with the purpose of maintaining all asynchronous timer events of the Mac layer (e.g. beacon interval, superframe duration, time slots and backoff periods). Nevertheless, the software architecture is the same for both platforms.

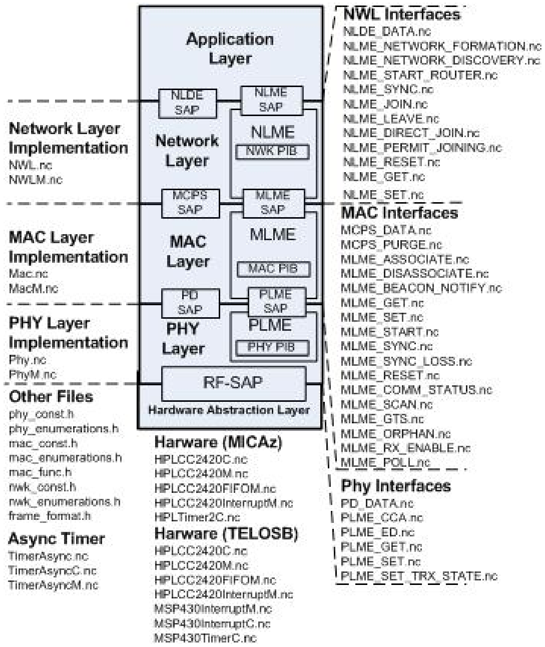


Figure 27 - Protocol stack software architecture

Table 3 summarizes the implemented functionalities in each module of the protocol stack.

Table 3 - Functionalities of the implemented protocol stack components

Component	Functionalities
Phy	Activation and deactivation of the radio transceiver; Energy detection within the current channel; Transceiver data management, Received Signal Strength Indication (RSSI) readings and channel frequency selection; Clear Channel Assessment (CCA) procedure for the CSMA/CA mechanism; Data transmission and reception management.
Mac	Beacon generation if the device is a Coordinator; Synchronization services; PAN association and disassociation procedures; CSMA/CA as a contention access mechanism; GTS management mechanism.
NWL	Definition of the network topology (by enabling the device operation as a ZC, ZR or ZED); Association mechanisms; ZigBee addressing schemes; Maintenance of neighbour tables; Tree-Touting.

Figure 27 and Figure 28 present the layered view of the different TinyOS components and interfaces of the IEEE 802.15.4/ZigBee protocol stack implementation. The organization in modules enables the easy and fast development of adaptations/extensions to the current implementation. Each of these modules makes use of auxiliary files to implement some generic functions (e.g. functions for bit aggregation into variable blocks), constants declaration (e.g. layer constants), enumerations (e.g. data types, frame types, response status) and data structure definitions (e.g. frame construction data structures).

The interface files (Figure 27 right side) are used to bind the components and represent one Service Access Point (SAP). Each of these interfaces provide functions that are called from the higher layer module and are executed/implemented in the lower layer module. The interfaces also provide functions used by the lower layer modules to signal functions that are executed/implemented in the higher layer modules. For example the *PD_DATA.nc* interface is used by the *MacM* module to transfer data to the *PhyM* module, that is going to be transmitted, and also enables the signalling by the *PhyM* in the *MacM* of received data.

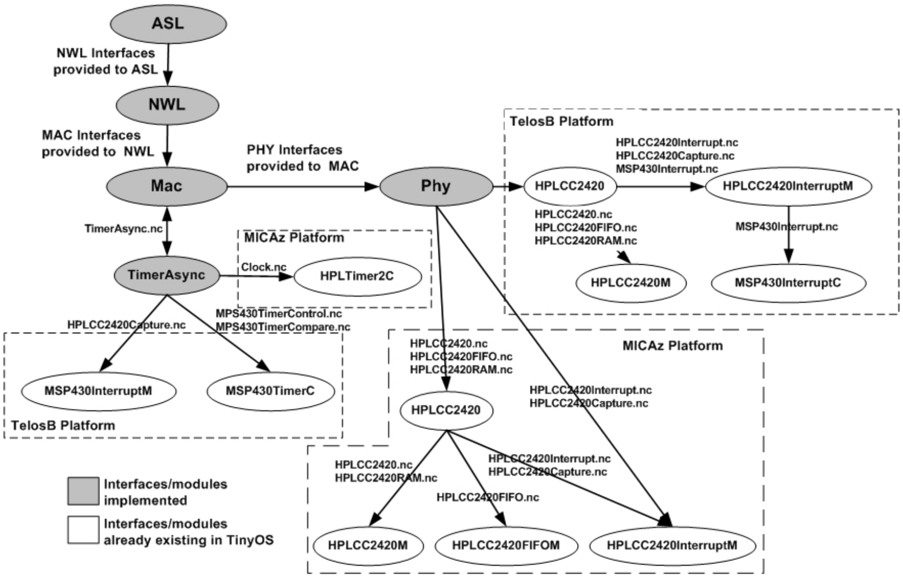


Figure 28 - TinyOS implementation diagram

Figure 28 depicts the relations between different components of the IEEE 802.15.4/ZigBee protocol stack implementation. Note that some components used in our IEEE 802.15.4/ZigBee protocol stack implementation are already part of the TinyOS operating system, namely the hardware components (e.g. the HPL<…>.nc and the MSP430<…>.nc modules).

In this implementation, there is no direct interaction with the hardware. In fact, TinyOS already provides hardware drivers forging a hardware abstraction layer used by the *Phy* component. In Figure 28, observe that the components filled in white are hardware components already provided by the TinyOS operating system.

Refer to an extended implementation technical report in [18] for a detailed description of the implementation functions, variables and protocol mechanisms.

3.4 IEEE 802.15.4 Implementation

3.4.1 Time synchronization and timers

An important aspect of the IEEE 802.15.4 protocol is the time synchronization. A first difficulty in the implementation of the IEEE 802.14.5 beacon-enabled mode was related to the TinyOS management of hardware timers provided by both MICAz and TelosB motes, which does not allow having the exact values in millisecond of the beacon interval, superframe, time slots and backoffs durations, as specified by the IEEE 802.15.4 standard. Also due to the difference between the hardware timers used by the two platforms is not possible to achieve the same timer granularities. To accomplish a precise synchronization, a timer component was developed. This has an asynchronous behaviour regarding the code execution, based on the hardware clock. This asynchronous timer (the *TimerAsync* component) has two different implementations, one for the MICAz mote using the *HPLTimer2C* TinyOS component and another for the TelosB mote using the *MSP430TimerC* TinyOS component.

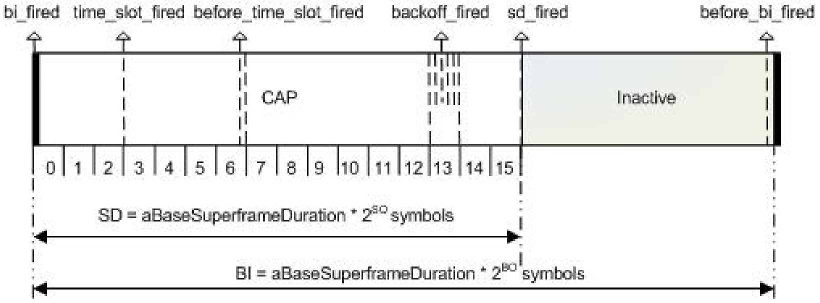


Figure 29 - Timer events in superframe structure

Figure 29 depicts the asynchronous events handled by the *TimerAsync* component. These events are used to maintain the operational behaviour of the IEEE 802.15.4 protocol in the beacon-enabled mode. As seen in Figure 29 there are several events fired: the beacon interval event (*bi_fired* event) that defines the interval between beacons depending on the beacon order (*BO*) value; the time slot event (*time_slot_fired* event), firing at the beginning of every time slot in the CAP; the event that fires before each beacon interval (*before_bi_fired*), used to switch the transceiver to receive or transmit mode before the reception/transmission of the beacon; the event that fires before each time slot (*before_time_slot_fired* event) and before each GTS allocation, thus dealing with the time needed for the transceiver to tackle this operations; the event that fires at the beginning of every backoff period (*backoff_fired* event), used in the implementation of the slotted CSMA/CA algorithm; and the event that fires at the end of the CAP (*sd_fired* event), defining the end of the active period.

Table C.1 and C.2 in Annex C present the duration in symbols, microseconds, backoff periods and number of clock tick of the timeslots and beacon intervals theoretical values as defined in the IEEE 802.15.4 protocol standard. Also in Annex C, the effective timer durations for the MICAz and TelosB motes are presented in Table C.3/C.4 and Table C.5/C6 respectively. The values are obtained taking into consideration an ideal clock tick granularity of the timer used to maintain them. The effective clock tick granularity obtained in the available hardware timers of the used platforms (MICAz and TelosB) is slightly different from the theoretical values. In this implementation, the clock ticks granularity used had to be approximated to the values that best fit the protocol requirements.

Because of the different hardware timers used in each platform two *TimerAsync* components were implemented.

MICAz implementation of the *TimerAsync* component

The MICAz mote hardware timer has a frequency of 7.3728 MHz. Figure 30 depicts the TinyOS components used by the *TimerAsync* component. The MICAz hardware clock is implemented in the *HPLTimer2C* component, already provided in TinyOS, and is defined by two parameters: the *SCALE* that defines the scale division of the timer frequency and the *INTERVAL* defining the number of ticks per clock firing.

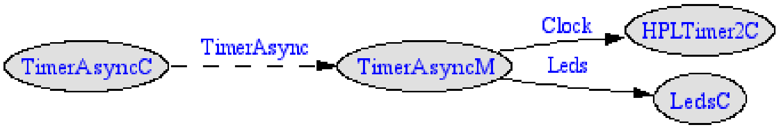


Figure 30 - MICAz mote *TimerAsyncC* component graph

The clock tick granularity of the MICAz mote that best fits the implementation requirements is equal to 69.44 microseconds, which approximately corresponds to four symbols (configuration with *SCALE* equal to 4 and *INTERVAL* equal to 1), assuming a 250 kbps bit rate. The 69.44 µs is achieved through dividing the clock frequency by 256 (*SCALE* of 4) resulting in frequency of 28.8 kHz, which approximately corresponds to 34,72 µs and 2 interval counts (*INTERVAL* of 1) resulting in a clock tick every 69.44 µs. This value corresponds to the duration of four symbols (16 bits) and is a fair approximation to the theoretical value of 64 µs.

Table 4 - MICAz clock ticks granularity comparison

MICAZ		
	Effective	Theoretical
Backoff Symbols	20	20
Symbol Duration (µs)	17,362	16
Backoff Duration (µs)	347,24	320
Granularity (µs)	69,44	64
Backoff Clock Ticks	5	5

In fact, the four symbols duration have a theoretical value of 64 microseconds which leads to a cumulative effect on the discrepancy with theoretically values of beacon intervals, superframe durations and time slot durations, especially for high superframe and beacon orders. For instance, the theoretical superframe duration with *SO=3* is equal to 122.88 ms, while it is equal to 133.36 ms using the MICAz motes and the TinyOS time management of the clock granularity. Table 4 compares the effective with the theoretical values used.

TelosB implementation of the *TimerAsync* component

The hardware timer available in the TelosB is based on a 32768 Hz clock that fires approximately every 30.5 microseconds. Comparing with the MICAz timer, this timer does not allow the set of a scale or interval parameters. Instead, this is a continuous timer that counts from 0 to 0xFFFF and when it overflows it triggers an interrupt (*AlarmCompare.fired)* and starts again from 0. The only parameterization allowed is the number of overflows counts before the issuing of the interrupt. Figure 31 depicts the TinyOS components used in the implementation of the *TimerAsync* component. The TelosB hardware clock is implemented in the *MSP430TimerC* module, already provided in TinyOS. The *HPLCC2420InterruptM* module implements the interrupt of the timer fired as well as all the other hardware interrupts.

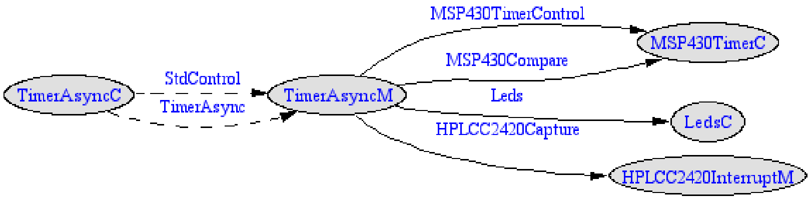


Figure 31 - TelosB mote *TimerAsyncC* component graph

The requirement that best fit this implementation is to trigger the timer on every backoff. The IEEE 802.15.4 defines that one backoff is 20 symbols, that theoretically correspond to 16 microseconds. With this timer granularity, the value obtained for each symbol is approximately 16.775 microseconds, leading to a backoff period duration of 335.5 microseconds instead of the 320 microseconds defined in the IEEE 802.15.4 protocol standard. A summary of the clock ticks granularity used in TelosB is presented in Table 5.

Table 5 - TelosB clock ticks granularity comparison

	TelosB	
	Effective	Theoretical
Backoff Symbols	20	20
Symbol Duration (µs)	16,775	16
Backoff Duration (µs)	335,5	320
Granularity (µs)	30,5	64
Backoff Clock Ticks	11	5

3.4.2 Frames construction

Frame construction is based on several structures and their option flags (refer to Annex A and B for the IEEE 802.15.4/ZigBee frame formats). All frames are based on a basic structure, the MPDU (MAC Protocol Data Unit), which contains the length of the frame, the frame control fields, a sequence number and an array that completes the maximum allowed packet size (127 bytes). Initially, when constructing a frame, a MPDU variable is created followed by the assignment of the packet length and the frame control field. Next, depending of the options selected (e.g. frame type), new structures are used to build the next fields. For a practical intuition of this technique consider the example of building a data frame with the following options in the address fields: short destination address and a long source address. The code outlined in Figure 32 and Figure 33 exemplifies the relevant structure definitions and the steps needed toward the construction of the frame.

```
#define DEST_SHORT_LEN 4
#define SOURCE_LONG_LEN 10

typedef struct dest_short
{
        uint16_t destination_PAN_identifier;
        uint16_t destination_address;
}dest_short;

typedef struct source_long
{
        uint16_t source_PAN_identifier;
        uint32_t source_address0;
        uint32_t source_address1;
}source_long;

typedef struct MPDU
{
        uint8_t length;
        uint8_t frame_contro1l;
        uint8_t frame_control2;
        uint8_t seq_num;
        uint8_t data[121];
}MPDU;
```

Figure 32 - Frame structure definition

```
MPDU frame_pkt;
MPDU * frame_pkt_ptr;
dest_long *dest_long_ptr;
source_short *source_short_ptr;

frame_pkt_ptr = (MPDU *) &frame_pkt;
dest_short_ptr = (dest_short *) &frame_pkt->data[0];

source_long_ptr = (source_long *) &frame_pkt->data[DEST_SHORT_LEN];
frame_pkt->length = data_len + msduLength + MPDU_HEADER_LEN;
frame_pkt->frame_contro1l = /* frame type( in this case TYPE_DATA) +
security flag + frame pending flag + acknowledge request flag */
frame_pkt->frame_contro12 = /* destination address mode (in this case
SHORT_ADDRESS) + source address mode (in this case LONG_ADDRESS)*/
frame_pkt->seq_num =/*Data Sequence Number*/;

dest_short_ptr->destination_PAN_identifier= /*Destination PAN Address*/;
dest_short_ptr->destination_address=/*Destination Address*/;

source_long_ptr->source_PAN_identifier=/*Source PAN Address*/;
source_long_ptr->source_address0=/*Source Address*/;
source_long_ptr->source_address1=/*Source Address*/;
```

Figure 33 - Data frame construction

3.4.3 Buffer management

The IEEE 802.15.4 protocol does not define how to implement the buffer mechanisms. The buffers implementation plays an important role in the correct behaviour of the protocol. On one hand, the implementation must avoid excessive memory copy operations, because it can cause synchronization problems due to the overload of the operating system memory stack and is very time consuming. On the other hand, the buffers have to be small and very well managed because of the devices memory constraints. For example, the MICAz motes only have approximately 4 Kbytes of RAM and the maximum packet length is about 127 bytes. If we increase the buffer size, the available memory of the mote will decrease rapidly. In case of the TelosB the available RAM is 10 Kbytes, allowing larger buffers.

This implementation uses 4 buffers:

- *buffer_msg* – Used to store the received messages;
- *send_buffer* – Used to store the messages that are ready to be send;
- *indirect_trans_queue* – Used by the Coordinator to store the messages that are send using the indirect transmission procedure. The messages stored need to be requested by the destination device in order to be sent. The Coordinator sends one of these messages by transferring it to the *send_buffer* queue.
- *gts_send_buffer* – Used to store the messages that are ready to be sent in one GTS during the CFP.

Sending and Receiving

The buffers used for receiving and sending are FIFO (First In First Out) buffers. The implementation consists on an array with a constant length, the buffer size, and two pointers. The first pointer (*in*) points to the next available slot to store a new message, and the second (*out*) points to the oldest message in the queue. There is also one variable that contains the current message count in the buffer; if its equal to the buffer size it means that the buffer is full.

Figure 34 depicts the implementation of these buffers, where A, B and C represent message frames.

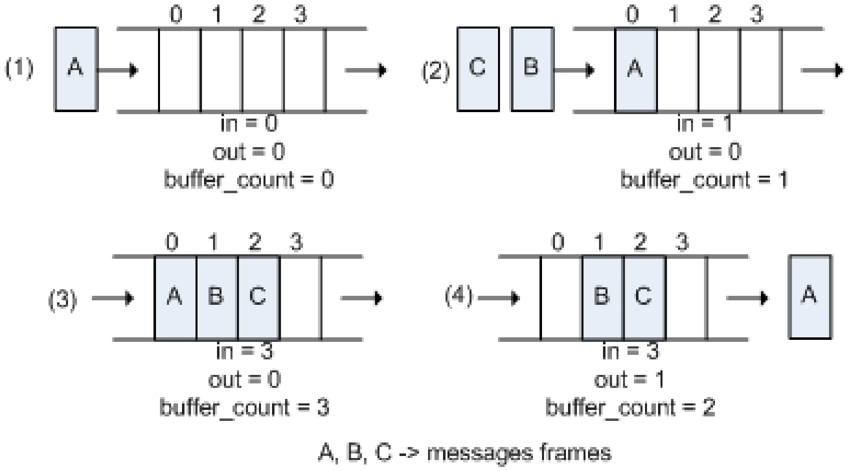


Figure 34 - Buffer management example

There are two ways for sending a message: using the CSMA/CA algorithm or without any channel assessment. The second way is only used to send beacons, GTS messages and acknowledgment frames. The CSMA/CA is used to send command and data frames. If the sent frame requires an acknowledgment, the sender must wait for it before sending a new message. The wait or the retransmission mechanism consists in a timer that is activated after a transmission that requires an acknowledgment.

Indirect Transmissions

The buffer used for the indirect transmissions is defined as a structure. When the Coordinator needs to send an indirect transmission, first it needs to search in the buffer for correct message. This procedure goes through all the positions of the message array comparing the destinations addresses until it finds the correct message, or ignores the indirect transmission request if there are no messages for the requested address. Each element in this buffer has a persistent time associated, after which the element is deleted.

GTS Buffer

The GTS buffer is used in two different ways. If the device is not a Coordinator the buffer is FIFO and it is used like the send and receive buffer with two pointers indicating the *in* and *out* of the messages and the total number of messages in the buffer. The messages are sent in the appropriate GTS allocated transmit time slot. If the device is a Coordinator, the buffer is maintained by an auxiliary structure with index pointers pointing to the appropriate message in the buffer, as depicted in Figure 35.

Also the auxiliary structure *gts_slot_list* is indexed with the available time slots that can be used for GTS transmission. This mechanism is used to avoid performing sequential and time consuming searches in the buffer to find the desired packet. Along with the *gts_send_buffer* buffer there is also one auxiliary array declared as *available_gts_index[GTS_SEND_BUFFER_SIZE],* storing the available indexes in the GTS buffer. The *GTS_SEND_BUFFER_SIZE* parameter defines the GTS maximum size. If the Coordinator wants to send data in the GTS, it must check if there are available indexes to store the message. When the message is sent, its *gts_send_buffer* position becomes available by inserting the *gts_send_buffer* index in the *available_gts_index* list.

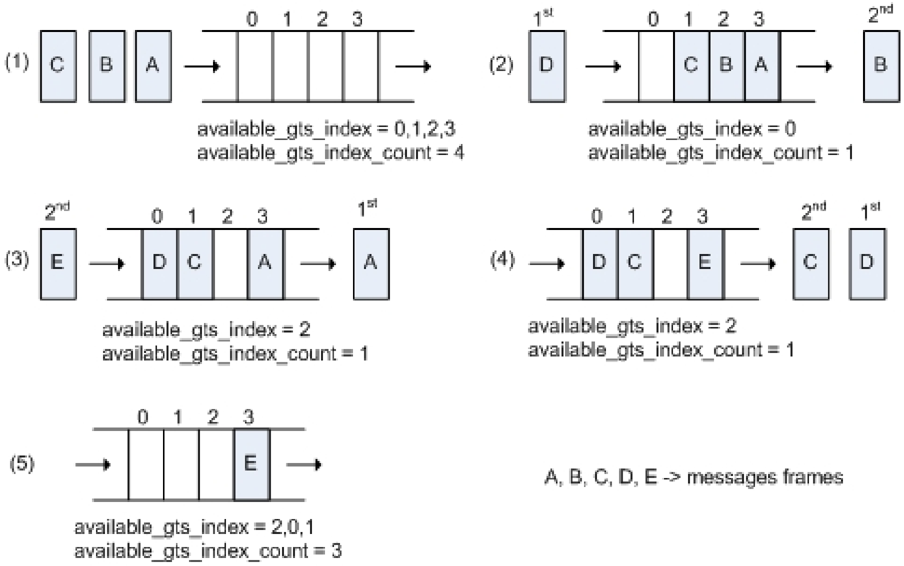

(1)
C
B
A
0 1 2 3
available_gts_index = 0,1,2,3
available_gts_index_count = 4
(2)
1st
D
0 1 2 3
C B A
2nd
B
available_gts_index = 0
available_gts_index_count = 1
(3)
2nd
E
0 1 2 3
D C A
1st
A
available_gts_index = 2
available_gts_index_count = 1
(4)
0 1 2 3
D C E
2nd
C
1st
D
available_gts_index = 2
available_gts_index_count = 1
(5)
0 1 2 3
E
available_gts_index = 2,0,1
available_gts_index_count = 3
A, B, C, D, E -> messages frames

Figure 35 - GTS buffer management

Each element in the *gts_slot_list* array represents one GTS time slot, up to the maximum of seven, defined in the protocol as the maximum number of GTS time slots available for GTS allocation. The *gts_slot_element* defines a FIFO buffer used to store indexes that reference positions in the *gts_send_buffer,* and it is maintained as the send and receive buffers.

3.4.4 Beacon management

The beacon frame contains all the information about the PAN (refer to Figure A.2 in Annex A.2 for the beacon frame format). The frames are broadcast by the Coordinators and embed important data such as the beacon order and superframe order, the GTS descriptors, the pending data information, among others. In beacon-enabled networks, the devices associated to a Coordinator use the beacon to synchronize and to be informed of relevant information.

The implementation of the beacon construction is done during every inactive period of the Coordinator, since it is a time consuming procedure and at the beginning of the next active period the beacon must be ready for transmission. In this implementation, the function *create_beacon()* is used for that purpose. In this function, the beacon construction has the following steps: (1) the MHR (MAC Header) is defined with the short address of the Coordinator and with a broadcast destination address (0xffff); (2) the superframe specification defines the PAN parameters such as the beacon order and the superframe order parameters, among others; (3) the Coordinator constructs the GTS descriptor fields including the allocated and deallocated GTS, if there are any; (4) if the Coordinator has pending data to be transmitted to an associated device, it constructs the pending address descriptors. This procedure takes place in the MAC sub-layer. If the Coordinator wants to change some of the PAN parameters, the NWK layers can issue the *MLME-START.request* primitive with the new values. The beacon is automatically updated in the following superframe. In Figure 36, the flow chart represents the steps for building a beacon frame in the *create_beacon()* function.

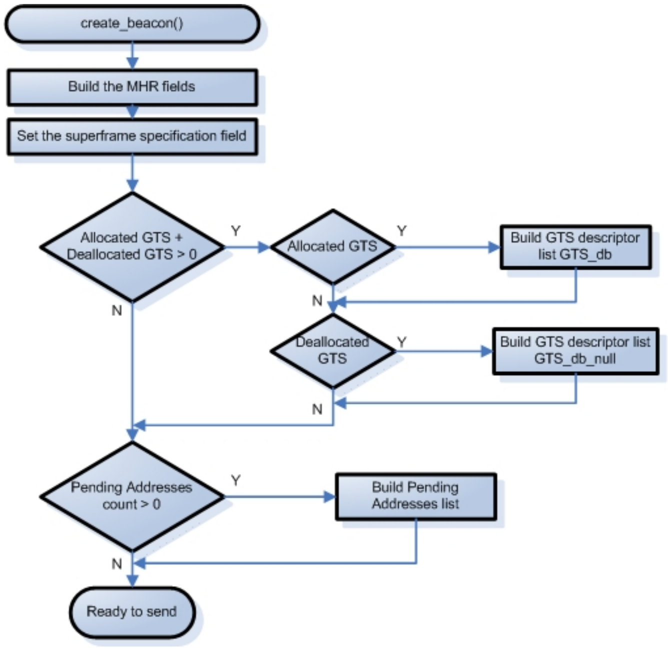

create_beacon()
Build the MHR fields
Set the superframe specification field
Allocated GTS + Deallocated GTS > 0
Y
N
Allocated GTS
Y
N
Build GTS descriptor list GTS_db
Deallocated GTS
Y
N
Build GTS descriptor list GTS_db_null
Pending Addresses count > 0
Y
N
Build Pending Addresses list
Ready to send

Figure 36 - Beacon creation flow chart

At the beginning of each superframe. all associated devices receive a beacon frame. After the processing of the beacon, the NWK layer is signalled with the information retrieved from the beacon processing. The processing of the beacon information is made in the *process_beacon(MPDU *packet)* function. This processing is divided into the following steps: (1) processing of the superframe specification field in order to read the values of the beacon order, the superframe order and the final CAP time slot; (2) compute the superframe duration, the beacon interval, the time slot and the backoff period in symbols; (3) process the GTS characteristics and, if there are GTS descriptors in the GTS list, compute each one in order to verify and search the correspondent device address. If the address is present, the device evaluates the descriptor information and updates its own GTS information, either for an allocation, deallocation or a confirmation of the current allocation request. This update is very important because the time slot(s) assigned to the device may change due to a reorder of the GTS descriptors in the Coordinator; (4) process the pending addresses information; (5) build the PAN descriptor information to inform the upper layer; (6) synchronize the device asynchronous timer with the Coordinator. In order to have a correct synchronization, the device must take into account the transmission time, transmission delay, the packet reception time and the packet processing time. On each *timer.fire()* event in the *TimerAsync* component, a counter is set to zero when the transceiver starts receiving data or the Start-of-Frame Delimiter (SFD) pin goes high. When the device wants to synchronize, it reads the SFD counter (*process_frame_tick_counter* variable in the *timer.fire()* function) to compute the processing time of the frame. The synchronization also takes into account the possible variables so that the internal time of the *TimerAsync* component can be set with the appropriate start value; (7) signal the NWK layer using the *MLME_BEACON_NOTIFY.indication* primitive with the PAN descriptor information.

3.4.5 Slotted CSMA/CA algorithm

The CSMA/CA algorithm is used when a device wants to transmit within the CAP. The transmission of beacon frames, acknowledgement frames and data frames in the CFP do not use the CSMA/CA algorithm; instead they are sent directly without any channel assessment.

The TinyOS is an event-driven operating system, meaning that all the mechanisms must be implemented with events, being hardware events or timer events. This way, it is not possible to have a process based concurrency model where it is possible to have multiple processes running at the same time and a process can wait while others are running. Therefore, the CSMA/CA has to be implemented based on timer events and state machines. The implementation of the CSMA/CA involves several functions and global variables, used in conjunction with the *TimerAsync.backoff_fired()* timer events. The function *send_frame_csma()* is called to start the application of the CSMA/CA mechanism and if the channel is clear (reading the *Clear Channel Assessment* (CCA) pin of the CC2420 transceiver) and there is data to be sent, it will send the frame. If there is no data in the *send_buffer,* the send procedure aborts. The function *send_frame_csma()* is called in the following cases:

- When a frame is created and it is already in the send buffer (ready to be send);

- When the last frame was successfully transmitted and there is still data in the buffer (ready to be send);
- At the beginning of the CAP, when the *TimerAsync.bi_fired()* timer event fires;
- In the retransmission of a frame requiring an acknowledgment;
- After a failed transmission and when there is still data in the buffer ready to be send.

When the function is called, it checks if the transmission occurs in the CFP and if there is data ready to be sent. If these conditions are true, the function sets the boolean variable *performing_csma_ca* to true, meaning that the execution of algorithm is in progress, and calls the *perform_csma_ca()* function (Figure 37), to proceed. Although the receiver of the device is enabled during the channel assessment part of this algorithm, the device must discard any frames received during this period. At this point, the algorithm is in step 1 (refer to Figure 16 in Chapter 2). Depending on the run mode of the device (beacon-enabled or non beacon-enabled), the function *perform_csma_ca()* follows with the execution of the slotted (beacon-enabled) or the unslotted version (non beacon-enabled). In the unslotted version the function calls the *init_csma_ca()* function to initialize the needed variables *NB* (number of backoffs) and *BE* (backoff exponent). The function also initializes the variable *delay_backoff_period* with a random value and sets the *csma_delay* boolean variable to start the delay mechanism, implemented in the *TimerAsync.backoff_fired()* timer event.

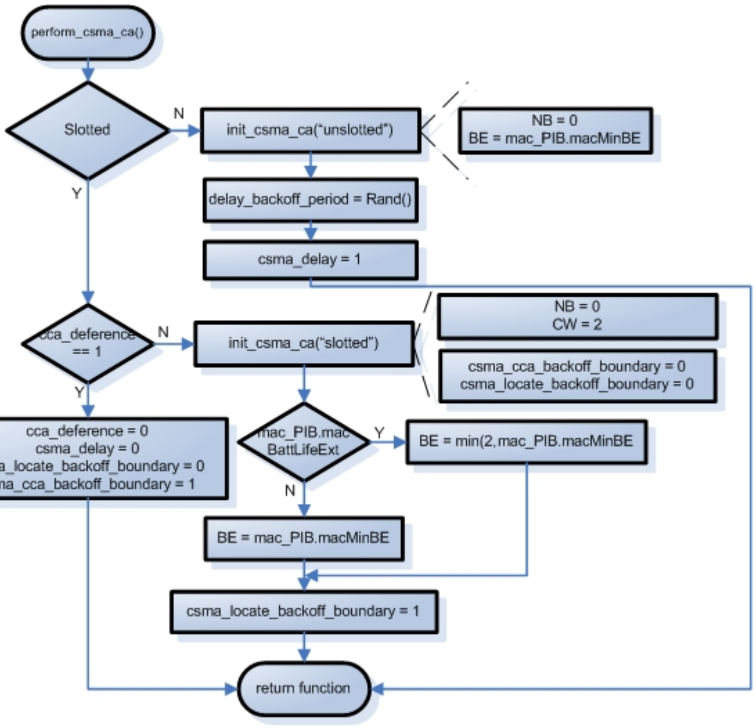

perform_csma_ca()
Slotted
N
Y
init_csma_ca("unslotted")
NB = 0
BE = mac_PIB.macMinBE
delay_backoff_period = Rand()
csma_delay = 1
cca_deference == 1
N
Y
init_csma_ca("slotted")
NB = 0
CW = 2
csma_cca_backoff_boundary = 0
csma_locate_backoff_boundary = 0
cca_deference = 0
csma_delay = 0
csma_locate_backoff_boundary = 0
csma_cca_backoff_boundary = 1
mac_PIB.mac BattLifeExt
Y
N
BE = min(2,mac_PIB.macMinBE
BE = mac_PIB.macMinBE
csma_locate_backoff_boundary = 1
return function

Figure 37 - *perform_csma_ca()* function flow chart

In the slotted version, the function calls the *init_csma_ca()* to initialize the needed variables namely the contention window (*CW*), the number of backoffs (*NB*) and other auxiliary variables. The backoff exponent (*BE*) is set depending on the battery life parameter. The boolean variable *csma_locate_backoff_boundary* is set to true, triggering the location of the backoff boundary, implemented in the *TimerAsync.backoff_fired()* timer event. At this point, the algorithm is in step 2 (refer to Figure 16 in Chapter 2).

The next steps of the algorithm are triggered in the *TimerAsync.backoff_fired()* timer event, depending on the state of the auxiliary variables, as depicted in Figure 38.

Figure 37 shows the flow chart for the *perform_csma_ca()* function. The auxiliary variables used in the implementation of the timer event are the following:

- *csma_delay* – used in Step 2 of the algorithm, to trigger the count down of the delay backoff periods;
- *csma_cca_backoff_boundary* – used in the Step 3 of the algorithm, after the delay period is over to perform the channel assessment;
- *delay_backoff_period* – number of backoff interval of the delay period;
- *csma_locate_backoff_boundary* – used to locate the backoff boundary of the first channel assessment in the slotted version of the algorithm.

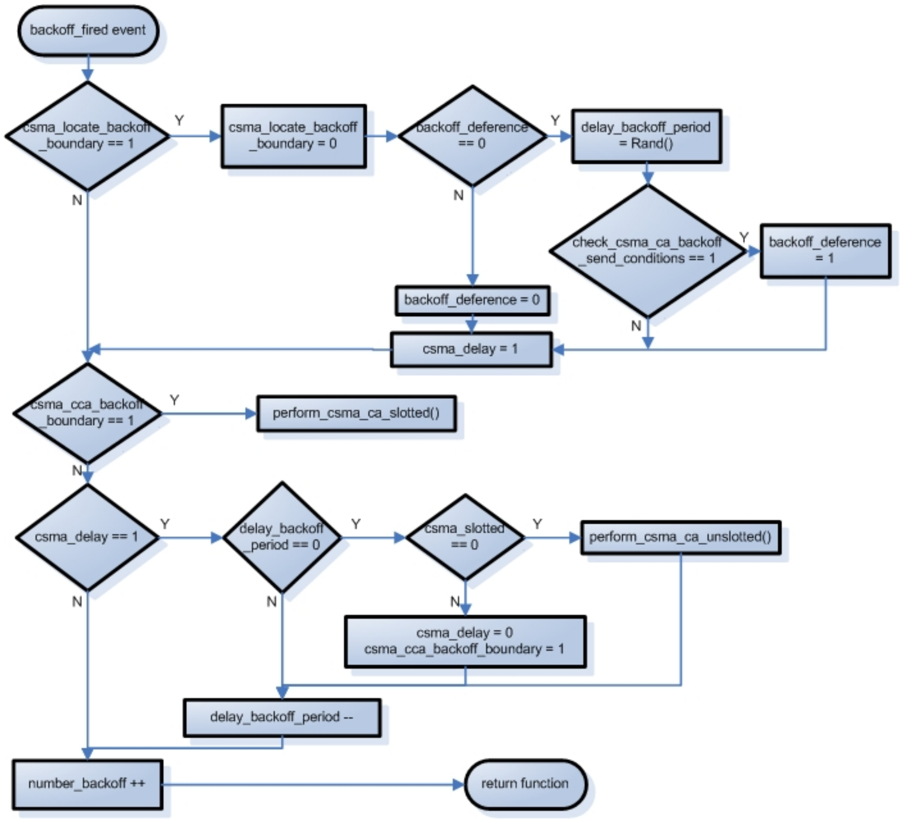

backoff_fired event
csma_locate_backoff_boundary == 1
Y
N
csma_locate_backoff_boundary = 0
backoff_deference == 0
Y
N
delay_backoff_period = Rand()
check_csma_ca_backoff_send_conditions == 1
Y
N
backoff_deference = 1
backoff_deference = 0
csma_delay = 1
csma_cca_backoff_boundary == 1
Y
N
perform_csma_ca_slotted()
csma_delay == 1
Y
N
delay_backoff_period == 0
Y
N
csma_slotted == 0
Y
N
perform_csma_ca_unslotted()
csma_delay = 0
csma_cca_backoff_boundary = 1
delay_backoff_period --
number_backoff ++
return function

Figure 38 - *backoff_fired* event flow chart

The function *perform_csma_ca_slotted()* implements the final steps of the algorithm. This function updates the global variables of the algorithm and sets the timer auxiliary variables, so that it can be called several times to accomplish the application of the CSMA/CA. The function *TOSH_READ_CC_CCA_PIN(),* available in TinyOS, is used to perform the clear channel assessment and evaluate if the channel is clear (1) or not (0). Figure 39 illustrates the operation of the *perform_csma_ca_slotted()* function.

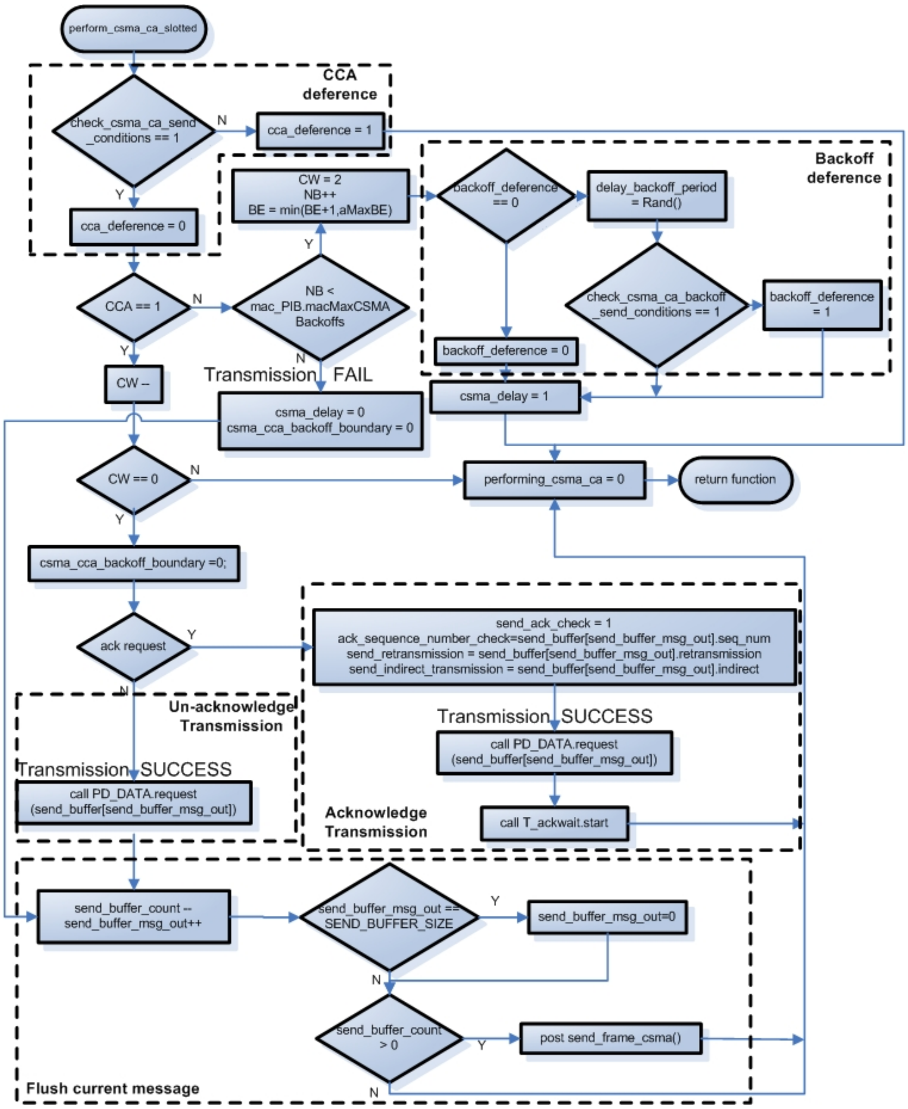

perform_csma_ca_slotted
CCA deference
check_csma_ca_send_conditions == 1
N
cca_deference = 1
Y
cca_deference = 0
CW = 2
NB++
BE = min(BE+1,aMaxBE)
Backoff deference
backoff_deference == 0
delay_backoff_period = Rand()
check_csma_ca_backoff_send_conditions == 1
backoff_deference = 1
backoff_deference = 0
CCA == 1
N
Y
NB < mac_PIB.macMaxCSMABackoffs
Y
N
Transmission FAIL
CW --
csma_delay = 0
csma_cca_backoff_boundary = 0
csma_delay = 1
CW == 0
N
Y
performing_csma_ca = 0
return function
csma_cca_backoff_boundary =0;
ack request
Y
N
send_ack_check = 1
ack_sequence_number_check=send_buffer[send_buffer_msg_out].seq_num
send_retransmission = send_buffer[send_buffer_msg_out].retransmission
send_indirect_transmission = send_buffer[send_buffer_msg_out].indirect
Un-acknowledge Transmission
Transmission SUCCESS
call PD_DATA.request (send_buffer[send_buffer_msg_out])
Transmission SUCCESS
call PD_DATA.request (send_buffer[send_buffer_msg_out])
Acknowledge Transmission
call T_ackwait.start
send_buffer_count --
send_buffer_msg_out++
send_buffer_msg_out == SEND_BUFFER_SIZE
Y
send_buffer_msg_out=0
N
send_buffer_count > 0
Y
post send_frame_csma()
N
Flush current message

Figure 39 - *perform_csma_ca_slotted()* function flow chart

The boolean variables *cca_deference* and *backoff_deference* are used to handle the transmission deference in the CSMA/CA mechanism.

The deference of the CCA occurs when the function *perform_csma_ca_slotted()* is called, after the backoff delay, and there is not enough time to complete the transmission of the frame (verified in the *check_csma_ca_send_conditions()* function) in the current superframe, as seen in Figure 39. When the *cca_deference* is set to 1, the CSMA/CA resumes the mechanism in the two CCA evaluations. This is accomplished in *perform_csma_ca()* function by setting to 0 the *csma_delay* and *csma_locate_backoff_boundary* variables, avoiding the random backoff delay and setting to 1 the *csma_cca_backoff_boundary* that triggers the call of the *perform_csma_ca_slotted()* function in the *backoff_fired* event.

The deference of the backoff countdown occurs when there is not enough time to complete the random backoff countdown (verified in the *check_csma_ca_backoff_send_conditions()* function) in the current superframe. If this occurs the backoff countdown must be resumed in the next superframe. The implementation of the backoff deference mechanism is done by avoiding the initialization of the random backoff (*delay_backoff_period* variable) in the *backoff_fired* event (Figure 38). If the *backoff_deference* is set to 1, in the beginning of the next superframe the *perform_csma_ca()* function is called and the location of the backoff boundary (*csma_locate_backoff_boundary* variable) occurs normally in the *backoff_fired* event, with the exception that the *delay_backoff_period* variable is not initialized, thus the backoff countdown (initialized in the last superframe) is resumed.

3.5 ZigBee Network Layer Implementation

3.5.1 Joining a ZigBee network

The join procedure is necessary for every ZigBee Router and ZigBee End Device to join the network. Only the ZC and ZRs are allowed to associate devices. The join procedure in the network layer is based of a distributed addressing scheme, as described Section 2.1.3. The join procedure described in this section is directed towards the construction of a cluster-tree topology. The join procedure is supported by the MAC sub-layer association mechanism as described in Section 2.2.4.

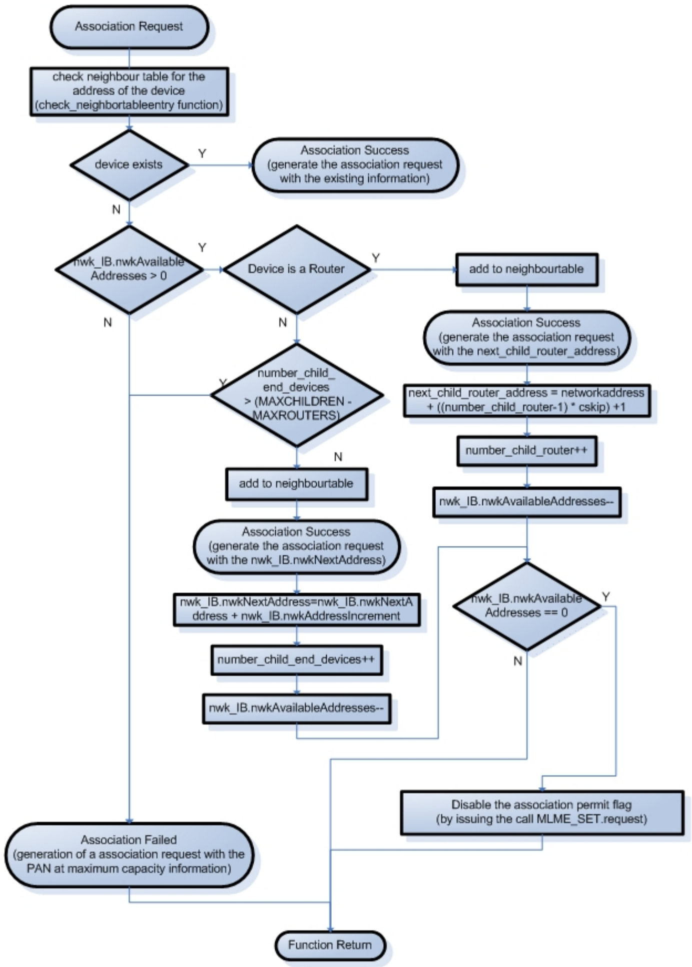

Association Request
check neighbour table for the address of the device (check_neighbortableentry function)
device exists
Y
Association Success (generate the association request with the existing information)
N
nwk_IB.nwkAvailable Addresses > 0
Y
Device is a Router
Y
add to neighbourtable
Association Success (generate the association request with the next_child_router_address)
next_child_router_address = networkaddress + ((number_child_router-1) * cskip) +1
number_child_router++
nwk_IB.nwkAvailableAddresses--
N
N
number_child_ end_devices > (MAXCHILDREN - MAXROUTERS)
Y
N
add to neighbourtable
Association Success (generate the association request with the nwk_IB.nwkNextAddress)
nwk_IB.nwkNextAddress=nwk_IB.nwkNextAddress + nwk_IB.nwkAddressIncrement
number_child_end_devices++
nwk_IB.nwkAvailableAddresses--
nwk_IB.nwkAvailable Addresses == 0
Y
N
Disable the association permit flag (by issuing the call MLME_SET.request)
Association Failed (generation of a association request with the PAN at maximum capacity information)
Function Return

Figure 40 - *MLME_ASSOCIATE.indication* flow chart

When a device receives an association request command frame the MAC sub-layer issues the *MLME_ASSOCIATE.indication* primitive to the network layer. Figure 40 depicts a flow chart showing the association procedure of a parent device.

In the *MLME_ASSOCIATE.indication* primitive, the parent will first check for the associating device address in the neighbour table, verifying if it is a reassociation and, in that case, the association is successful and the parent generates an association response command frame with the already stored information. If the device does not exist, the parent must verify what type of device (ZR or ZED) is trying the association. The type of device defines the address that must be assigned. Nevertheless, for a successful association, the variable *nwk_IB.nwkAvailableAddresses* in the NWK PAN Information Base (NWK PIB) must be greater than zero.

If the associating device is a ZR, the parent adds its information to the neighbour table and sends the association response command with the short address previously calculated in the *next_child_router_address* variable. Then, the parent updates the future child router address by calculating the formulation in (3.1) followed by an increment of the number of associated child routers (*number_child_router*).

$$next\_child\_router\_address = networkaddress + ((number\_child\_router - 1) \times Cskip) + 1 \qquad (3.1)$$

Note that the *Cskip* parameter is calculated in the initialization of the node and the *networkaddress* variable is the parent's device short address.

If the associating device is a ZED, the parent also adds its information to the neighbour table and generate the association respond command with the short address previously calculated in the *nwk_IB.nwkNextAddress* variable. After the response generation, the *nwk_IB.nwkNextAddress* variable is incremented by *nwk_IB.nwkAddressIncrement* followed by an increment of the number of associated child end devices.

There is a need to differentiate the type of associating device (ZR or ZED) and separately count them and their assigned addresses in order to comply with the address scheme (refer to Section 2.1.3). This address distribution enables the construction of the cluster-tree topology

3.5.2 Tree Routing

The Tree Routing procedure, as briefly explained in Section 2.1.5, is based on the addresses of the devices. When the MAC sub-layer of a device receives a data frame it issues the *MCPS_DATA.indication* primitive to the NWK layer. Figure 41 depicts a flow chart showing the procedures when the NWK receives a data frame.

The NWK layer, upon reception of a data frame, will first verify if the routing destination field equals its own short address and, if true, it transfers the data payload to the upper layer, by issuing the *NLDE_DATA.indication* primitive. If the routing destination address is not for itself, the device must calculate the next hop destination address.

In case of a ZigBee Coordinator, if the destination of the data frame is its own child, after checking in the neighbour table, it assigns the next hop with the short address of the respective child, present in its neighbour table. Otherwise, it needs to calculate the next hop by applying the Tree Routing formula for that effect (refer to Section 2.1.5). Note that routing is always downstream, since the ZC has no parent.

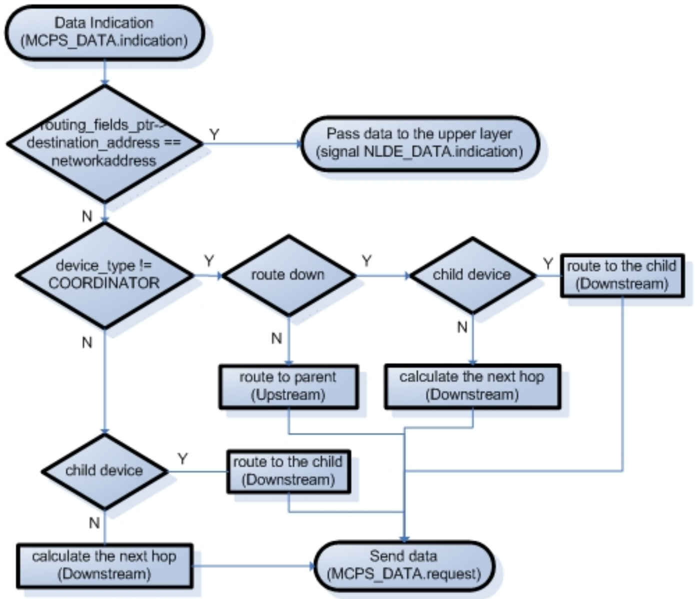

Data Indication (MCPS_DATA.indication)
routing_fields_ptr-> destination_address == networkaddress
Y
Pass data to the upper layer (signal NLDE_DATA.indication)
N
device_type != COORDINATOR
Y
route down
Y
child device
Y
route to the child (Downstream)
N
N
route to parent (Upstream)
N
calculate the next hop (Downstream)
child device
Y
route to the child (Downstream)
N
calculate the next hop (Downstream)
Send data (MCPS_DATA.request)

Figure 41 - MCPS_DATA.indication flow chart

In case of a ZigBee Routers there is an initial verification if the destination is upstream or downstream. This verification is done by applying the following conditions:

$$\begin{gathered} \mathit{networkaddress} < \mathit{routing\_fields\_ptr{-}{>}destination\_address} \\ \text{and} \\ \mathit{routing\_fields\_ptr{-}{>}destination\_address} < (\mathit{networkaddress} + \mathit{cskip\_routing}) \end{gathered} \tag{3.2}$$

If the above conditions are true, then routing is downstream. The device checks if the destination is a child device (by consulting its neighbour table) and if not it calculate the next hop by applying the Tree Routing formula for that effect (Section 2.1.5). If the conditions are false, the device just routes up to its parent. After the next hop decision, the message is transmitted by issuing the *MCPS_DATA.request* primitive to the MAC sub-layer.

The data frame transmission procedure is similar to the routing mechanism. After the creation of the frame, the device must assign a destination address to the routing fields. If the device is a ZED, the only option is to route to its parent (upstream). Otherwise, if the device is the ZigBee Coordinator or a ZigBee Router, it must check if the destination is a child device or must calculate the next hop address. Figure 42 depicts a flow chart of the *NLDE_DATA.request* primitive used to request a data transmission.

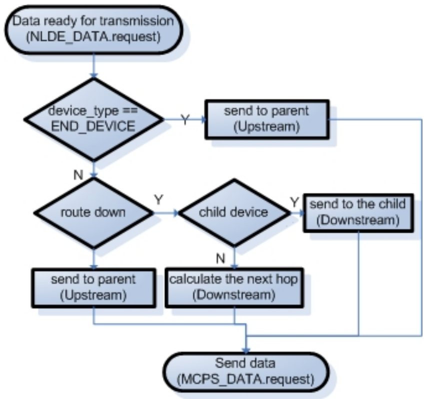


Figure 42 - NLDE_DATA request flow chart

3.6 Conclusions

The main challenge encountered while implementing the IEEE 802.15.4 protocol and the ZigBee Network Layer was related to hardware specificities and constraints. In that aspect the MICAz motes revealed to be more limited that the TelosB. Nevertheless, none of them provide enough processing power and radio performance for an implementation that fully complies with the IEEE 802.15.4 standard timing constraints, especially for small beacon orders ($BO < 2$) and superframe orders ($SO < 2$). Additionally, the available memory size is rather scarce. However it is possible to achieve a reasonable operational behaviour for higher beacon orders.

The timing requirements of the IEEE 802.15.4 protocol are very demanding. In the beacon-enabled mode all the devices must synchronize with a Coordinator beacon in order to align their superframe. If a device loses synchronization it cannot operate in the PAN and if it is not correctly synchronized there is the possibility of collisions in the GTS slots (when the CAP overlaps the CFP). As experienced during this implementation, the loss of synchronization can be caused by multiple factors: the processing of the beacon frame for low *BO/SO* configurations, the mote stack overflow that result in a block or a hard reset, the unpredictable delay of the wireless communications and the low processing power of the microcontroller in conducting some of the protocol maintenance tasks (e.g. creating the beacon frame, the maintenance of GTS expiration and the indirect transmissions).

The implementation of the CSMA/CA algorithms is also very demanding concerning the timers precision, since the IEEE 802.15.4 protocol defines that each backoff corresponds to 20 symbols (320μs). A first difficulty in the implementation of the beacon-enabled mode was related to the TinyOS management of the hardware timers provided by the motes, which do not allow having the exact values of the beacon interval, superframe, time slots and backoffs durations as specified by the IEEE 802.15.4 standard. This discrepancy, however, does not impact the correct behaviour of the implemented protocol, if the same mote platforms are used in the experiments (at least as ZC and ZRs), it is possible to experience a coherent network behaviour.

The frequency of the asynchronous software events (Figure 29 in Section 3.4.1), the hardware events and the microprocessor processing ability may lead to an insufficient processing power left to execute remaining protocol and high level application tasks as a great amount of interrupts have to be processed in short periods of time.

The IEEE 802.15.4 protocol has no reference concerning the implementation of the buffer mechanisms, which impacts on the correct behaviour of the protocol. On one hand, the protocol must avoid excessive memory copy operations because they may cause synchronization problems and are very time consuming. On the other hand, the buffers have to be small and very well managed due to the devices memory constraints.

Another constraint of the IEEE 802.15.4 Physical Layer is the turnaround time of 12 symbols (192 µs), the time that the CC2420 radio transceiver takes to switch from receive mode to transmit mode and vice-versa, to acknowledge messages. Unfortunately, this is not possible to achieve in most IEEE 802.15.4-compliant radio transceivers. For instance, the Chipcon CC2420 can take up to 192 µs to switch between these two modes, leaving no time for data transitions between the MAC sub-layer, the PHY layer and the chip transmit memory space.

Also, TinyOS imposes some overhead [57] in the primitive operations (e.g. posting tasks, calling commands) that may be considerable, taking into account the need to comply with the most demanding operational modes of the IEEE 802.15.4.

In spite of having no comparison between this TinyOS implementation with others, it is possible to assume that an implementation without any base operating system (OS) could have better performance results, since TinyOS can introduce some unnecessary processing overhead in its internal operations. In fact, there is a tradeoff between the benefits of using an OS, bringing in several functionalities that enable a faster development of high end applications and the processing overhead introduced. Considering that the embedded devices have limited resources it is reasonable to assume that a non-OS based implementation can be more optimized but not so flexible.

Nevertheless, the implementation still has space for improvement, as it is a work-in-progress and it is envisaged to implement the full functionalities of the IEEE 802.15.4 and the ZigBee Network Layer. Also, we aim at the migration of the protocol stack from TinyOS 1.15 to 2.0, in collaboration with the TinyOS Network Protocol Working Group [58].

The ZigBee specification still has several open-issues and some of them are very important to an effective use of this protocol. In spite this Thesis only focused on the Network Layer it is possible to draw some conclusions on the ZigBee design. There was a major evolution between the ZigBee Network layer of the first specification (2004) and the latest one (2006). Several issues were corrected while other were added introducing more complexity but with the advantage of adding more flexibility. Once more, the mesh network topology evolved with the addition of new functionalities, such as the possibility of multicast transmissions and a source route routing protocol, while the cluster-tree synchronized topology was left behind. Nevertheless, there are still many open-issues in the ZigBee standard that leaves room for improvement especially in the Cluster-Tree Network topologies.

Chapter 4

Engineering ZigBee Cluster-Tree Networks

ZigBee Cluster-Tree networks are quite appealing for supporting WSN applications with QoS requirements. Nevertheless, there are several open issues and ambiguities in the ZigBee Standard that turn its practical use a challenging task. This chapter addresses the way to engineer a ZigBee Cluster-Tree Network, particularly how to schedule beacons/superframes in each cluster such that no beacon collisions occur.

4.1 Introduction

The current IEEE 802.15.4/ZigBee specifications restrict the synchronization in the beacon-enabled mode to star-based networks, while it supports multi-hop networking using the peer-to-peer mesh topology, but with no synchronization. Even though both specifications mention the possible use of cluster-tree topologies, which combine multi-hop and synchronization features, the description on how to effectively construct such a network topology is missing.

The beacon-enabled mode of the IEEE 802.15.4/ZigBee protocol standards suffer from lack of scalability since, inherently to its operational behaviour, it is limited to star based networks. In the star topology the ZigBee Coordinator is a central node that periodically transmits beacons for synchronizing the nodes in its vicinity and centralizes all the communication (data exchange between nodes must be relayed by the ZigBee Coordinator). As a consequence, the network coverage is limited to the transmission range of the ZigBee Coordinator, which restricts the geographical region under monitoring/control. This is particularly unsuitable for WSNs, which are commonly accepted to be large-scale. Therefore, there is a paradox between supporting scalability at the cost of energy consumption and delay guarantees – mesh – and being able to guarantee real-time and energy-efficient communications – star. It would be more appropriate if both features (synchronization and scalability) could be simultaneously supported into the same network.

The Cluster-Tree network concept is outlined in the ZigBee specification. However, in the IEEE 802.15.4/ZigBee standards there is not a clear description on how the cluster-tree model can be implemented. The available information regarding this topology gives a broad overview on how the cluster-tree network should operate and some details on the tree routing algorithm proposed by MOTOROLA [34].

More specifically, the Cluster-Tree model includes more than one ZigBee Router that periodically generate beacons to synchronize nodes (or clusters of nodes) in their neighbourhood. If these periodic beacon frames are sent in an unorganized fashion, without any particular schedule, they will collide with each other or with other frames. These collisions results in the loss of synchronization between a parent ZigBee Router and their child devices, which prevents them to communicate. Therefore, beacon frame scheduling mechanisms must be defined to avoid beacon frame collisions in ZigBee cluster-tree networks.

Only some basic approaches dealing with this problem were proposed for discussion by the Task Group 15.4b [5], which is a group aiming to improve some inconsistencies of the original specification. However, no algorithms for providing collision-free beacon frame generation have been proposed so far.

4.2 Related Work

Clustering and multi-hop network synchronization are common problems in Wireless Sensor Networks that have been addressed in many research works (e.g. [59-62]). The RT-Link, presented in [59], provides a multi-hop synchronization scheme but it does not consider clustering, although being similar to the IEEE 802.15.4 protocol. The LEACH protocol, proposed in [60], is a clustering-based protocol using a randomized rotation and selection of cluster-heads to optimize energy consumption. This protocol does a random selection of cluster-heads and all the other nodes decide to which cluster they belong by informing the corresponding cluster-head (using CSMA/CA) of their decision. After the reception of all join requests, cluster-heads compute a TDMA (Time Division Multiple Access) schedule according to the number of nodes in their cluster. This schedule is broadcast back to the node in the cluster. Inter-cluster interference is mitigated using different CDMA (Code Division Multiple Access) codes in each cluster. This clustering and synchronization approach differs from the ZigBee approach in three aspects, which turns our problem quite different.

- concerning clustering in ZigBee Networks, the ZC and the ZRs (or cluster-heads) are fixed (do not change during run-time);
- the synchronization is not made using a TDMA schedule, but by means of periodic beacon frame transmissions, which has the advantage of higher flexibility (TDMA is not scalable and is vulnerable to dynamic network changes);
- ZigBee does not allow the use of CDMA to avoid inter-cluster interferences, which leads to collisions between beacon and data frames issued in different clusters.

Also, the impact of beacon collision in a ZigBee node is the loss of synchronization and consequently the node becomes disconnected an prevented to communicate with the network. Hence, there is a need to schedule different beacon frames from different clusters to avoid beacon frame losses that lead to undesirable synchronization problems.

This problem is relevant toward the real applicability of ZigBee Cluster-Tree networks. In that sense, the Task Group 15.4b [5] has been working on an improved version of the IEEE 802.15.4 standard and proposed for discussion some basic approaches for avoiding beacon frame collisions that may be adopted in the upcoming extension of the standard. Currently there are two approaches (as described in Section 4.4). A first approach, called the Beacon-Only Period approach, consists in having a time window at the beginning of each superframe reserved for beacon frame transmissions. The second approach, based on time division, proposes that beacon frames of a given cluster are sent during the inactivity periods of the other clusters. However, these approaches do not define how to schedule beacon frame transmission, more specifically how to choose the time offsets of different beacons. Surprisingly, these approaches, discussed within the Task Group 15.4b, were not fully included in the new versions of the standard IEEE 802.15.4b [4] and ZigBee specification [6].

4.3 The Beacon Collision Problem

It is easy to perceive that in ZigBee Cluster-Tree networks, where each cluster must send a beacon to synchronize the nodes in their vicinity, sending beacons without any special care on timing issues may result in a beacon frame collision in the nodes that are in the range of more that one Coordinator (ZigBee Coordinator or ZigBee Routers). The beacon frame collision problem in cluster-tree ZigBee networks has been addressed as 'Request for Comments' in the Task Group 15.4b.

Two types of beacon frame collisions were identified: (1) direct beacon frame collisions and, (2) indirect beacon frame collisions, which are briefly explained next.

4.3.1 Direct Beacon Frame Collisions

Direct beacon frame collisions occur when two ore more Coordinators are in the transmission range of each other (direct neighbours or parent-to-child relation) and send their beacon frames at approximately the same time, as shown in Figure 43.

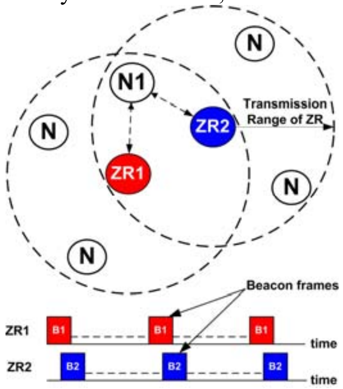


Figure 43 - Direct Beacon Frame collisions

In this figure, assuming that node N1 is a child of ZR1, which sends its beacon frame at approximately the same time as ZR2, node N1 loses its synchronization with its parent ZR1 due to the collision of the two beacon frames.

4.3.2 Indirect Beacon Frame Collisions

Indirect beacon frame collisions occur when two ore more Coordinators cannot hear each other, but have overlapped transmission ranges (indirect neighbours) and send their beacon frames at approximately the same time, as shown in Figure 44. In this figure, node N1, which is located in the overlapped region of the transmission ranges of ZR1 and ZR2, will not be able to synchronize with its parent since the beacon frames from ZR1 and ZR2 will collide.

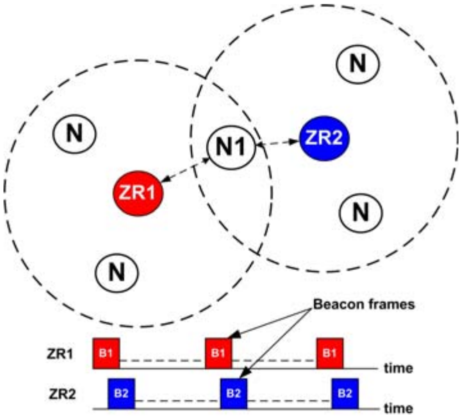


Figure 44 - Indirect Beacon Frame Collisions

It is important to refer that collisions between data and beacon frames may also happen when a Coordinator sends its periodic beacon frame during the active period of an adjacent cluster.

The beacon collision problem in a ZigBee cluster-tree network can be roughly formulated as follows: Given an IEEE 802.15.4/ZigBee network with several Coordinators generating periodic beacon frames and organized in a cluster-tree topology, how to schedule the generation time offsets of beacon frames issued from different Coordinators to completely avoid beacon frame collisions with each other and with data frames [28].

4.4 Task Group 15.4b approaches for Beacon Collision Avoidance

Since no mechanism to avoid beacon frame collisions is considered in the current IEEE 802.15.4/ZigBee specifications, some proposals have been discussed by Task Group 15.4b. These approaches were proposed as pattern ideas to trigger the design of solutions to the direct and indirect beacon frame collision problems.

Two approaches were proposed to avoid the direct beacon frame collision problem: (1) A time division approach and, (2) a beacon-only period approach. Also, regarding the indirect beacon collision problem, Task Group 15.4b proposed two alternatives: (1) a reactive approach and, (2) a proactive approach.

4.4.1 Direct Beacon Collision Avoidance – Time Division Approach

In this approach, time is divided such that beacon frames and the superframe duration of a given Coordinator are scheduled in the inactive period of its neighbour Coordinators, as shown in Figure 45. Each Coordinator uses a starting time relative to the Coordinator beacon (*Beacon_Tx_Offset)* to transmit its beacon frames. The beacon offsets must be different for each router so that each active period uses a different time window. This approach requires that a Coordinator wakes up both in its active period and in its parent's active period to track its beacon. Communication between different clusters must be accomplished by the means of indirect transmissions. Observe that *Beacon_Tx_Offset* must be chosen adequately, not only to avoid beacon frame collisions, but also to enable efficient utilization of inactive periods, thus maximizing the number of clusters in the same network.

The limitations of this approach are:

- It constraints the duty cycles, since they will be dependent on the number of interfering Coordinators (which must operate in different time windows);
- Direct communication between sibling Coordinators (Coordinators with the same parent) is not possible, since adjacent cluster operate at different time windows;
- It is not possible to allocate GTS time slots for data transmission between the different clusters due to the impossibility of direct transmissions.

The density of devices that can be supported is inversely proportional to the ratio of the beacon order and superframe orders, assuming that all *BOs* and *SOs* are equal for all clusters. This problem is more challenging when the superframe orders and beacon orders are different from one cluster to another.

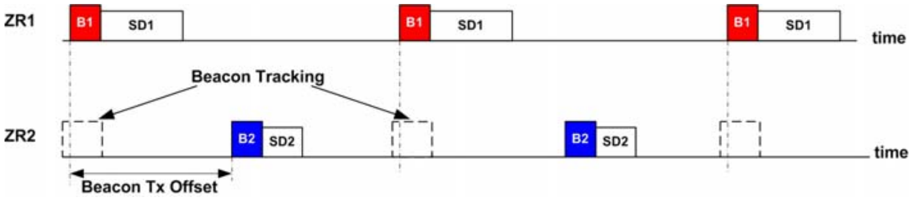


Figure 45 - Beacon Frame Collision Avoidance - The Time Division Approach

This approach has been included in the 2006 ZigBee standard [6]. Nevertheless, the details on how to schedule and implement the different Coordinator beacon transmission offset is missing.

4.4.2 Direct Beacon Collision Avoidance – Beacon-Only Period Approach

In this approach, a time window, denoted as Beacon-Only Period, is reserved at the beginning of each superframe for the transmission of beacon frames in a contention-free fashion, as depicted in Figure 46. Each Coordinator chooses a sending time offset by selecting a contention-free time slot (CFTS) such that its beacon frame does not collide with beacon frames sent by its neighbours. The advantage of this approach as compared to the time division approach is that the active periods of the different clusters start at the same time, thus direct communication between neighbour nodes is possible, and there is no constraint on the duty cycle.

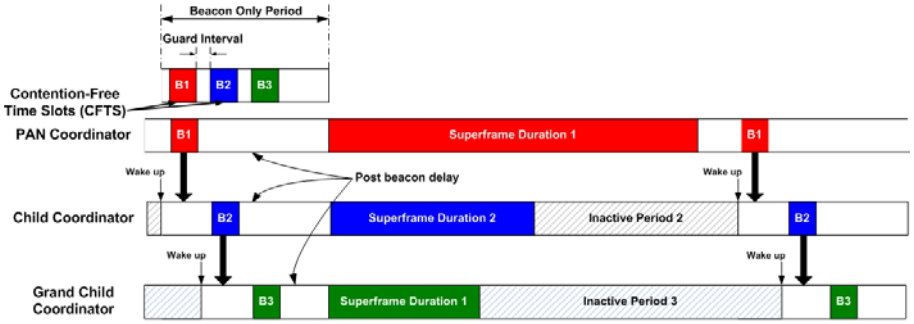


Figure 46 - Beacon Frame Collision Avoidance - The Beacon-Only Period

The main complexity of this approach is the dimensioning of the duration of the beacon-only period for a given network topology. This duration depends on the number of nodes in the network, their parent-child relationship and also the scheduling mechanism used to allocate the CFTS to each Coordinator. Additionally, the GTS mechanism cannot be implemented (at least in accordance to the specification), since transmission from nodes belonging to different clusters may collide. Thus, transmissions are only allowed during the CAP, which is shared by different clusters. Importantly, oppositely to the time division approach, the beacon-only period approach implies a non-negligible change to the standard protocol

4.4.3 Indirect Beacon Frame Collision Avoidance

The problem of indirect beacon frame collisions is more complex than the one of direct beacon frame collisions. There is a need to not only know the neighbour Coordinators, but also all other Coordinators that are two-hops away. The two alternatives proposed by the Task Group 15.4b are based on a reactive and a proactive behaviour of the Coordinator nodes.

In the reactive approach, a Coordinator does not carry any specific procedure to avoid indirect beacon frame collision during the association with its parent. Once a beacon frame conflict is detected by a given node, it initiates a recovery procedure to resolve the problem, which may take a long time.

In the proactive approach, Coordinators try to avoid the indirect beacon frame conflict at the association phase by the collection of specific data about beacon frame transmission times of their neighbours. In this approach, each potential Coordinator must have the ability to forward the beacon frame time offset of its parent to its neighbour Coordinators. This approach is more complex than the reactive approach, but it completely avoids beacon frame collisions during network run-time.

4.5 Time Division Beacon Scheduling Mechanism (TDBS)

This section presents the algorithm for scheduling the beacon transmission of the Coordinator in a cluster-tree network, the Superframe Duration Scheduling (SDS) [28], and also describes the particularities of implementing this mechanism over the IEEE 802.15.4 and the ZigBee Network layer.

The SDS algorithm proposed and presented in [28] and in Annex D, aims at an efficiently organization of the superframe durations of different Coordinators in a non overlapping manner, based on their superframe orders and beacon orders. The SDS algorithm performs the schedulable analysis of a set of superframes with different durations and beacon intervals, and provides a schedule if the set is schedulable. The algorithm also holds for equal superframe durations. First, for being schedulable, it is necessary that the set of devices satisfy that the sum of the duty-cycles percentage is lower than 1. Then, the SDS identifies the major and the minor beacon intervals. The schedule of the devices begins with the devices that have smaller beacon order and superframe duration.

To give a practical intuition of the SDS considers the PAN configuration example in Table 6, where there are 6 Coordinator devices with an envisage set of superframe durations (SD) and beacon intervals (BI).

Table 6 - PAN configuration example

Coordinator	SD	BI
C1	4	16
C2	1	8
C3	2	16
C4	1	32
C5	4	32
C6	2	16

By applying the SDS algorithm (refer to Annex D) the result is a major beacon interval cycle of 32 and a minor cycle of 8.

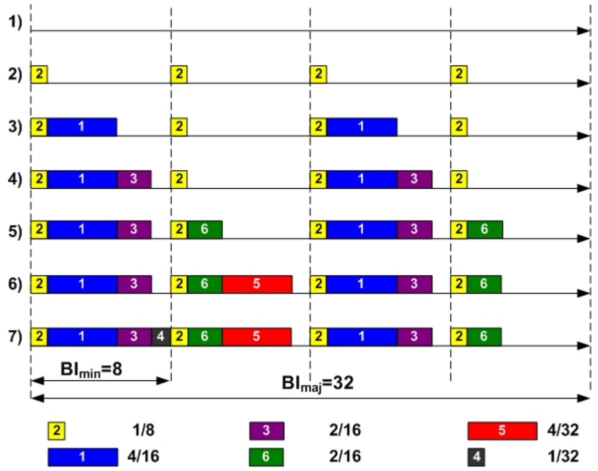


Figure 47 - SDS schedule example

The application of SDS algorithm is presented in Figure 47, where the lines are single steps of the algorithm and the last line presents the final schedule. Observe in Figure 47-7 that this set of coordinators, arranged as (C2, C1, C3, C6, C4, C5), is schedulable since all superframe durations are periodic and are not overlapping within the major cycle.

4.6 Implementation Details of the TDBS Approach

The Time Division Beacon Scheduling (TDBS) mechanism can be implemented in a simple manner, with only minor add-ons to the protocol [29].

The implementation of this mechanism assumes the following:

1. The ZigBee Network Layer supports the tree-routing mechanism, thus the network addresses of the devices are assigned accordingly.
2. The ZigBee Coordinator is the first node broadcasting beacons in the network.
3. The ZigBee Routers start to send beacons only after a successful negotiation.
4. The same Beacon Interval (*BI*) is used by every ZigBee Router.

The TDBS approach relies on a negotiation prior to beacon transmission. Upon success of the association to the network, a ZR (initially behaving as a ZED) sends a negotiation message to the ZC (routed along the tree) embedding the envisaged (*BO*, *SO*) pair, requesting a beacon broadcast permit. Then, in the case of a successfully negotiation, the ZC replies with a negotiation response message containing a beacon transmission offset (the instant when the ZR must start transmitting the beacon). In case of rejection, the ZR must disassociate from the network.

Figure 48 depicts the architecture of the TDBS implementation in the IEEE 802.15.4/ZigBee protocol stack.

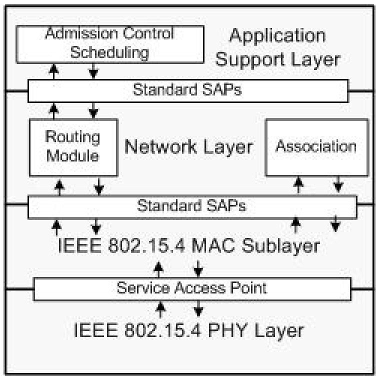


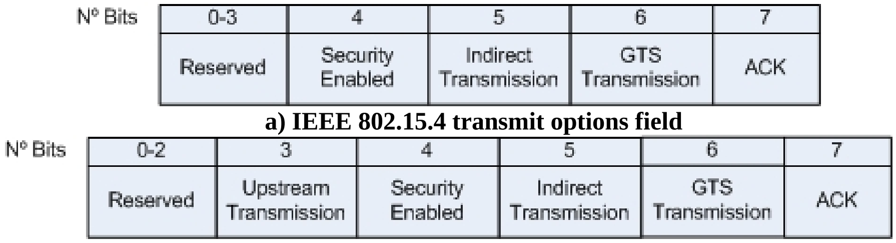
Figure 48 - Time Division Beacon Scheduling Implementation Architecture

The admission control algorithm is implemented in the Application Support Layer behaving as a service module of this layer.

The TDBS requires minor changes to the Network Layer. Thus, it is necessary to add a *StartTime* argument in the *MLME-START.request* primitive, as already proposed in the ZigBee Specification, and to the *NLME-START-ROUTER.request* primitive. The *StartTime* parameter will be used as a transmission offset referring to the parent ZigBee Router (ZR). In the ZC, the value of this parameter is 0.

After a successful negotiation of the beacon transmission, the ZR will have two active periods: its own (the superframe duration) and the parent's superframe duration. In its own active period, the ZR is allowed to transmit frames to its associated devices or relay frames to the descendant devices in the tree. The frames destined upstream are sent during its parent's active period. To accomplish this behaviour, there is a need to implement a different buffer mechanism for each message flow - the downstream to the device descendants and the upstream to the device ascendants.

The buffer mechanism is implemented in the MAC sub-layer, that uses the downstream buffer or the upstream buffer depending of the transmission options parameter of the *MCPS_DATA.request* primitive. The transmit options or *TxOptions* parameter, last argument of the primitive, define the transmission options for the data frame, allowing the frame to be sent in the GTS or during the CAP period. This parameter also defines if the transmission uses the upstream or the downstream buffer, as depicted in Figure 49.



Nº Bits	0-3	4	5	6	7
	Reserved	Security Enabled	Indirect Transmission	GTS Transmission	ACK

a) IEEE 802.15.4 transmit options field

Nº Bits	0-2	3	4	5	6	7
	Reserved	Upstream Transmission	Security Enabled	Indirect Transmission	GTS Transmission	ACK

b) Time Division Beacon Scheduling transmit options field

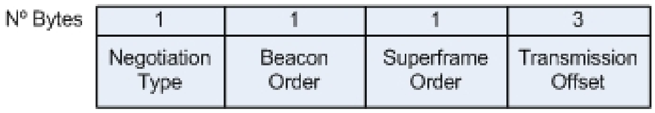
Figure 49 - Transmit options field comparison

During the ZR superframe, all the frames that need to be transmitted to its parent are stored in the upstream buffer. When the device enters the parent superframe, it tries to transmit the messages.

To enable the use of the two message buffers the device must wake up during its parent's superframe. This is accomplished by adding two new timer events to the MAC sub-layer. One is triggered at the beginning of the parent's superframe and turns *on* the transceiver in receive mode and another at the end turning the transceiver *off*.

Guaranteeing the synchronization of all devices is the major challenge in this implementation. The devices must be always synchronized with their respective parents.

The negotiation of the beacon transmission is performed through a simple protocol that uses the data frames payload with a predefined format.



Nº Bytes	1	1	1	3
	Negotiation Type	Beacon Order	Superframe Order	Transmission Offset

Figure 50 - Time Division Beacon Scheduling negotiation field

Figure 50 depicts the Time Division Beacon Scheduling negotiation frame format, which includes the following fields:

- *Negotiation type* – Indicates the type of the negotiation command. This field can have the following values: 1 for a negotiation request, 2 for a negotiation accept and 3 for a negotiation deny;
- *Beacon Order* – Indicates the beacon order of the ZR device;
- *Superframe Order* – Indicates the superframe order of the ZR device;
- *Transmission Offset* – Indicates the transmission offset schedule by the ZC in a negotiation accept command.

In the negotiation request, the Beacon Order and Superframe Order fields indicate the intended ZR superframe configuration. In the negotiation response, the configuration may not be the same as the requested. Instead, the ZC can assign a different *BO*, *SO* configuration according to its scheduling.

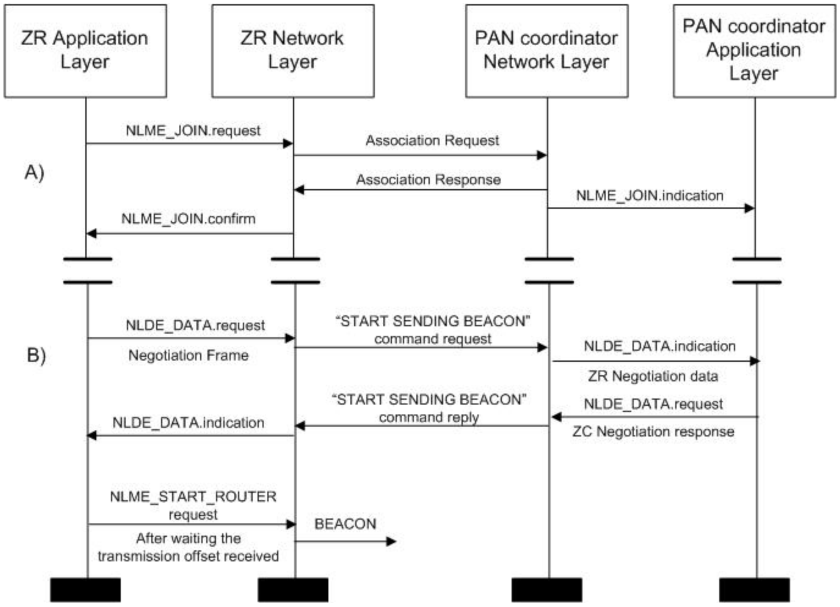


Figure 51 - Time Division Beacon Scheduling Negotiation diagram

Figure 51 depicts a diagram with the sequence of Network Layer events from the association of the ZR (Part A) until the beacon transmission after a successful negotiation (Part B).

4.7 Experimental Evaluation

This section presents the results of the implementation of the time division beacon scheduling approach. This experimental work demonstrates the feasibility of this approach using the TelosB motes [23]. Other experiments are presented in [29].

The network configuration/parameters presented in Figure 52 were considered.

In this example scenario, the cluster-tree network contains 15 cluster heads that consist of one ZC and 14 ZRs. The Beacon Order (*BO*) is set to 8 for all Coordinators, which gives a Beacon Interval of 245760 symbols (≅ 4122.624 ms). Hence, we must have at least 24 = 16 Beacon/Superframe time windows, each with duration of 15360 symbols (≅ 257.664 ms). This restricts the (maximum) Superframe Order (*SO*) to 4 (i.e. Superframe Duration (*SD*) = 15360 symbols). In our experimentation, we choose a *SO* = 4 (*SD* = 15360 symbols (≅ 257.664 ms)). The cluster-tree network parameters (for setting up the tree routing mechanism) consist in a maximum depth equal to *Lm=3*, a maximum number of child nodes per parent router equal to *Cm=6*, and a maximum number of child routers per parent router equal to *Rm=4*. As shown in

Figure 52, the network comprises the ZC at depth 0, two ZRs at depth 1, four ZRs at Depth 2 and eight ZRs at depth 3. The ZED (0x0400) was also considered for performing out a message routing test.

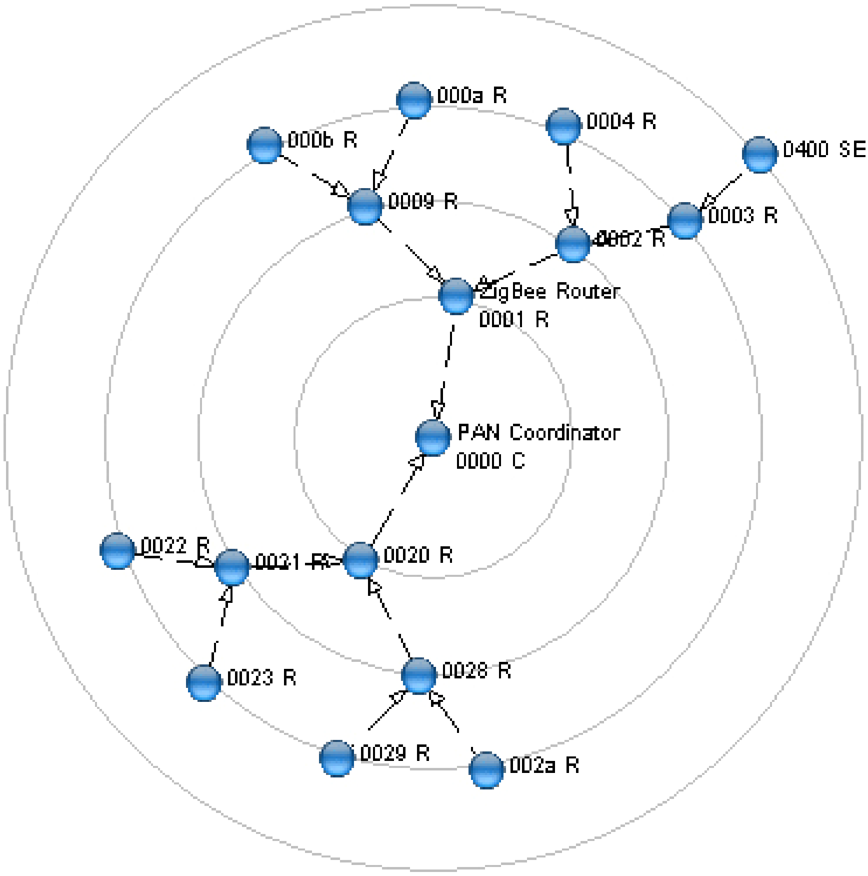


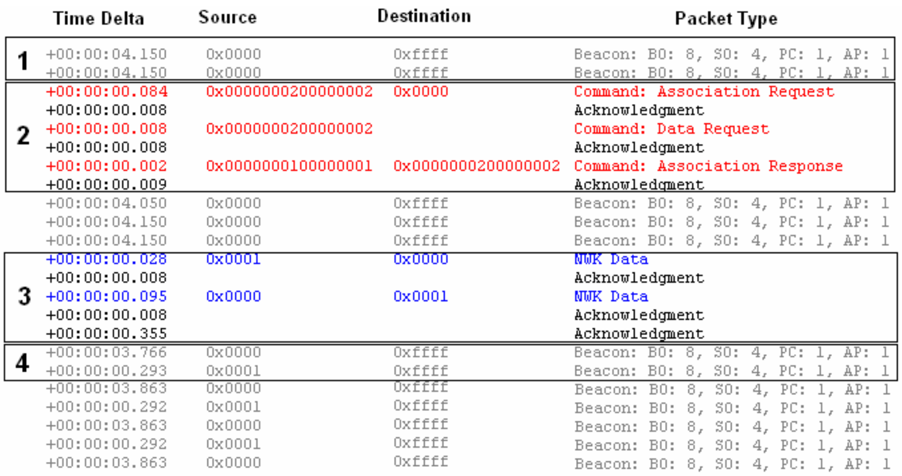
Figure 52 - Experimental cluster-tree network configuration

Table 7 shows the schedule for the beacon interval. Each associated ZR will be assigned the respective time window according to their position in the network tree. For viewing purposes, only the 2 last bytes of the short addresses are shown.

Table 7 - TDBS time window schedule

Time Window	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Short Address	00	01	02	03	04	09	0a	0b	20	21	22	23	28	29	2a	

In Figure 53, marked as 1, is the beacon broadcast of the ZC containing the network configuration (*BO*, *SO*), as seen in the *Packet Type* field. Note that the Time Delta (4150 ms) between beacons represents the beacon interval. The sequence of messages marked as 2 represents the association procedure. The ZR with the extended address of 0x0000000200000002 sends an association request to the ZC (0x0000). The ZC acknowledges the reception of the request and informs the ZR that there is pending data (using the pending data field in the acknowledge frame). Then, the ZR sends a data request command frame requesting the pending data. The ZC replies with the association response command frame containing the status of the association (that in this case is successful) and the ZR is assigned the short address 0x0001.



	Time Delta	Source	Destination	Packet Type
1	+00:00:04.150	0x0000	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:04.150	0x0000	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:00.084	0x0000000200000002	0x0000	Command: Association Request
	+00:00:00.008			Acknowledgment
2	+00:00:00.008	0x0000000200000002		Command: Data Request
	+00:00:00.008			Acknowledgment
	+00:00:00.002	0x0000000100000001	0x0000000200000002	Command: Association Response
	+00:00:00.009			Acknowledgment
	+00:00:04.050	0x0000	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:04.150	0x0000	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:04.150	0x0000	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:00.028	0x0001	0x0000	NWK Data
	+00:00:00.008			Acknowledgment
3	+00:00:00.095	0x0000	0x0001	NWK Data
	+00:00:00.008			Acknowledgment
	+00:00:00.355			Acknowledgment
4	+00:00:03.766	0x0000	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:00.293	0x0001	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:03.863	0x0000	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:00.292	0x0001	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:03.863	0x0000	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:00.292	0x0001	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	+00:00:03.863	0x0000	0xffff	Beacon: BO: 8, SO: 4, PC: 1, AP: 1

Figure 53 - Association and negotiation example

Now, the ZR is associated as a ZED and can therefore communicate in the network, but it still needs to request the ZC for a beacon broadcast transmission permit and a time window slot (transmission offset). The negotiation procedure is marked as 3. Until this point, and after the network association, the ZR behaves as a normal ZED. When the negotiation for beacon transmission finishes (successfully), the ZR starts to broadcast beacons in its assigned time window, as seen in Figure 53 marked as 4. Note that both the association and negotiation for beacon transmission took place during the ZC superframe.

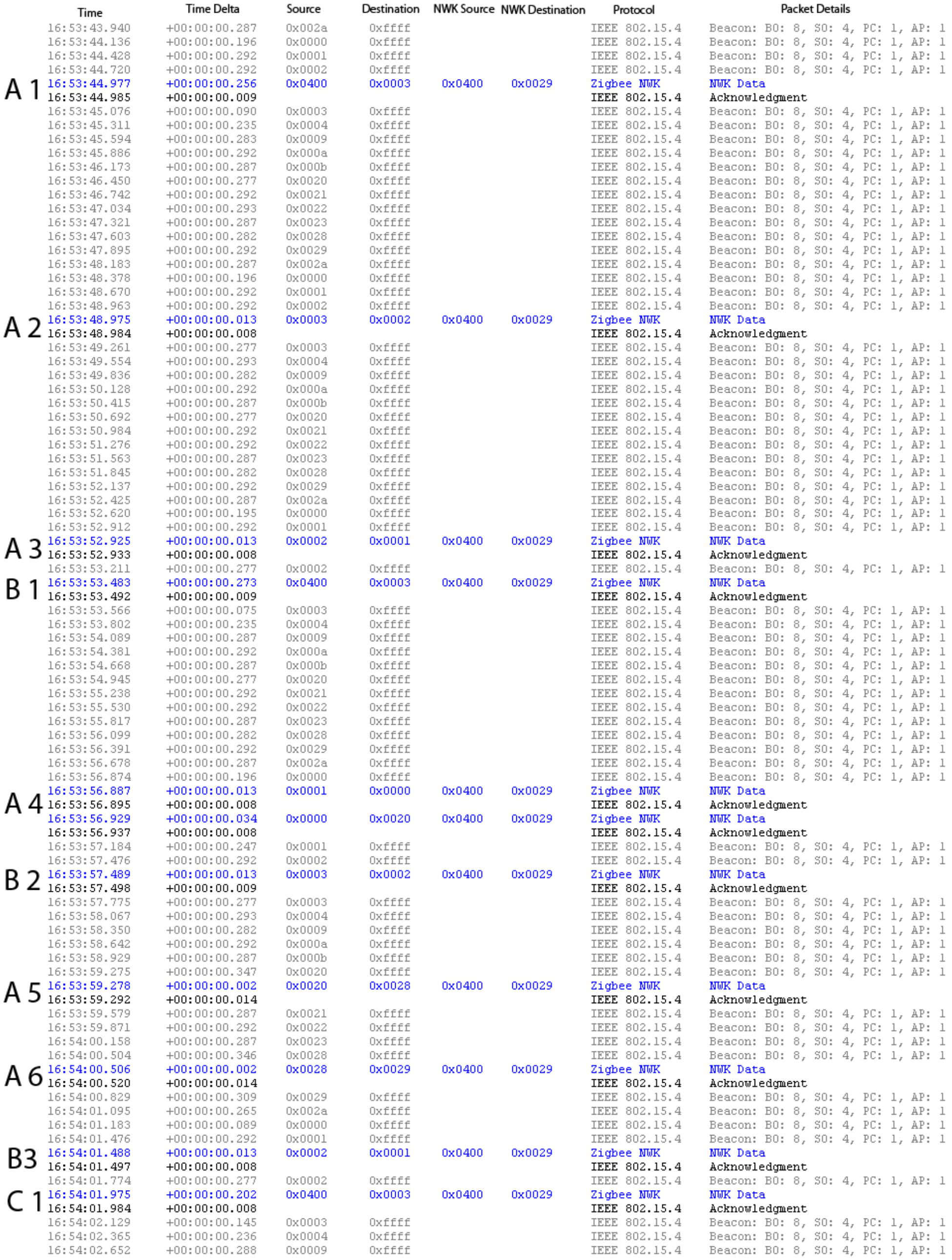
Figure 54 shows the negotiation packets decoded. In Figure 54.A the negotiation request (from ZR 0x0001 to ZC 0x0000) and in Figure 54.B is the negotiation accept (from ZC 0x0000 to ZR 0x0001). Highlighted is the data frame payload, in green (the first byte) is the negotiation type of message, in blue (the second and third bytes) the information of the beacon order and superframe order and in yellow (the fourth to sixth bytes) the beacon transmission offset value in symbols.

Frame 15 (Length = 27 bytes) Time Stamp: 14:53:34.863 Frame Length: 27 bytes Capture Length: 27 bytes Link Quality Indication: 136 Receive Power: -49 dBm IEEE 802.15.4 Frame Control: 0x8821 Sequence Number: 165 Destination PAN Identifier: 0x1234 Destination Address: 0x0000 Source PAN Identifier: 0x1234 Source Address: 0x0001 Frame Check Sequence: Correct ZigBee NWK Frame Control: 0x0004 Destination Address: 0x0000 Source Address: 0x0001 Radius = 1 Sequence Number = 97 NWK Payload: 01:08:04:00:00:00	Frame 17 (Length = 27 bytes) Time Stamp: 14:53:34.867 Frame Length: 27 bytes Capture Length: 27 bytes Link Quality Indication: 164 Receive Power: -42 dBm IEEE 802.15.4 Frame Control: 0x8821 Sequence Number: 34 Destination PAN Identifier: 0x1234 Destination Address: 0x0001 Source PAN Identifier: 0x1234 Source Address: 0x0000 Frame Check Sequence: Correct ZigBee NWK Frame Control: 0x0004 Destination Address: 0x0001 Source Address: 0x0000 Radius = 1 Sequence Number = 79 NWK Payload: 02:08:04:00:3c:00
A) Negotiation request	**B) Negotiation response**

Figure 54 - Negotiation mechanism packet decode example

The ZigBee End Device 0x0400 has associated with ZR 0x0003 with the purpose of periodically transmit data frames thought the cluster-tree in order to test the topology and the tree-routing mechanism. In Figure 55 the message flows are marked with capital characters (e.g. A, B, C) and the hop count with numbers (e.g. A1, A2, A3).

In Figure 55, marked as A1, the first transmission of the packet from the ZED (0x0400) to its parent (ZR 0x0003) is shown. Note that this transmission is carried out during ZR 0x0003 superframe. The routing of the data frame from ZR 0x0003 to its parent in the cluster-tree (ZR 0x0002) is marked as A2. The multi-hop continues with the routing of the frame from ZR 0x0002 to ZR 0x0001 (A3). In B1, a new message flow is initiated by the ZED (0x0400). Then, in A4, the message is relayed from ZR 0x0001 to ZC (0x0000) and to ZR 0x0020. This transmission sequence is carried out during the ZC superframe. The multi-hop continues in A5 between ZR 0x0020 and ZR 0x0028. The last hop is carried out in A6 with ZR 0x0028 relaying it to its final destination, ZR 0x0028.



	Time	Time Delta	Source	Destination	NWK Source	NWK Destination	Protocol	Packet Details
	16:53:43.940	+00:00:00.287	0x002a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:44.136	+00:00:00.196	0x0000	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:44.428	+00:00:00.292	0x0001	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:44.720	+00:00:00.292	0x0002	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 1	16:53:44.977	+00:00:00.256	0x0400	0x0003	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:44.985	+00:00:00.009					IEEE 802.15.4	Acknowledgment
	16:53:45.076	+00:00:00.090	0x0003	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:45.311	+00:00:00.235	0x0004	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:45.594	+00:00:00.283	0x0009	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:45.886	+00:00:00.292	0x000a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:46.173	+00:00:00.287	0x000b	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:46.450	+00:00:00.277	0x0020	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:46.742	+00:00:00.292	0x0021	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:47.034	+00:00:00.293	0x0022	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:47.321	+00:00:00.287	0x0023	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:47.603	+00:00:00.282	0x0028	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:47.895	+00:00:00.292	0x0029	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:48.183	+00:00:00.287	0x002a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:48.378	+00:00:00.196	0x0000	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:48.670	+00:00:00.292	0x0001	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:48.963	+00:00:00.292	0x0002	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 2	16:53:48.975	+00:00:00.013	0x0003	0x0002	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:48.984	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:53:49.261	+00:00:00.277	0x0003	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:49.554	+00:00:00.293	0x0004	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:49.836	+00:00:00.282	0x0009	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:50.128	+00:00:00.292	0x000a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:50.415	+00:00:00.287	0x000b	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:50.692	+00:00:00.277	0x0020	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:50.984	+00:00:00.292	0x0021	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:51.276	+00:00:00.292	0x0022	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:51.563	+00:00:00.287	0x0023	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:51.845	+00:00:00.282	0x0028	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:52.137	+00:00:00.292	0x0029	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:52.425	+00:00:00.287	0x002a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:52.620	+00:00:00.195	0x0000	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:52.912	+00:00:00.292	0x0001	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 3	16:53:52.925	+00:00:00.013	0x0002	0x0001	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:52.933	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:53:53.211	+00:00:00.277	0x0002	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
B 1	16:53:53.483	+00:00:00.273	0x0400	0x0003	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:53.492	+00:00:00.009					IEEE 802.15.4	Acknowledgment
	16:53:53.566	+00:00:00.075	0x0003	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:53.802	+00:00:00.235	0x0004	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:54.089	+00:00:00.287	0x0009	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:54.381	+00:00:00.292	0x000a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:54.668	+00:00:00.287	0x000b	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:54.945	+00:00:00.277	0x0020	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:55.238	+00:00:00.292	0x0021	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:55.530	+00:00:00.292	0x0022	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:55.817	+00:00:00.287	0x0023	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:56.099	+00:00:00.282	0x0028	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:56.391	+00:00:00.292	0x0029	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:56.678	+00:00:00.287	0x002a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:56.874	+00:00:00.196	0x0000	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 4	16:53:56.887	+00:00:00.013	0x0001	0x0000	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:56.895	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:53:56.929	+00:00:00.034	0x0000	0x0020	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:56.937	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:53:57.184	+00:00:00.247	0x0001	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:57.476	+00:00:00.292	0x0002	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
B 2	16:53:57.489	+00:00:00.013	0x0003	0x0002	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:57.498	+00:00:00.009					IEEE 802.15.4	Acknowledgment
	16:53:57.775	+00:00:00.277	0x0003	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:58.067	+00:00:00.293	0x0004	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:58.350	+00:00:00.282	0x0009	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:58.642	+00:00:00.292	0x000a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:58.929	+00:00:00.287	0x000b	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:59.275	+00:00:00.347	0x0020	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 5	16:53:59.278	+00:00:00.002	0x0020	0x0028	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:59.292	+00:00:00.014					IEEE 802.15.4	Acknowledgment
	16:53:59.579	+00:00:00.287	0x0021	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:59.871	+00:00:00.292	0x0022	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:00.158	+00:00:00.287	0x0023	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:00.504	+00:00:00.346	0x0028	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 6	16:54:00.506	+00:00:00.002	0x0028	0x0029	0x0400	0x0029	Zigbee NWK	NWK Data
	16:54:00.520	+00:00:00.014					IEEE 802.15.4	Acknowledgment
	16:54:00.829	+00:00:00.309	0x0029	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:01.095	+00:00:00.265	0x002a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:01.183	+00:00:00.089	0x0000	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:01.476	+00:00:00.292	0x0001	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
B3	16:54:01.488	+00:00:00.013	0x0002	0x0001	0x0400	0x0029	Zigbee NWK	NWK Data
	16:54:01.497	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:54:01.774	+00:00:00.277	0x0002	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
C 1	16:54:01.975	+00:00:00.202	0x0400	0x0003	0x0400	0x0029	Zigbee NWK	NWK Data
	16:54:01.984	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:54:02.129	+00:00:00.145	0x0003	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:02.365	+00:00:00.236	0x0004	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:02.652	+00:00:00.288	0x0009	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1

Figure 55 - Message Flow - Data Frame multi-hop thought the cluster-tree

4.8 Conclusions

The Time Division Beacon Scheduling mechanism was implemented successfully and enabled the engineering of a ZigBee Cluster-Tree network. This chapter demonstrated the feasibility of this mechanism enabling a real deployment of a ZigBee Cluster-Tree. This topology has several advantages such as the possibility to ensure end-to-end delays by allocating GTS. Comparing with the ZigBee mesh topology, the Cluster-Tree is more energy efficient as it allows inactive periods on all devices in the network. In mesh network topologies, the inactive periods are only possible in the ZEDs as all the ZRs must be active at all times. The results shown in this chapter are an important step towards the real use of the ZigBee Cluster-Tree topology.

The beacon only period proposed by Task Group 15.4b imposes major changes in the protocol, does not allow GTS allocations and it needs a complex scheduling mechanism. Another problem of the beacon only period is how to define the beacon transmission window size that can lead to scale limitations of this approach.

All the overviewed mechanism can be improved if the Coordinators positions are known. This way is possible to adjust the scheduling mechanisms in order to allow the sharing of time slots by Coordinators with non-overlapping transmission ranges. Nevertheless, this position based approach can be quite complex to implement.

Another important factor is that the available IEEE 802.15.4/ZigBee solutions (from different manufacturers) only implement a partial subset of the IEEE 802.15.4/ZigBee protocols, using the mesh network topologies in opposition to the Cluster-Tree. This trend is explained due to the complexity of implementing a synchronized topology as the cluster-tree. Moreover the standards do not specify how to effectively construct a synchronized cluster-tree topology. The commercialized mesh solutions enable this new ZigBee technology but we believe that it is worth while to explore the potentially of Cluster-Tree topologies for supporting scalable, energy-efficient and time-sensitive WSN applications.

Chapter 5

Tackling ZigBee Router Faults in ZigBee Cluster-Tree Networks

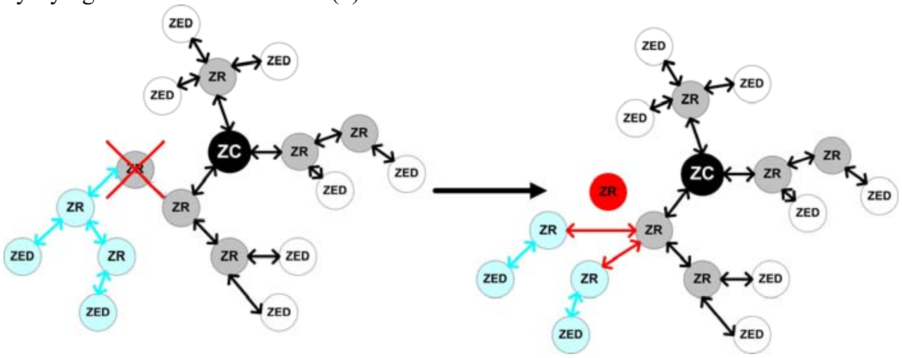
This chapter addresses fault-tolerance mechanisms for the specific case of ZigBee cluster-tree WSNs, which are prone the single point of failure problem in the ZigBee Routers (or cluster-heads). This chapter analyzes common problems associated to ZigBee Routers failures or link degradation, the default orphan realignment mechanism and presents a proactive fault-tolerance mechanism in order to overcome the referred situations. The chapter ends with some implementation guidelines for integrating these add-ons in ZigBee.

5.1 Introduction

Wireless Sensor Networks are inherently unpredictable and are very prone to failures. These failures can create blind spots in the network by isolating some of the devices or introduce large inaccessibility times in communications that can lead to abnormal behaviours of the applications. Furthermore, in case of large-scale WSNs, these failures can lead to the collapse of the entire network or a significant part of it. The current specification of the IEEE 802.15.4/ZigBee protocol stack lacks efficient fault-tolerance mechanisms for tackling ZigBee Router (ZR) failures. This is true since the basic orphan realignment mechanism imposes large inaccessibility times that may not be acceptable for certain time critical applications. This chapter presents a proactive fault-tolerance approach that was initially proposed in [30] that avoids link loss when there are alternative parents in the vicinity, based on a regular assessment of the parent-to-child link quality. If the quality of the link goes down a certain threshold, the child device anticipates the link degradation and chooses another parent device if available. The proactive mechanism uses a quality indicator (PAI - Parent Adoption Indicator) to choose a new parent, as described in Section 5.4.1. However, in order to avoid the *ping-pong* effect, a certain link quality hysteresis between the old and the new parent must be defined. This mechanism is backward compatible with the IEEE 802.15.4/ZigBee standards as it is only implemented in the ZEDs without any loss to the overall network [30].

5.2 Network Model and the Default Fault-Tolerance Mechanism

As already addressed in Chapter 2, in ZigBee Cluster-Tree topologies, one ZC identifies the entire network and each ZR assumes the role of cluster-head, allowing the association of other ZRs and ZEDs in a parent-child relationship. There can be multiple clusters in a network. When the association process is successful, the child device (ZED or ZR) has joined (or associated) the network through its parent (ZR). Inside a cluster, the communication is established via the cluster-head i.e. direct communication between two children in the same cluster is not possible. Figure 56 depicts a scenario of a cluster-tree network where a ZR failed (a), leaving all the devices within the clusters depending on the failing parent isolated from the network and unable to communicate. The orphan nodes must try to find alternatives in order to overcome the parent ZR failure by trying to associate to another (b).



a) b)

Figure 56 - ZigBee Cluster-Tree failure

IEEE 802.15.4/ZigBee supports a native fault-tolerance mechanism denominated as the orphaned device realignment. This recovery/repair procedure is triggered when there are repeated communication failures in the requests for data transmissions (e.g. data frames sent without receiving the requested acknowledgment) between the device and its parent or when the device loses synchronization with its parent. The MAC sub-layer defines the constant *aMaxLostBeacons* to specify the maximum allowed beacon frame losses and *aMaxFrameRetries* as the maximum number of retries after a transmission failure.

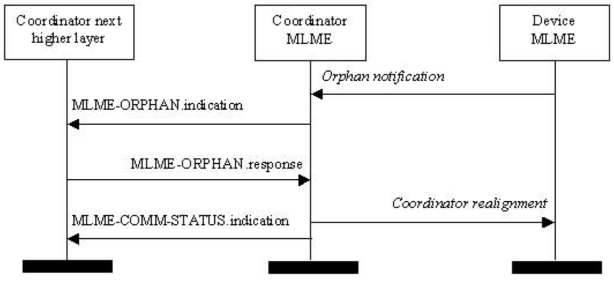


Figure 57 - Message sequence chart for orphan notification [4]

Upon the conclusion that the device is orphaned the MAC sub-layer can have two different behaviours. It can either perform the orphaned realignment procedure or reset the MAC parameters leading to a new association procedure to the network. The orphan realignment procedure relies on two command frames, the orphan notification frame (as depicted in Figure A.11 in Annex A), which is broadcast by the orphan device including its extended address, and the *Coordinator realignment* frame (Figure A.12 in Annex A) sent in response by the parent containing the information about the device (e.g. short address allocated) and about the network. The orphan association starts with an orphan scan procedure where the orphan device performs a physical channel scan on all available (or pre-defined) radio channels and sends orphan notifications, as depicted in Figure 57.

If the parent device receives the orphan notification command, it will reply with a *Coordinator realignment* frame, after a search in its neighbour table (this table contains information about the neighbours, including the associated devices) verifying if the command was sent by one of its child devices. The orphan device stops the channel scan procedure upon reception of the realignment frame, updating its PAN information. If the orphan device completes the channel scan without finding its parent, it must start a new association to the network. In the association mechanism, the device performs a channel scan searching for a suitable parent. After the synchronization with the new parent, the device starts the association procedures. During the time to scan the channels, synchronize with the chosen parent and associate with it, the device cannot transmit nor receive any messages. Figure 58 depicts a flow chart describing the orphan scan procedure.

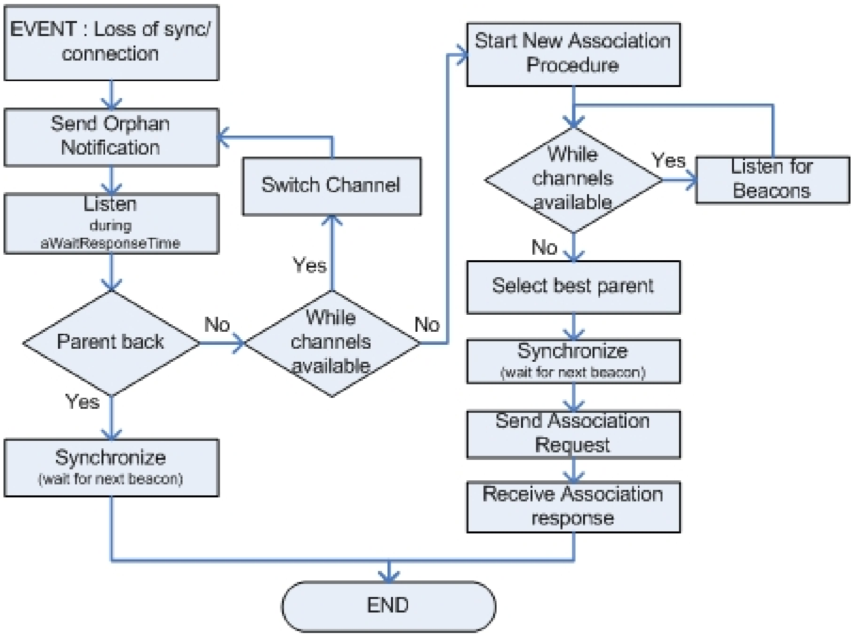

EVENT : Loss of sync/ connection
Send Orphan Notification
Listen during aWaitResponseTime
Parent back
Yes
No
Synchronize (wait for next beacon)
Switch Channel
Yes
While channels available
No
Start New Association Procedure
While channels available
Yes
Listen for Beacons
No
Select best parent
Synchronize (wait for next beacon)
Send Association Request
Receive Association response
END

Figure 58 - IEEE 802.15.4 orphan realignment mechanism flow chart

The default orphan scan realignment presents several limitations and some of them may jeopardize the integrity of the cluster-tree network due to collisions between the orphan realignment frames and beacon frames. Figure 59 depicts the collision problem as there are several collision scenarios or failed realignment attempts. In Figure 59 four situations are shown: the first (Figure 59-1) the orphan realignment attempt fails because it is sent during the inactive period of the lost parent; in the second (Figure 59-2) there is a collision with the beacon that will cause synchronization problems in other child devices and Coordinators depending on that beacon; the third scenario (Figure 59-3) is the only successful attempt, as the orphan realignment command is sent during the parents CAP; the fourth scenario (Figure 59-4) is another failed attempt due to collision with GTS data (that must be transmitted without contention) leading to problems with the nodes that required the guaranteed time slots for time-critical transmissions. These scenarios can also happen when the device switches to a new channel in the parent search.

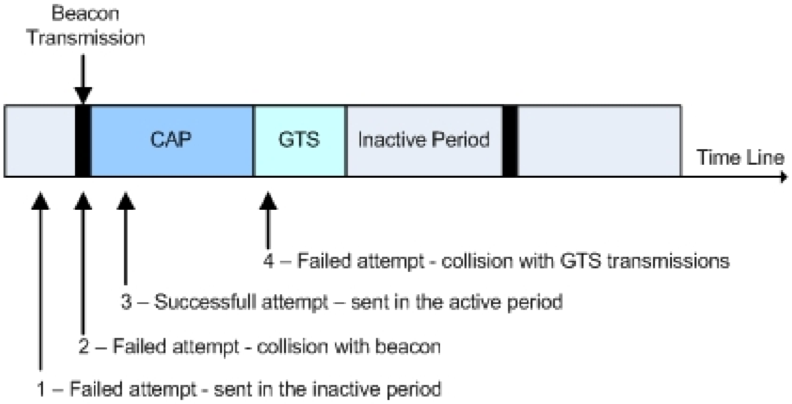

Beacon Transmission
CAP
GTS
Inactive Period
Time Line
4 – Failed attempt - collision with GTS transmissions
3 – Successfull attempt – sent in the active period
2 – Failed attempt - collision with beacon
1 – Failed attempt - sent in the inactive period

Figure 59 - Orphan realignment commands - collision problem

In sum, sending orphan realignment messages is not an efficient way to solve the orphaning problem; furthermore it can isolate network sectors due to collisions. The transmission of the orphan realignment must be improved in a way that it is only transmitted if the orphan device internal timers maintain their integrity, otherwise the device must not send anything and the mechanism should only rely on listening for beacons. Also, the mechanism can not operate with the transmission of orphan realignment command in a new channel, from whom the device does not have any information. Worst case, an orphan realignment transmission should only happen once and in the devices current channel.

Another problem with the default orphan mechanism is its timing behaviour, as the mechanism is time consuming in the search for the current or new parent. This problem is extended if the network is setup using many radio channels.

It is very important to improve the network responsiveness to failures i.e. when there is a communication failure, the reaction must be triggered fast. These considerations must be taken into account during the configuration of the network parameters. For example, the default value for the *aMaxLostBeacon* is 4, meaning that only after 4 consecutive beacon losses the device considers itself orphan, triggering the orphan realignment mechanism. This can introduce several problems in the network behaviour due to collisions because the device continues its normal transmissions during the 4 consecutive beacon losses.

5.3 Related Work

There are several reasons for a communication link or a device/node to fail. Fault-tolerance mechanisms tackle these abnormal situations. Generally there is a trade-off between the reliability improvement obtained by a fault-tolerance mechanism and the performance of the network.

There are several research works that analyze and propose fault-tolerance mechanisms for WSNs. Some have assessed the problem of fault tolerance in order to improve the reliability of a WSN network, minimize hardware and software failures and improve the topology changes. The fault tolerance problem has been addressed in classic wired networks and some proposals became popular such as the Spanning Tree Protocol [63] or IP Multicast [64].

In Reference [65], the authors present the limitations of WSN communications and their need for a certain level of reliability. They have also overview the different characteristics and design decisions to be considered when implementing a fault-tolerance system for WSNs. Finally, they depict the main characteristics of an ideal fault-tolerance system. In Reference [66], the authors present an overview of the main problems impairing reliability, review some existing fault-tolerance mechanisms and discuss some of the related open issues.

The evaluation of the real impact of a node failure in a WSN has been studied in [67]. The authors designed and developed a simulation tool based on a Bayesian model that shows the impact of a node failure on a mesh network. In addition, they assess the reliability improvement when using the hardware redundancy technique as a simple fault-tolerance system, noticing that hardware redundancy can increase the overall reliability of the network but not sufficiently enough to justify the cost of a global hardware redundancy. This latest work points out the need to fault-tolerance systems in the situation of vulnerable WSNs.

The authors in [68] proposed fault-tolerance mechanisms specifically for Cluster-Tree WSNs (different from the ZigBee cluster-tree model), based on a consensus approach. The protocol proposed in [68] evolves in two phases: a fault detection phase followed by a fault recovery phase. The main objective of the detection phase is to determine if there is really a fault in the network. For that reason, a consensus mechanism is used between the different gateways, if they have coherent information about a node failure, it is considered faulty and the recovery process begins. The gateways maintain the current state of other gateways by the means of “state updates”. When a failure is detected, the failure recovery procedure starts and the backup gateway that centralizes the information about the sensor nodes in vicinity, decides which sensors are to be attached to it.

Another protocol supporting a fault-tolerance mechanism has been proposed in [60]. LEACH is a re-clustering protocol that can be considered to be fault-tolerant, since it distributes the failure probabilities between all routers in the network, even though it was not designed for that specific purpose.

5.4 Proactive Re-Association Mechanism

5.4.1 The Parent Adoption Indicator (PAI)

The IEEE 802.14.5 protocol standard bases the selection criteria of a new parent on the Link Quality Indicator (LQI) measured at a given moment as a quality indicator for the potential new parent. However, this indicator cannot provide the child node with an accurate idea on the real performance of the potential parent. Thus, to provide the node with a more accurate knowledge, this section introduces the Parent Adoption Indicator (PAI) that does not only rely on the LQI measure but also on other important parameters that may affect the selection of the new parent in the cluster-tree namely: the depth (*Dp*) in the tree, the traffic load (T_l) and the energy indicator (*Ei*) of the candidate parent. The Parent Adoption Quality Indicator (PAI) is expressed as:

$$PAI = LQI \cdot (a \cdot Ei) \cdot \left(\frac{b}{Dp} \right) \cdot \left(\frac{c}{T_l} \right) \tag{5.1}$$

The coefficients *a*, *b* and *c* are integer weighting factors that can take different values depending on the importance given to each quality parameter. Since higher values of *Dp* and T_l parameters indicate less potential of the candidate parent, the computation of their inverse reflects the degradation they introduce. The most suitable parent will have the highest PAI value.

In terms of implementation, the PAI is an 8-bit value varying from 0 to 255.

Table 8 summarizes the different parameters that affect the PAI with their variation range and the resulting values considered.

Table 8 - Parent Adoption Indicator input values

Parameter Name	Bit range	Variation	Admissible values
LQI - Link Quality Indicator	0x00 – 0xff	0 – 255	0 – 255
E_i -Energy Indicator	0000 – 1100	Critical, 33%, 66%, 100%	0,2; 0,6; 0,8; 1
D_p - Parent Depth	0x00–*nwkcMaxDepth*	0 - *nwkcMaxDepth*	1 – *nwkcMaxDepth*
T_f - Transmit Failure	0x00 – 0xff	0 – 255	0 – 255

The PAI is used to access the potential parent's quality according to the following criteria: bad PAI, between 0 and 54, the parent is a bad alternative; normal PAI, between 55 and 200; the potential parent seems to be stable; and excellent PAI, between 200 and 255, the potential parent is in excellent condition.

5.4.2 Proactive Re-association Mechanism for ZigBee Cluster-Tree Networks

A proactive re-association approach (proposed in [30]) enables a preventive change of the parent in order to avoid the loss or extreme degradation of the current parent-child link. The proactive approach must guarantee that certain conditions are verified before switching to a new parent. The estimation of the parent's link degradation must be as accurate as possible because using weak switching conditions can cause child nodes to change parents too often without real need (ping-pong effect), thus inducing frequent inaccessibility periods and increased energy consumption.

The proactive mechanism parameters are initialized during the first execution of the mechanism, usually when turning *on* the node or during the deployment phase determining the behaviour of the mechanism. In fact, varying the model parameters has a significant influence on the overall behaviour of the mechanism; that is why the values of the different attributes must be chosen in a manner that optimizes the performance of the mechanism.

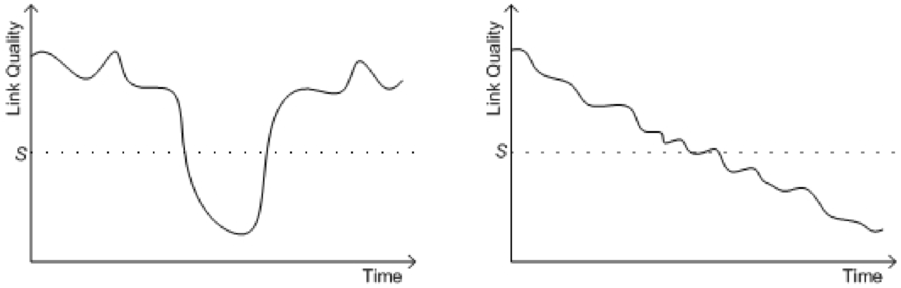
The proactive re-association mechanism requires five parameters: the samples period (*r*), the minimum parent quality threshold (*S*), the threshold adaptation factor (*Tu*), the number of confirmation packets (*CP*) and the parent compare hysteresis factor (*K*).

The sampling period (*r*) expresses how often the received messages from the current parent (i.e. data and beacon frames) are tested (by computing the PAI), thus assessing the parent's quality. The lower the value of *r*, the better is the child's knowledge of the parent-child link. The higher that value, the less frequently the parent-child link will be assessed, thus it could be more prone to deterioration without the child initiating the proactive procedure in order to avoid parent loss. Naturally there must be a trade-off between the sampling period and the resource consumption it introduces, namely energy and processing power.

The parent quality threshold (*S*) is the minimum value of the PAI indicator that ensures sufficient transmission conditions between the parent and child devices, thus it also defines the link quality level under which the child device triggers the proactive re-association mechanism. If the quality of the parent (PAI) is sufficiently good (PAI > *S*) there is no need to switch to a new parent even if there are potential parent devices that present better quality indicators.

The threshold adaptation factor (*Tu*) is used in the dynamic adaptation of the *S* value. It is a user defined percentage of deterioration that is still acceptable by the child device when it is connected to a new parent. The higher the *Tu* value, the lower will be the new *S* value, so less frequent will be the triggering of the mechanism. If the *Tu* value is low, the *S* value will be very close to the PAI of the actual parent.

The number of confirmation packets (*CP*) is the number of consecutive parent packets that are used to compute the PAI, in order to confirm that all of them are below the *S* threshold, avoiding unnecessary triggering of the mechanism in case of momentary link degradation (Figure 60.A) or be dynamically updated during the execution of the proactive mechanism using the threshold adaptation factor (*Tu*). The dynamic adaptation of this threshold can avoid unnecessary triggering of the proactive mechanism (ping-pong effect) by a better reflection of the current parent-child link conditions.



A) Temporary link degradation **B) Gradual link degradation**

Figure 60 - Link quality behaviours

The parent hysteresis factor (*K*) is introduced in the comparison between the old parent and the new one to avoid the ping-pong effect, that causes child devices to alternate too often between parents. This factor also ensures that the new parent has a sufficiently higher quality that the old one.

Figure 61 depicts the proactive re-association mechanism procedure.

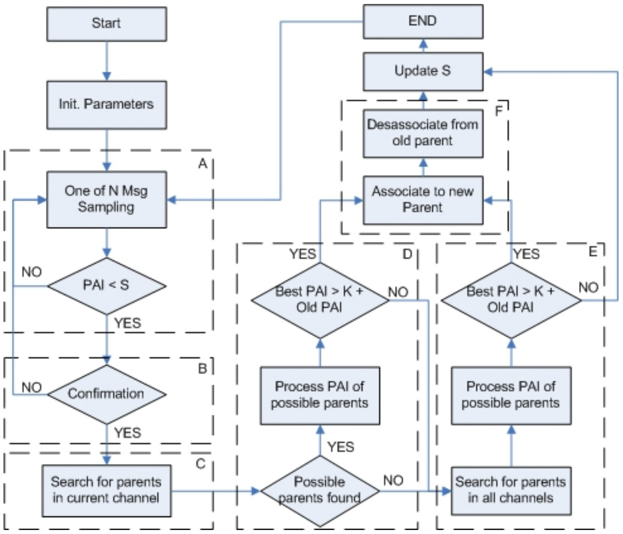


Figure 61 - The proactive re-association mechanism flow chart

During normal operation, a node samples one out of every *N* incoming messages (reflected by the sample period) in order to assess the quality of its current parent. If the PAI is under *S* (Figure 61.A), the child triggers the confirmation phase (Figure 61.B) during which processes the PAI for a given number of consecutive received packets. If all packets are below *S*, the mechanism is sure that the parent is degrading; otherwise it goes back to the normal sampling phase. If the child detects a parent failure, it scans its current channel (Figure 61.C) during its superframe inactive period looking for alternative parents within its own network (channel). If candidate parents are found within its network, the device calculates their PAI.

The candidate parent with the best PAI must also fulfil the following condition (Figure 61.D):

$$PAI_{new} > PAI_{old} + K \tag{5.2}$$

where *K* is the expected minimum quality improvement gained by changing parents. If such a parent device is found, the device associates to it and disassociate from its former parent. In this case, there is no inaccessibility time since the device is always connected to the PAN (Figure 61.F). If there is no device fulfilling the condition, the child device launches a channel scan on all the available pre-defined channels searching for possible parents during the inactive period of its superframe (Figure 61.E). After calculating the PAI of the potential parents found during the scan, the device searches for a candidate parent. If such ZR is found, the device associates to it and disassociates from its current parent (Figure 61.F), otherwise it stays connected to its current parent. The last step of the mechanism is the dynamic adaptation of the *S* threshold. If activated, the child device updates the value of the *S* threshold using the following formula:

$$S = PAI_{new} - Tu \tag{5.3}$$

where *Tu* is the deterioration factor expressed in percentage.

The proactive re-association approach introduces interesting advantages that improves energy balancing by ordering the candidate parents based on energy information, traffic load, number of associated nodes and link quality information. Ultimately, the proactive re-association mechanism leads to an establishment of connections that offer the best transmission conditions between all the nodes of the network. However, this mechanism does not eliminate blind spots in the networks when there is a complete failure of the parent node and there are no alternative parents in the vicinity.

5.5 Timing behaviour

5.5.1 General assumptions

This section presents an evaluation of the performance, in terms of inaccessibility time for the proactive mechanism and the default IEEE 802.15.4 orphan realignment mechanism, in order to give a practical intuition on the improvement and behaviour of the mechanisms. The inaccessibility time is defined as the duration starting from the instant when the fault occurs until it is completely recovered and the node resumes its normal network operation. The total inaccessibility time can be expressed as the sum of the following time components:

(1) Time to detect of the link failure (T_{loss}) with the parent. The device detects that it is orphan only after a pre-defined number of consecutive beacon losses, defined in the *aMaxLostBeacons* constant;

(2) Duration of the channel scan procedure (T_{scan}) where the device scans for the current or for a new parent. The scan phase consists in switching to all channels in the pre-defined logical channel list (LCS) that are currently being used. If the device is performing an orphan scan it sends a realignment command in each channel, waiting for a reply during *aWaitResponseTime*, being the total time calculated by:

$$LCS \times aWait\,Re\,sponseTime \tag{5.4}$$

where *aWaitResponseTime* = 32 x *aBaseSuperframeDuration*

Considering that the device finds its current (old) parent, the worst case scenario is considered when the device finds its current parent in the last scanned channel. The scan procedure for a new parent (when the device first enters the network or when the device does not find its current parent) consists in scanning each channel during a pre-defined time. The duration on each channel depends on the *ScanDuration* argument in the *MLME-SCAN.request* primitive (used to start the channel scan). The *ScanDuration* can assume the integer value of *n* varying from 0 to 14. In this analysis this value is considered equal to the device desired (in the case of a new association) or the current beacon order (*BO*) allowing the sufficient time to capture a beacon. Then the channel scan duration (T_{c_scan}) is calculated as follows:

$$Tc\_scan = aBaseSuperframeDuration \times \left(2^{n} + 1\right) \tag{5.5}$$

where *aBaseSuperframeDuration* = *aBaseSlotDuration* x *aNumSuperframeSlots*
aBaseSlotDuration = 60 symbols

(3) The synchronization time (T_{sync}) with the new or with the current parent by receiving its beacon. In the worst case scenario, the synchronization procedure can last for an entire beacon interval (BI);

(4) The association time (T_{asso}) can be calculated by the time span from the moment the child issues an association request until it receives the parent association response. In this analysis this time is considered negligible, taking into account the duration of a channel scan.

The total inaccessibility time (T_{inacc}) can then be expressed by the following equation:

$$T_{inacc} = T_{loss} + T_{scan} + T_{sync} + T_{asso} \tag{5.6}$$

In this analysis, the latency of the CSMA/CA medium access and the latency imposed by layers transfers and internal procedures are also considered negligible.

5.5.2 IEEE 802.15.4 orphan realignment procedure

In the IEEE 802.15.4 orphan scan mechanism, two different situations must be considered: (1) when the orphan device finds its current (old) parent during the orphan scan procedure and realigns with it; or (2) when de orphan device does not find its current (old) parent during the orphan scan procedure and has to perform a new scan to find a new parent. In the first case, the inaccessibility time assuming that the parent is found in the last channel of the logical channel list (LCS) can be computed according to Eq. 5.6 as follows:

$$T_{inacc-std^1} = aMaxLostBeacons \times BI + LCS \times aWait\,Re\,sponseTime + BI \qquad (5.7)$$

The inaccessibility time for the second case can be roughly formulated as follows:

$$T_{inacc-std^2} = aMaxLostBeacons \times BI + LCS \times aWaitResponseTime + LCS \times csd + BI \qquad (5.8)$$

The inaccessibility time increases considerable when more radio channels are used in the same network.

5.5.3 Proactive re-association mechanism

The network inaccessibility times in the proactive re-association mechanism can be considered 0, because the mechanism is triggered by the consecutive reception of bad packets and not a link failure. Also there is no time during the scan procedures because the mechanism scans the possible parents during the inactive period. Even during the association to a new parent the child device is still associated to its current (old) one. Network connectivity is always preserved, meaning that:

$$T_{inacc\_proactive} = 0$$

This value of IT is only valid in upstream communications, as downstream communications will still experience delays since that after the child device associated to a new parent, the network will have to update the routing tables, addressing information and other parameters relative to the new situation of the node.

5.5.4 Time comparison

To compare the different mechanisms timing behaviour of the native and proactive mechanism, the parameters in Table 9 were assumed, as suggested in the IEEE 802.15.4 standard. The channel scan duration is calculated using Eq. 5.5 where *n* is assumed to be the beacon order (*BO*).

Table 9 - Timing Comparison parameters

Parameter name	Value (symbols)
aBaseSlotDuration	60
aNumSuperframeSlots	16
aBaseSuperframeDuration	960
aResponseWaitTime	1920
aMaxLostBeacons	4
LCS	6

This scenario considers that 1 symbol corresponds to 17,362 µs (real value for the MICAz motes).

The graph in Figure 62 depicts the obtained values for each mechanism.

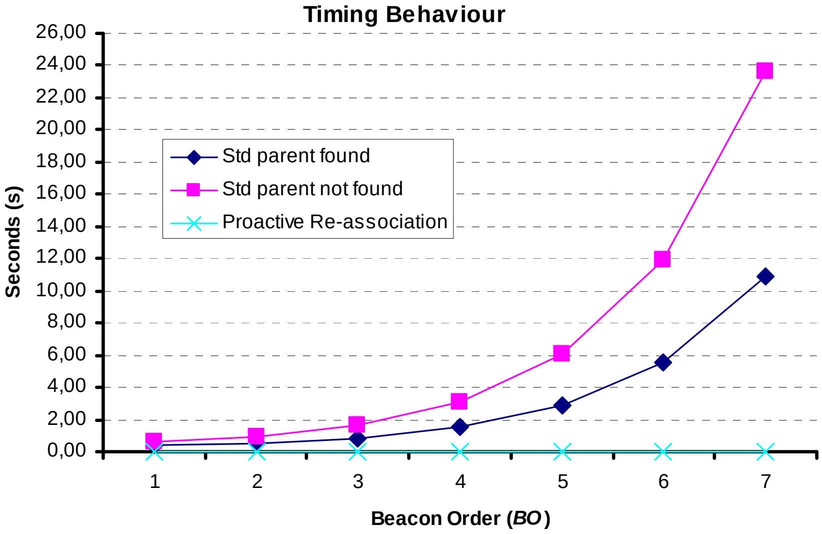


Figure 62 - Timing behaviour comparison graph

In Figure 62, it is possible to observe that network inaccessibility times of the standard mechanism increase with the beacon order. Also, when the current (old) parent is not found, there is a large inaccessibility time due to the need to find a new parent.

Considering that above mentioned problems with the orphan realignment mechanism there is a higher possibility that the device cannot accomplish the realignment with its parent and needs to search for a new one. Consider the case of *BO* = 7. When there is a fault and the current parent is not found the standard mechanism takes approximately 24 seconds to recover and associate with a new parent, while if the parent is found it only takes approximately 11 seconds. When the *BO* = 7 the beacon interval is about 2 seconds and the 13 seconds of different if the parent found or not is quite considerable, mainly in time critical applications.

For the proactive approach, the inaccessibility time is 0, just requiring some minor add-ons to the protocol and keeping backward compatibility with legacy nodes.

5.6 Implementation Guidelines for the Proactive Mechanism

The implementation of the proposed proactive fault-tolerance mechanism and its integration as a module in Open-ZB [24] is a work in progress.

Figure 63 depicts the implementation architecture of the current Open-ZB stack and the localization of the fault-tolerance module. Also the used IEEE 802.15.4 service access points are highlighted. The implementation of the proposed approaches only introduces minor add-ons to the protocol stack. Furthermore, total backward compatibility is assured, thus enabling the coexistence of both devices that do and not implement these add-ons.

The proactive re-association mechanism is implemented in the Network Layer and uses the standard defined interfaces to communicate with the adjacent layers. In addition, it does not affect the operation of the MAC sub-layer. The proactive mechanism effectively starts after the confirmation phase is validated (as described in section 5.4.2).

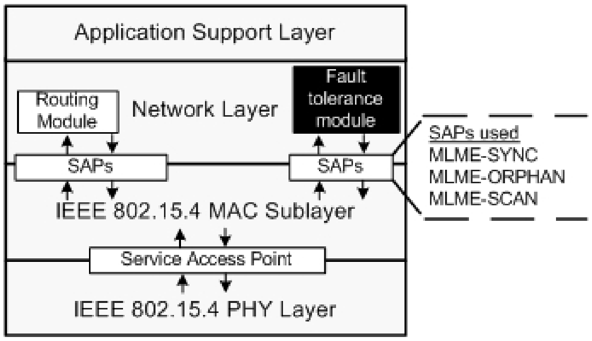


Figure 63 - Fault tolerance implementation architecture

The child device calls a passive scan procedure during its superframe inactive period using the standard MAC Service Access Points (SAPs) *MLME-SCAN.request*. The child device sets the *ScanChannels* parameter to the number of channels supported by the PHY layer, in order to go through all the defined channels. It ensures that the scan duration lasts long enough to detect all potential parents during the device's inactive period. As soon as it gets all the information regarding the potential parents, it processes the associated PAIs and stores that information in a field in the *nwkNeighborTable* created for that purpose. If a suitable new parent is found, the child device sends a *MLME-ASSOCIATE.Request* to the MAC sub-layer requesting the association to the new elected parent. After a successful association, the device sends a disassociation command to the current (old) parent.

5.7 Conclusions

This chapter presented a proactive re-association mechanism for handling router link degradation or failure in ZigBee cluster-tree networks. The proactive approach has the advantage to avoid the device orphaning procedures by planning in advance its re-association to a more reliable parent. Also, network inaccessibility times are eliminated and the overall ZigBee Cluster-Tree network reliability is improved. The proactive mechanism offers interesting potentiality for supporting mobility. Ultimately, the proactive re-association mechanism leads to an establishment of connections that offer the best transmission conditions between all nodes in the network. However, this mechanism does not eliminate blind spots, when there is a complete failure of the parent node and there are no alternative parents in the vicinity.

Chapter 6

On the IEEE 802.15.4 GTS Mechanism

This chapter discusses the potential under-utilization of the IEEE 802.15.4 of the Guaranteed Time Slot (GTS) mechanism. Since the allocation of GTS is limited to the maximum of seven (explicit) GTS allocations per ZigBee Router/Cluster, the available resources can rapidly disappear and, moreover, they can be underutilized resulting in wasted bandwidth. This chapter focuses on the implementation test and validation of an implicit GTS allocation mechanism (i-GAME) that improves on the native explicit mechanism by allowing more than one node share the same GTS.

6.1 Introduction

Timeliness guarantee is an important feature of the IEEE 802.15.4 protocol, turning it quite appealing for Wireless Sensor Network (WSN) applications under timing constraints. In fact, when operating in beacon-enabled mode, this protocol allows nodes with real-time requirements to allocate Guaranteed Time Slots (GTSs), in the contention-free period. However, the protocol natively supports explicit GTS allocation, i.e. a node allocates a number of time slots in each superframe for exclusive use. The limitation of this explicit GTS allocation is that GTS resources may quickly disappear, since a maximum of seven GTSs can be allocated in each superframe, preventing other nodes to benefit from guaranteed service. Moreover, the GTS may be underutilized, resulting in wasted bandwidth.

The i-GAME [31,69], an implicit GTS Allocation Mechanism overcomes these limitations. The i-GAME approach enables the use of one GTS by multiple nodes, still guaranteeing that all their (delay, bandwidth) requirements are satisfied. For that purpose, the i-GAME proposes an admission control algorithm that enables to decide whether to accept a new GTS allocation request or not, based not only on the remaining time slots, but also on the traffic specifications of the flows, their delay requirements and the available bandwidth resources.

The IEEE 802.15.4 GTS mechanism provides a minimum service guarantee for the corresponding nodes and enables the prediction of the worst-case performance for each node's application. However, the GTS mechanism presents some limitations in terms of efficiency and deployment in WSNs with a large number of nodes. In fact, during each superframe (divided into sixteen time slots) only up to seven GTSs can be allocated, forming the Contention-Free Period (CFP). The remaining time slots in the superframe compose the Contention Access Period (CAP) using CSMA/CA.

Since each GTS is exclusively assigned to one node, the number of nodes involved in the CFP is limited to seven or less. This is because the IEEE 802.15.4 standard assumes that a node performs an explicit GTS allocation request by asking the Coordinator for a certain number of time slots. A node is allowed to transmit during the CFP, if the number of available time slots in the superframe is higher than requested, and the minimum CAP length is not violated due to that allocation.

Two negative impacts may result from this explicit allocation scheme: (1) the GTSs can be quickly consumed by a few number of nodes, preventing the others from having a guaranteed service; (2) a node with a low arrival rate that has allocated a GTS, may use it partially (when the amount of guaranteed bandwidth is higher than its arrival rate). This leads to underutilization of the GTS bandwidth resources. Due to the pre-fixed time slot duration in a superframe, it is practically impossible to balance the arrival rate of a node and its guaranteed GTS bandwidth. The amount of wasted bandwidth increases with the variance between the guaranteed bandwidth and the arrival rate. Note that this wasted bandwidth could be used by the CAP.

6.2 The Implicit GTS Allocation Mechanism (i-GAME)

The i-GAME mechanism consists in each GTS to be shared between multiple nodes, instead of being exclusively dedicated to one node, if a certain schedule that satisfies the requirements of all requesting nodes exists. Sharing a GTS by several nodes means that the time slots of this GTS are dynamically allocated to different nodes in each superframe, according to a given schedule. In contrast, the native IEEE 802.15.4 explicit allocation statically devotes a GTS to only one node in all subsequent superframes.

The i-GAME admission control and scheduling [31,69](Annex E) mechanism is based on the traffic specification of the requesting nodes, their delay requirements, and the available GTS resources. Instead of asking for a fixed number of time slots, a node that wants to have a guaranteed service sends its traffic specification and delay requirement to its Coordinator (ZC or ZRs). Then, the latter runs an admission control algorithm based on this information and the amount of available GTS resources. The new allocation request will be accepted if there is a schedule that satisfies its requirements and those of all other previously accepted allocation requests; otherwise, the new allocation request is rejected.

The i-GAME algorithm manages the time slots used for the implicit GTS allocation based on the result of the admission control algorithm. This function stores the number of implicit flows allocated and the number of time slots currently used. On the reception of a new implicit allocation, the management algorithm processes the admission control algorithm for the received flow specifications to verify its acceptance and, if necessary, updates the number of allocated GTS time slots.

When creating the beacon, each Coordinator builds the descriptors distributing the allocated GTS in the available time slots, using a round robin policy.

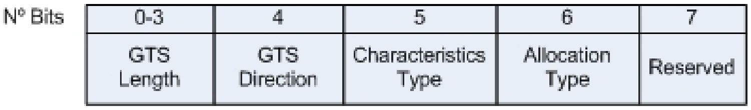
6.3 Implementation Details of the i-GAME Mechanism

This section presents some practical considerations for the implementation of the i-GAME mechanism in IEEE 802.15.4. An interesting feature of i-GAME is that its implementation only requires minor add-ons to the standard protocol. A detailed standard-like description of the interfaces added to the Network layer and the enhancements to the MAC sub-layer for supporting the i-GAME mechanism is presented in [32].

The i-GAME mechanism is implemented in the MAC sub-layer and in the Network Layers by defining a new service access point (SAP) between these two layers, the *MLME-iGAME*. The i-GAME SAP provides 4 primitives:

- *MLME-iGAME.request* primitive - allows a device to send a request to the PAN Coordinator to allocate a new implicit GTS.
- *MLME-iGAME.confirm* primitive - reports the result of a request to allocate a new implicit GTS.
- *MLME-iGAME.indication* primitive - indicates that an implicit GTS was requested.
- *MLME-iGAME.response* primitive - used to initiate a response to a *MLME-iGAME.indication* primitive with the details of a new implicit request.

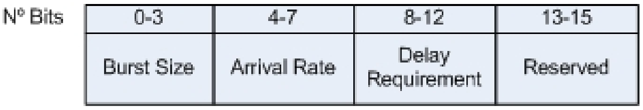
The idea consists in using the reserved 6th bit in the GTS characteristics frame, embedded in a GTS allocation request command field (Figure 64). This bit is referred to as *Allocation Type*.



Nº Bits	0-3	4	5	6	7
	GTS Length	GTS Direction	Characteristics Type	Allocation Type	Reserved

Figure 64 - i-GAME GTS Characteristics extension field format

The *Allocation Type* bit set to 0 corresponds to an explicit GTS allocation. In this case, the allocation process will follow the IEEE 802.15.4 standard. If it is set to 1, it refers to the i-GAME implicit allocation mechanism. In this case, to keep the IEEE 802.15.4 with no changes, the flow specification information (burst size, arrival rate and delay requirement) must be embedded in the higher layer packets, as presented in Figure 65.



Nº Bits	0-3	4-7	8-12	13-15
	Burst Size	Arrival Rate	Delay Requirement	Reserved

Figure 65 - i-GAME Flow Specification Field Format

The admission control algorithm is implemented in the Network Layer and returns the decision to the MAC sub-layer. Figure 66 depicts the implementation architecture of the i-GAME mechanism.

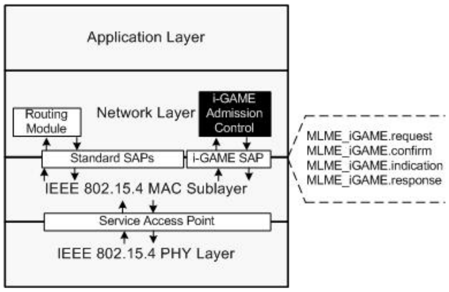


Figure 66 - i-GAME implementation architecture

Hence, upon reception of an implicit GTS allocation request (*Allocation Type* = 1), the MAC sub-layer of the Coordinator forwards the *Flow Specification* field (Figure 65) to the higher layer for processing by the admission control module. The *Burst Size* and the *Arrival Rate* fields should be expressed by four bits each. The *Delay Requirement* field is expressed by five bits. Using this frame format, the Coordinator should define a fixed range for each value (or class) of the corresponding field. These patterns should be known in advance by all nodes associated to the network before initiating an implicit allocation.

When the flow specification is received by the admission control module, it evaluates the admission of the new flow. The decision should be notified to the MAC sub-layer through the corresponding service access point. In case of acceptance, the MAC sub-layer allocates the time slots in the CFP in round robin order to all accepted nodes. For that purpose, the MAC sub-layer establishes a certain order to allocate the time slots according to round robin scheduling. Each beacon frame must indicate which nodes are allowed to use the GTS in the current superframe, according to the established order.

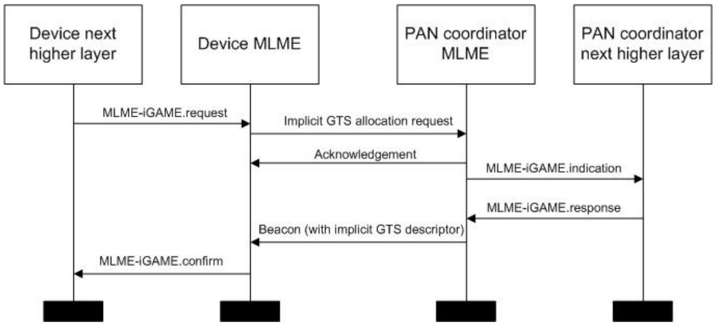


Figure 67 - Implicit GTS allocation (initiated by a device)

Figure 67 and Figure 68 illustrate the sequence of messages necessary for successful implicit GTS management. The first demonstrates the message flow for the case in which the device initiates the implicit GTS allocation while the second shows the message flow for the two cases in which an implicit GTS deallocation occurs, first, by a device (A) and, second, by the Coordinator (B).

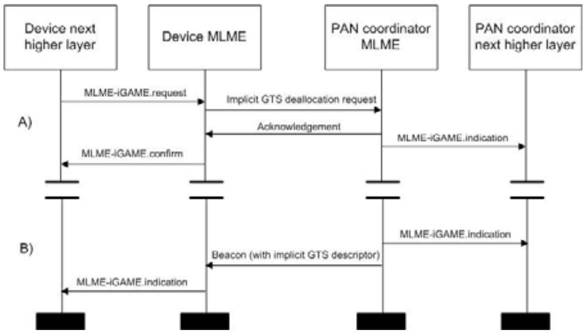


Figure 68 - GTS deallocation initiated by a device (A) and the Coordinator (B)

6.4 Experimental Testbed

The experimental testbed is based on an IEEE 802.15.4 star network operating in a beacon-enabled mode, with one PAN Coordinator and 7 nodes (Figure 69). The superframe structure is configured with a Beacon Order of 3 (*BO* = 3) and a Superframe Order of 3 (*SO* = 3). Each node runs an application that generates periodic traffic with a period of 200ms (P = 120) and a frame size of 120 bits (L = 120, 104 bits of MAC header + 16 bits of data payload). The arrival curve $\alpha(t) = b+r.t$ corresponding to this periodic data flow is defined by the average rate r = L/P = 0.6 kbps and the burst size b = L = 120 bits [31,69]. In this scenario, the bandwidth guaranteed by one time slot is experimentally evaluated, which was found to be equal to 2.70 kbps. This small value is due to the acknowledgment overhead in each data frame transmission.

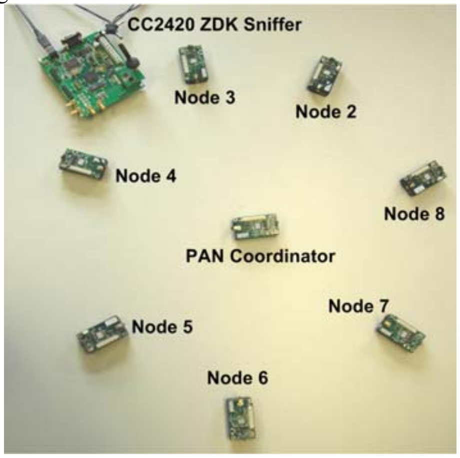


Figure 69 - Experimental testbed: 8 MICAz motes plus one PAN Coordinator

In this experiment, 5 classes were defined to map the burst size and the average rate, and 5 classes to map the delay bound. These values were mapped in the PAN Coordinator and in the nodes, as presented in Table 10.

Table 10 - Classes of Traffic Flow parameters

Class	Burst Size (bits)	Arrival Rate (kbps)	Delay Bound (ms)
0x0	80	0.6	300
0x1	120	1.2	500
0x2	160	2.4	700
0x3	200	4.8	900
default	1016	9.6	2000

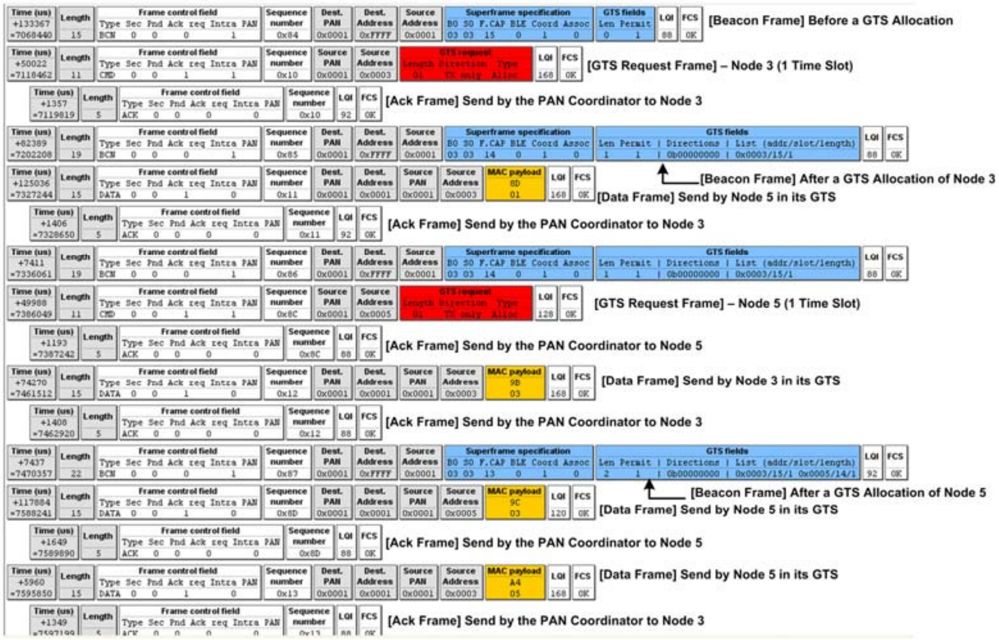
Hence, based on the specification of the Flow Specification frame format (Figure 65), a node using the i-GAME mechanism for requesting an implicit GTS allocation must configure its request by choosing the corresponding class. Thus, assuming for instance, a delay bound of 300 ms, nodes that send implicit GTS allocations configure their *Flow Specification Field* to the value (0x1, 0x0, 0x0), which is interpreted by the PAN Coordinator. Note that the configuration mapping table will depend on the application requirements and should be adapted either manually by the designer or automatically according to a user-defined specification. Nevertheless, note that this table applies to all nodes in the network.

The next section reports the observations and the results of explicit and implicit GTS allocations corresponding to this network scenario. The Chipcon IEEE 802.15.4 Packet Analiser [41] to intercept and visualize network traffic. The address of the PAN Coordinator is set to 0x0001 and the other addresses are set as shown in Figure 69.

6.5 Experimental Results

This section presents experimental results. It first starts by analysing the standard IEEE 802.15.4 GTS allocation - the explicit allocation. Then, the i-GAME implicit allocation mechanism is analysed. This section concludes with a comparison between the two mechanisms.

Figure 70 shows a message sequence chart of two explicit GTS allocations made by two nodes. In this example, observe that one time slot per node has been allocated. We run the network for a significant time span to compute the maximum delay experienced by the flows allocating GTSs. The maximum delay is embedded and printed in the MAC payload of each data packet, as seen in Figure 70. Note that the delays observed in this example do not correspond to the final measured maximum delays since it only presents the first packet that has been sent. The effective experimental maximum delay bound that we have measured is equal to 124.59 ms, which is very close to the theoretical delay bound of 125.51 ms, when applying the delay bound analysis of an explicit GTS allocation [35]. The accuracy of the theoretical bound is mainly a result of the good approximation of the arrival curve of the periodic traffic and the stair service curve of the GTS allocation mechanism.

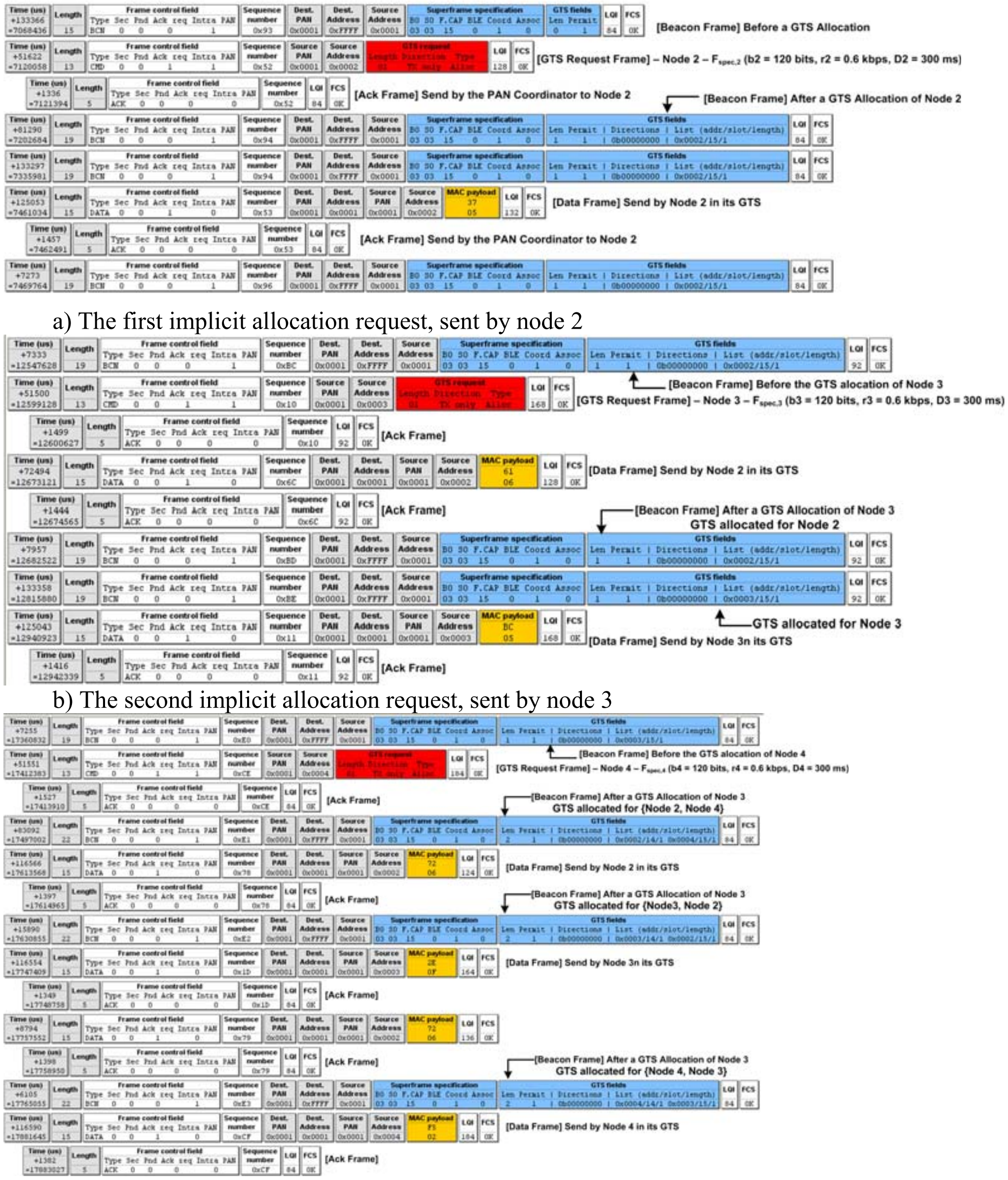


```
Time (us)  Length  Frame control field                Sequence  Dest.   Dest.    Source   Superframe specification         GTS fields   LQI  FCS
+133367            Type Sec Pnd Ack req Intra PAN     number    PAN     Address  Address  BO SO F.CAP BLE Coord Assoc      Len Permit
=7068440   15      BCN   0   0     0         1        0x84      0x0001  0xFFFF   0x0001   03 03  15    0    1     0        0     1     88   OK
[Beacon Frame] Before a GTS Allocation

Time (us)  Length  Frame control field                Sequence  Source  Source   GTS request                    LQI  FCS
+50022             Type Sec Pnd Ack req Intra PAN     number    PAN     Address  Length Direction Type
=7118462   11      CMD   0   0     1         1        0x10      0x0001  0x0003   01     TX only   Alloc          168  OK
[GTS Request Frame] – Node 3 (1 Time Slot)

Time (us)  Length  Frame control field                Sequence  LQI  FCS
+1357              Type Sec Pnd Ack req Intra PAN     number
=7119819   5       ACK   0   0     0         0        0x10      92   OK
[Ack Frame] Send by the PAN Coordinator to Node 3

Time (us)  Length  Frame control field                Sequence  Dest.   Dest.    Source   Superframe specification    GTS fields                                                LQI  FCS
+82389             Type Sec Pnd Ack req Intra PAN     number    PAN     Address  Address  BO SO F.CAP BLE Coord Assoc Len Permit | Directions | List (addr/slot/length)
=7202208   19      BCN   0   0     0         1        0x85      0x0001  0xFFFF   0x0001   03 03  14    0    1     0   1     1    | 0b00000000 | 0x0003/15/1                88   OK
[Beacon Frame] After a GTS Allocation of Node 3

Time (us)  Length  Frame control field                Sequence  Dest.   Dest.    Source  Source   MAC payload  LQI  FCS
+125036            Type Sec Pnd Ack req Intra PAN     number    PAN     Address  PAN     Address  8D
=7327244   15      DATA  0   0     1         0        0x11      0x0001  0x0001   0x0001  0x0003   01           168  OK
[Data Frame] Send by Node 5 in its GTS

Time (us)  Length  Frame control field                Sequence  LQI  FCS
+1406              Type Sec Pnd Ack req Intra PAN     number
=7328650   5       ACK   0   0     0         0        0x11      92   OK
[Ack Frame] Send by the PAN Coordinator to Node 3

Time (us)  Length  Frame control field                Sequence  Dest.   Dest.    Source   Superframe specification    GTS fields                                                LQI  FCS
+7411              Type Sec Pnd Ack req Intra PAN     number    PAN     Address  Address  BO SO F.CAP BLE Coord Assoc Len Permit | Directions | List (addr/slot/length)
=7336061   19      BCN   0   0     0         1        0x86      0x0001  0xFFFF   0x0001   03 03  14    0    1     0   1     1    | 0b00000000 | 0x0003/15/1                88   OK

Time (us)  Length  Frame control field                Sequence  Source  Source   GTS request                    LQI  FCS
+49988             Type Sec Pnd Ack req Intra PAN     number    PAN     Address  Length Direction Type
=7386049   11      CMD   0   0     1         1        0x8C      0x0001  0x0005   01     TX only   Alloc          128  OK
[GTS Request Frame] – Node 5 (1 Time Slot)

Time (us)  Length  Frame control field                Sequence  LQI  FCS
+1193              Type Sec Pnd Ack req Intra PAN     number
=7387242   5       ACK   0   0     0         0        0x8C      88   OK
[Ack Frame] Send by the PAN Coordinator to Node 5

Time (us)  Length  Frame control field                Sequence  Dest.   Dest.    Source  Source   MAC payload  LQI  FCS
+74270             Type Sec Pnd Ack req Intra PAN     number    PAN     Address  PAN     Address  9B
=7461512   15      DATA  0   0     1         0        0x12      0x0001  0x0001   0x0001  0x0003   03           168  OK
[Data Frame] Send by Node 3 in its GTS

Time (us)  Length  Frame control field                Sequence  LQI  FCS
+1408              Type Sec Pnd Ack req Intra PAN     number
=7462920   5       ACK   0   0     0         0        0x12      88   OK
[Ack Frame] Send by the PAN Coordinator to Node 3

Time (us)  Length  Frame control field                Sequence  Dest.   Dest.    Source   Superframe specification    GTS fields                                                LQI  FCS
+7437              Type Sec Pnd Ack req Intra PAN     number    PAN     Address  Address  BO SO F.CAP BLE Coord Assoc Len Permit | Directions | List (addr/slot/length)
=7470357   22      BCN   0   0     0         1        0x87      0x0001  0xFFFF   0x0001   03 03  13    0    1     0   2     1    | 0b00000000 | 0x0003/15/1 0x0005/14/1    92   OK
[Beacon Frame] After a GTS Allocation of Node 5

Time (us)  Length  Frame control field                Sequence  Dest.   Dest.    Source  Source   MAC payload  LQI  FCS
+117884            Type Sec Pnd Ack req Intra PAN     number    PAN     Address  PAN     Address  9C
=7588241   15      DATA  0   0     1         0        0x8D      0x0001  0x0001   0x0001  0x0005   03           120  OK
[Data Frame] Send by Node 5 in its GTS

Time (us)  Length  Frame control field                Sequence  LQI  FCS
+1649              Type Sec Pnd Ack req Intra PAN     number
=7589890   5       ACK   0   0     0         0        0x8D      88   OK
[Ack Frame] Send by the PAN Coordinator to Node 5

Time (us)  Length  Frame control field                Sequence  Dest.   Dest.    Source  Source   MAC payload  LQI  FCS
+5960              Type Sec Pnd Ack req Intra PAN     number    PAN     Address  PAN     Address  A4
=7595850   15      DATA  0   0     1         0        0x13      0x0001  0x0001   0x0001  0x0003   05           168  OK
[Data Frame] Send by Node 5 in its GTS

Time (us)  Length  Frame control field                Sequence  LQI  FCS
+1349              Type Sec Pnd Ack req Intra PAN     number
=7597199   5       ACK   0   0     0         0        0x13      88   OK
[Ack Frame] Send by the PAN Coordinator to Node 3
```

Figure 70 - A scenario of an explicit GTS allocation of two nodes

Figure 71 presents the sequence of implicit GTS allocations using the i-GAME mechanism. In Figure 71.a, a first request made by node 2 is shown. With one flow there is no difference to the explicit allocation (only in the frame format sent by the node). From Figure 70 and Figure 71, is possible to observe that the length of an implicit GTS allocation frame (13 bytes, in Figure 71) is two bytes more than the length of an explicit GTS allocation frame (11 bytes, in Figure 70), which corresponds to the 16 additional bits for the flow specification field.

A second implicit GTS allocation request is performed by Node 3, as shown in Figure 71.b. Observe that contrarily to the scenario in Figure 70, the PAN Coordinator still uses the same one time slot GTS alternatively for both nodes (using round robin scheduling). This is because the admission control mechanism accepted Node 3 for sharing the GTS (one time slot) with Node 2 since their requirements are still satisfied. In fact, based on this configuration, the delay bound computed by the admission control mechanism is equal to 258.82 ms, which is lower than the maximum delay requirement of 300 ms.



[Beacon Frame] Before a GTS Allocation

[GTS Request Frame] – Node 2 – $F_{spec,2}$ (b2 = 120 bits, r2 = 0.6 kbps, D2 = 300 ms)

[Ack Frame] Send by the PAN Coordinator to Node 2

[Beacon Frame] After a GTS Allocation of Node 2

[Data Frame] Send by Node 2 in its GTS

[Ack Frame] Send by the PAN Coordinator to Node 2

a) The first implicit allocation request, sent by node 2

[Beacon Frame] Before the GTS alocation of Node 3

[GTS Request Frame] – Node 3 – $F_{spec,3}$ (b3 = 120 bits, r3 = 0.6 kbps, D3 = 300 ms)

[Ack Frame]

[Data Frame] Send by Node 2 in its GTS

[Ack Frame]

[Beacon Frame] After a GTS Allocation of Node 3
GTS allocated for Node 2

GTS allocated for Node 3

[Data Frame] Send by Node 3n its GTS

[Ack Frame]

b) The second implicit allocation request, sent by node 3

[Beacon Frame] Before the GTS alocation of Node 4

[GTS Request Frame] – Node 4 – $F_{spec,4}$ (b4 = 120 bits, r4 = 0.6 kbps, D4 = 300 ms)

[Ack Frame]

[Beacon Frame] After a GTS Allocation of Node 3
GTS allocated for {Node 2, Node 4}

[Data Frame] Send by Node 2 in its GTS

[Ack Frame]

[Beacon Frame] After a GTS Allocation of Node 3
GTS allocated for {Node3, Node 2}

[Data Frame] Send by Node 3n its GTS

[Ack Frame]

[Ack Frame]

[Beacon Frame] After a GTS Allocation of Node 3
GTS allocated for {Node 4, Node 3}

[Data Frame] Send by Node 4 in its GTS

[Ack Frame]

c) The third implicit allocation request, sent by node 4

Figure 71 - i-GAME GTS allocation scenario with three nodes

The sequence corresponding to a third implicit GTS allocation request made by Node 4 is presented in Figure 70.c. Observe that after this request, the PAN Coordinator increments the number of allocated time slots. In case of an allocation of one time slot by these three nodes, the delay bound returned by the admission control mechanism is equal to 391.16 ms, which is greater than the one required (300 ms). When incrementing the GTS length by one time slot (to two), the delay bound guaranteed for the three nodes is then equal to 258.82 ms, which satisfies the 300 ms delay requirement. Observe that the round robin scheduling algorithm is maintained by the PAN Coordinator in each new superframe.

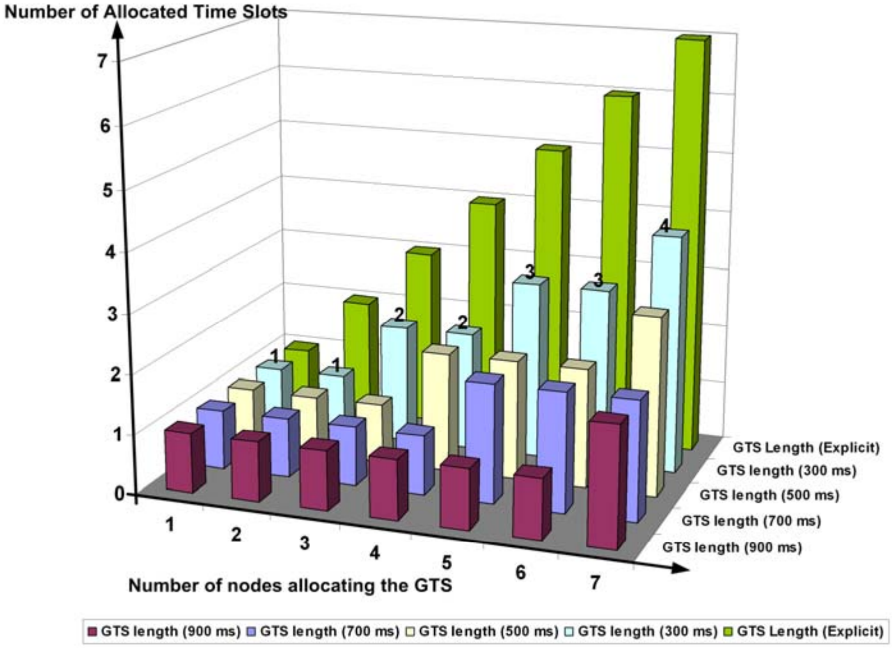


Figure 72 - Nodes allocating a GTS with i-GAME versus the GTS length

To observe the impact of the delay requirement on the improvement of the GTS efficiency, we have run the experimental testbed for three additional scenarios, in which nodes choose the other delay classes (0x1, 0x2, and 0x3) for their delay requirements, according to Table 10. The results are plotted in Figure 72 and perfectly confirmed by the analytical formulations in [35,69].

Observe that relaxing the delay bound of 7 nodes requesting GTS enables to save, in the case of 900 ms of delay requirement, up to 5 time slots as compared to explicit allocation, while still satisfying the delay bounds. This (saved) time can be used by the contention-access period, thus improving the utilization of the network.

6.6 Conclusions

While the IEEE 802.15.4 protocol is sufficiently flexible to fit the requirements of WSN applications with timing requirements, a standard for low-rate wireless personal area networks, it still leaves room for potential improvements. The definition of the i-GAME approach [69] contributes to a new GTS allocation mechanism motivated by the bandwidth utilization inefficiency of the explicit GTS allocation mechanism supported by the IEEE 802.15.4 protocol for flows with low rates. i-GAME overcomes this problem by allowing to share the same GTS between multiple flows based on their traffic specifications and delay requirements. This theoretical analysis has been experimentally validated by a testbed that firstly demonstrates the practical feasibility of i-GAME, which has been implemented in a real platform, and secondly confirms the improvement resulting from using the implicit GTS allocation mechanism over the classical explicit GTS allocation in terms of bandwidth utilization. This chapter also showed that the implementation of i-GAME only requires minor add-ons to the IEEE 802.15.4 protocol and ensures backward compatibility with the standard, making it easily implemented in Commercial-Off-The-Shelf (COTS) IEEE 802.15.4 platforms.

Chapter 7

On the IEEE 802.15.4 Slotted CSMA/CA Mechanism

This chapter addresses the performance evaluation of the Slotted CSMA/CA mechanism comparing experimental and simulation results. In this chapter the experimental results are compared with simulation results. The comparison is performed taking into account the probability of successful transmissions and network throughtpt and as a function of the Offered Load, for several Superframe Orders. The chapter ends with some conclusions on the comparison.

7.1 Introduction

In this chapter, our implementation of the Slotted CSMA/CA algorithm is assessed, enabling alto to analyse the real performance of the mote technology under use.

Additionally, we have use our Simulation Model [26] of the IEEE 802.15.4 protocol, developed in OPNET [25], to enable the comparision between experimental and simulation results.

We used simulation and experimental Testbeds to analyse the performance limits of the slotted CSMA/CA mechanism for broadcast transmissions (e.g. without acknowledgements). This was done for different network settings, in order to understand the impact of the protocol attributes on the network performance, namely in terms of Network Throughput (*S*) and Probability of Successful (*Ps*), given a certain offered load (*G*). Simulation results and analysis are based on an extensive study presented in [70,71].

7.2 OPNET Simulation Model

The IEEE 802.15.4 OPNET simulation model [26], supports the Physical Layer and the Medium Access Contrl (MAC) sub-layer. This simulator was used due to its accuracy and sophisticated graphical user interface. The OPNET tool is an industry leading discrete-event network modelling and simulation environment. OPNET allows several add-on modules including a wireless module that enables an accurate modelling, simulation and analysis of wireless networks. Currently the simulation model [26] only supports the IEEE 802.15.4 star topology. The structure of the Simulation Model is presented in Figure 73.

The *wpan_sensor_node* object is used to represent one sensor node. This object can assume two different behaviours in the network: (1) PAN Coordinator (unique and central node in the network) and (2) End Device, object that communicates with the central node. The *wpan_analyzer_node* object is used to capture global statistical data from the network.

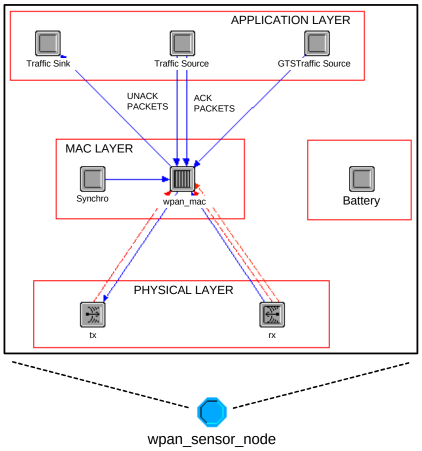


Figure 73 - Structure of the IEEE 802.15.4 Simulation Model

The structure of the IEEE 802.15.4 sensor nodes object (*wpan_sensor_node*) is composed of four functional blocks: (1) the *Physical Layer* - consists of a wireless radio transmitter and receiver compliant with the IEEE 802.15.4 specification, operating at the 2.4 GHz frequency band and a data rate equal to 250 kbps. The transmission power is set to 1 mW and the modulation technique is Quadrature Phase Shift Keying (QPSK); (2) the *MAC Layer* - implements the Slotted CSMA/CA and GTS mechanisms. The GTS data traffic incoming from the Application Layer is stored in a buffer with a specified capacity and dispatched to the network when the corresponding GTS is active. The data frames, stored in an unbounded buffer, use the slotted CSMA/CA algorithm for transmission during the Cotention Access Period (CAP). This block is also responsible for generating beacon frames and synchronizing the network; (3) the *Application Layer* - consists of several timers, two generating data traffic (i.e. traffic source and GTS traffic source) and one traffic sink. The traffic source generates unacknowledged and acknowledged data frames for transmissions during the CAP (using Slotted CSMA/CA). The GTS traffic source can produce unacknowledged or acknowledged time-critical data frames transmission using the GTS mechanism. The *Traffic Sink* process module receives frames forwarded from lower layers and performs network statistics; (4) the *Battery Module* - computes the consumed and the remaining energy levels. The default values of the current draws are set to those of the MICAz mote specification.

The values of all constants and variables in this model are considered for the 2.4 GHz frequency band, corresponding to a 250 kbps data rate, which is supported by the MICAz and TelosB motes. In this case, one symbol corresponds to 4 bits. For other frequency bands and data rates it is necessary to reconfigure some parameters in the Simulation Model.

7.3 Experimental and Simulation Testbeds

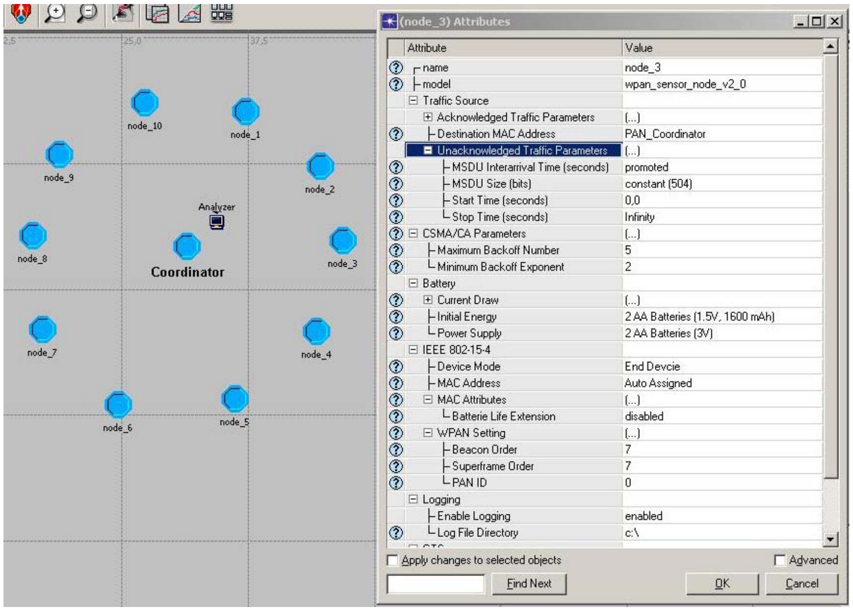
In the experiments, the Slotted CSMA/CA performance was tested for a number of Superframe configurations operating at full duty-cycle (no inactive period, *BO* = *SO*) and the only variation was the superframe order.

In general, both the simulation and experimental scenarios consist of 1 PAN Coordinator and 10 End Devices generating traffic (data frames with 63 bytes of length) at pre-programmed inter-arrival times (at the Application Layer) and a network/protocol analyzer capturing all the data for later processing and analysis. Several metrics were defined in order to evaluate the performance of the Slotted CSMA/CA mechanism. These metrics are a means of comparison between experimental and simulation results. The performance metrics analysed are the following:

- Network Throughput (*S*) – the fraction of correctly received traffic by the network analyzer normalized to the 250 kbps network capacity of the IEEE 802.14.5 Physical Layer.
- Success probability (*Ps*) – reflect the degree of reliability achieved by the network for successful transmissions. This metric is computed as *S* divided by the MAC sub-layer offered load (*G)*. It reflects the degree of reliability achieved by the network for successful transmissions.

The simulation and the experimental scenarios are depicted in Figure 74 and Figure 75, respectively.

In Figure 74 is possible to observe the network layout and the attributes of each Snd Device node (*wpan_sensor_node* model).



Attribute	Value
name	node_3
model	wpan_sensor_node_v2_0
Traffic Source	
Acknowledged Traffic Parameters	(...)
Destination MAC Address	PAN_Coordinator
Unacknowledged Traffic Parameters	(...)
MSDU Interarrival Time (seconds)	promoted
MSDU Size (bits)	constant (504)
Start Time (seconds)	0,0
Stop Time (seconds)	Infinity
CSMA/CA Parameters	(...)
Maximum Backoff Number	5
Minimum Backoff Exponent	2
Battery	
Current Draw	(...)
Initial Energy	2 AA Batteries (1.5V, 1600 mAh)
Power Supply	2 AA Batteries (3V)
IEEE 802-15-4	
Device Mode	End Devcie
MAC Address	Auto Assigned
MAC Attributes	(...)
Batterie Life Extension	disabled
WPAN Setting	(...)
Beacon Order	7
Superframe Order	7
PAN ID	0
Logging	
Enable Logging	enabled
Log File Directory	c:\

Figure 74 - Simulation Test bed

Figure 75 depicts the experimental testbed, using MICAz motes. The configuration node is used to setup the message inter-arrival times and frame size. When the Coordinator node is turned on, the end nodes synchronize with its beacon and start transmitting data frames with the respective configurations (inter-arrival time and frame size). The packet analyser used to capture all the generated packets was the Chipcon CC2420 Packet Analyser [41]. It generates a text file with all the received packets and the corresponding timestamps, enabling to retrieve all necessary data (embedded in the packets payload) using a parser application, in order to avoid serial communications.

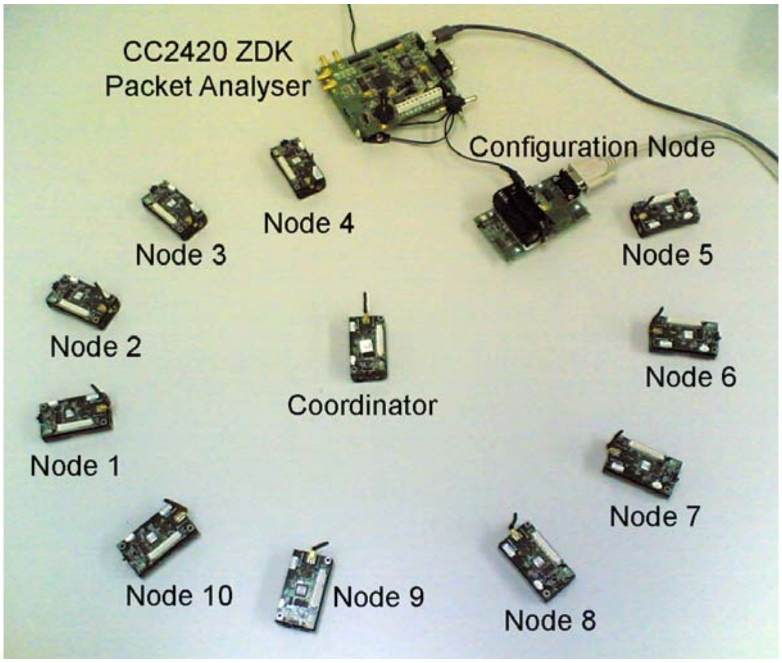
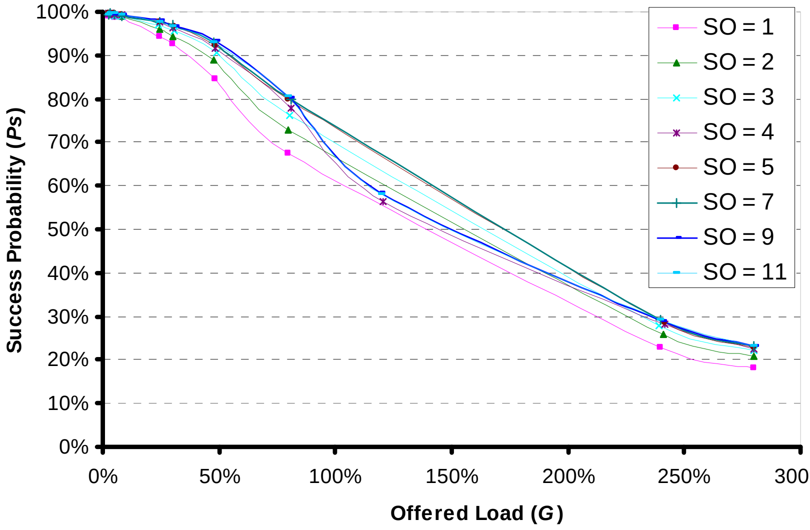


Figure 75 - Experimental Test bed

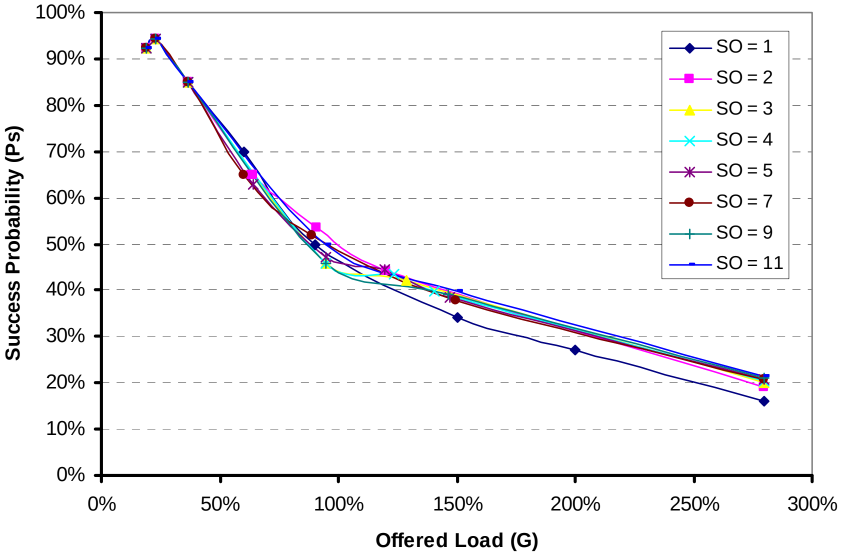
Both the simulation and experimentation scenarios conditions are considered identical. Nevertheless is reasonable to admit that the experimental results suffer from uncontrollable factors, such as RF interferences, processing limitations and memory constraints. Actually, as already mentioned in Chapter 3, TinyOS also imposes some overhead in the internal operations that can decrease the overall behaviour of each mote.

7.4 Performance Analysis

In this section the simulation and the experimental results are presented and briefly analysed. The charts in Figure 76 (*a* and *b*) compares the transmission Success Probability (*Ps*) and the offered load, for a given superframe order (*SO*). Both denote that *Ps* decreases with the increase of traffic and for lower *SO*. A reason for this behaviour can be the increased number of failures in the Slotted CSMA/CA mechanism due to higher medium congestion.



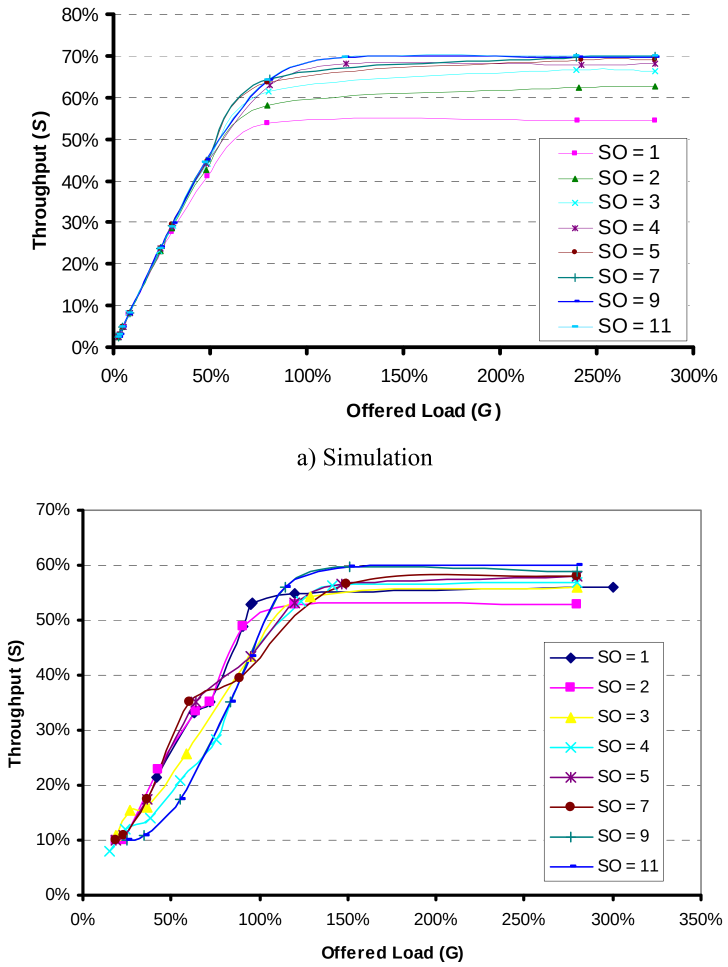
a) Simulation



b) Experimental

Figure 76 - Success Probability vs Offered Load

Figure 77 depicts the comparison between the network Throughput (*S*), that is the percentage of bandwidth utilization (with a maximum of 250 kbps), and the Offered Load (G). In the simulation chart (Figure 77.a) it is possible to observe that the network reaches a saturation state with approximately 70% of bandwidth utilization, while in the experimental evaluation (Figure 77.b) the saturation state is reached at approximately 60%. This maximum throughput is obtained for higher *SO* values.



a) Simulation

b)Experimental

Figure 77 - Throughput vs Offered Load

7.5 Conclusions

Simulation and experimental results allowed observing similar behaviours, which consolidates the consistency of the implemented version of the Slotted CSMA/CA mechanism and of the IEEE 802.15.4 protocol in general.

As it could be expected, the simulation results for Throughtput and Probability of Success are higher that the experimental results. We believe that this is mainly because the simulation model does not consider some of the physical constraints of the MICAz mote, especially the processing power, the internal delays due to TinyOS overheads and the normal interferences of a real wireless medium.

Considering the exemplifying case of the experiment where *SO* = *BO* = 5, Figure 78 depicts the Throughput and the Success Probability curves for different network loads. In this figure, it is possible to observe that the simulation and experimental curves have the same behaviour.

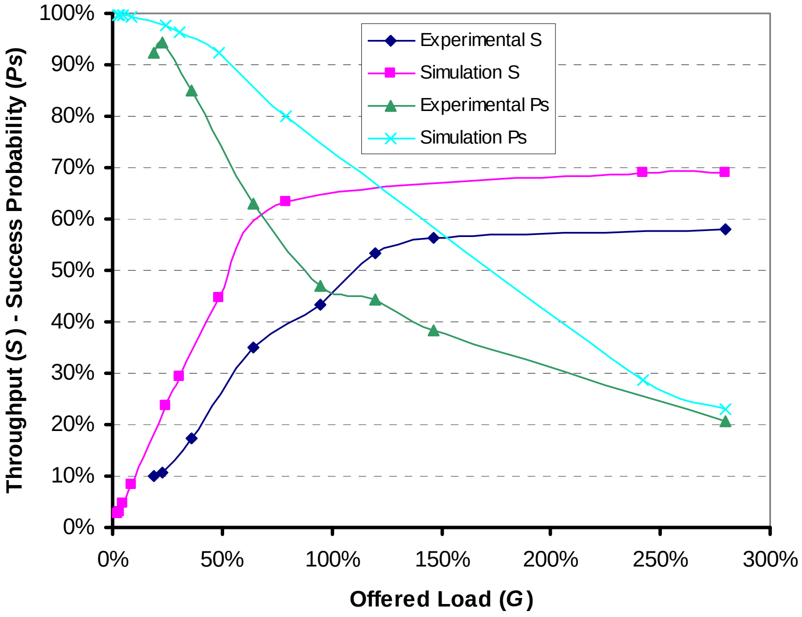


Figure 78 - Experimental vs Simulation (*SO*=*BO*=5)

One of the reasons for a lower performance with lower *SO* is due to a more probability of transmission deference (e.g. number of frames that were deferred to the next superframe because the device could not send them in the current one). The transmission deference problem is more frequent with lower Superframe Orders (*SO*) as the Superframe Duration is smaller. Another factor for the lower performance is the overhead of the beacon frame transmission, which is more significant in lower *SO* values.

Chapter 8

General Conclusions and Future Work

This chapter reviews the research objectives of this Thesis and summarises its major results, highlighting how the research contributions fulfilled the original research objectives. Finally, this chapter ends with some remarks about the future work in the area of Wireless Sensor Networks and ubiquitous computing.

8.1 General Conclusions

Research in the areas of distributed embedded systems, ubiquitous computing systems and wireless sensor/actor networks is gaining more and more momentum. However, in spite of the unlimited number of target applications for these systems that have already been identified by the community, just a few of them have seen the light. This is mainly because there is no mature and cost-efficient technology able to fulfil the needs of these large-scale systems.

It is a fact that, most of the times, standardization efforts lead to an easier, faster, and widespread development and use of commercial-off-the-shelf (COTS) technologies. Our feeling is that the same case may apply to the above mentioned areas, that is, using standard and COTS technologies will speed up the development of real applications.

This Thesis is about using the IEEE 802.15.4 and ZigBee protocols and technologies to engineer wireless sensor network applications. We are particularly concerned with those applications where timeliness plays a non negligible role, i.e. where the temporal behaviour of the different agents (network, task, processes, operating system) is as much important as their logical correctness. These systems are called real-time computing systems.

The ubiquity and pervasiveness of future distributed systems will lead to a very tight integration and interaction between embedded computing devices and the physical environment, via sensing and actuating actions. Such cyber-physical systems require a rethinking in the usual computing and networking concepts, and given that the computing entities closely interact with their environment, timeliness is of increasing importance.

In this context, the IEEE 802.15.4 and ZigBee communication protocols are potentially appealing for Wireless Sensor Networks. ZigBee supports several network topologies (star, mesh and cluster-tree), security mechanisms and application profiles. IEEE 802.15.4 allows dynamically adjustable duty-cycles per cluster, enabling energy-efficiency (nodes can sleep up to 100% of the time). The Medium Access Control (MAC) protocol is very flexible, enabling the differentiation between real-time traffic (contention-free; bandwidth/delay guarantees) through the GTS (Guaranteed Time Slot) mechanism, and best effort traffic (contention-access) through the Slotted CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) mechanism. Finally, there is an exponential growth in available ZigBee technology, although the cluster-tree network solution is not commercially supported yet.

Throughout the last year, our bet was to conceive an implementation of the IEEE 802.15.4/ZigBee protocol stack, which is now available as open-source [24]. This implementation was developed in nesC/TinyOS for the MICAz and TelosB motes and is described in Chapter 3. Since energy-efficiency and timeliness are major concerns, we have been favouring the synchronized cluster-tree topology over the mesh topology. For that purpose, the required Network Layer services (e.g. association/disassociation), synchronization and beacon/superframe scheduling mechanisms were also implemented, tested and validated, as described in Chapter 4.

ZigBee cluster-tree networks suffer from the single point of failure problem in the ZigBee Routers. ZigBee has a native fault-tolerant mechanism that is able to assign a new parent (ZigBee Router) to an orphan device. Nevertheless, since this mechanism reacts to link loss, it leads to network inaccessibility times that may be unacceptable for some critical applications. Therefore, we describe (Chapter 5) a proactive re-association mechanism that is able to find an alternative parent upon the detection of link quality degradation below a predefined threshold. Importantly, this mechanism eliminates network inaccessibility times and maybe an interesting means for mobility support.

The implemented IEEE 802.15.4 protocol stack also enabled the experimental assessment of the GTS and Slotted CSMA/CA mechanisms. In Chapter 6, we describe how an implicit GTS allocation mechanisms (i-GAME) that leads to a better utilization of network bandwidth by enabling several nodes to share the same GTS has been experimentally validated. A comparison between the experimental and simulation behaviour of the Slotted CSMA/CA mechanism was carried out and presented in Chapter 7.

In summary, we confirm the initial hypothesis of this Thesis, i.e., the IEEE 802.15.4 and ZigBee protocols can be used for Wireless Sensor Network applications. Nevertheless, some open and ambiguous issues in the standard specifications require some protocol add-ons and appropriate system planning and network dimensioning. This is particularly true for supporting scalable, energy-efficient and time-sensitive applications with ZigBee Cluster-Tree network topologies.

Notably, all the contributions in this Thesis witnessed a positive feedback from the scientific and industrial communities and some of them were supported by publications in top-ranked conferences and journals.

8.2 Future Work

The work underneath this Thesis served both as a sink of some of our recent scientific results and also as a spring of many research challenges, extensions and evolutions. A snapshot of ongoing and future work around this Thesis is presented next.

Concerning the IEEE 802.15.4/ZigBee protocol stack, we aim at:

- migrating the current implementation to TinyOS 2.0; this is being performed within the TinyOS Network Protocol Working Group [58];

- porting it to other hardware platforms (other types of motes) and other operating systems (e.g. Contiki) or even simpler/lighter OS-less kernels; ATMEL [55] is currently porting our protocol stack to their platform;
- supporting ZigBee mesh networks.

In what relates to ZigBee cluster-tree networks, we envisage to:

- implement, test and validate the proactive re-association mechanism presented in Chapter 5;
- analyse how resynchronization of a cluster-tree network, after a beacon/superframe rescheduling (refer to Chapter 4) can be performed without (or minimizing) interrupting normal communications;
- perform an experimental assessment of the throughput and message delay (min, mean, max) using the GTS mechanism in cluster-tree networks, comparing to the theoretical analysis in [36].
- find a solution for neighbour nodes belonging to different clusters to directly communicate (avoiding multi-hop communication via the tree-routing protocol); this is a problem since two non-sibling neighbour nodes will be active at different time windows;

On IEEE 802.15.4 CSMA/CA, we plan to:

- perform an experimental assessment of Slotted CSMA/CA with the traffic differentiation mechanisms proposed in [72];
- an experimental assessment of a mechanism (h-NAME, under development) that resolves the hidden-node problem;
- supporting the non beacon-enabled mode (used in ZigBee mesh networks), namely the unslotted CSMA/CA MAC mechanism.

Chapter 8 - Conclusions and Future Work

References

[1] Google Maps, http://maps.google.com, 2007

[2] Microsoft Researsh, Sensor Maps, http://atom.research.microsoft.com/sensormap/, 2007

[3] The Economist, “When everything connects”, April 28th – May 4th, 2007

[4] IEEE-TG15.4, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE standard for Information Technology, 2003.

[5] IEEE 802.15 WPAN™ Task Group 4 (TG4), http://grouper.ieee.org/groups/802/15/pub/TG4.html

[6] ZigBee Alliance (2006), ZigBee Specification 2006, http://www.zigbee.org/

[7] J. Zheng and J. L. Myung, "Will IEEE 802.15.4 Make Ubiquitous Networking a Reality? A Discussion on a Potential Low Power, Low Bit Rate Standard", IEEE Communications Magazine, vol. 42, No. 6, pp. 140- 146, , 2004.

[8] D. Geer, "Users Make a Beeline for ZigBee Technology", IEEE Computer Society Press, vol. 38, Issue 12, pp. 16-19, Dec., 2005.

[9] J. Adams, "Building low power into wireless sensor networks using ZigBee technology", Industrial Embedded Systems Resource Guide, Networking: Technology, pp. 26-30, 2005.

[10] T. Culter, "Deploying ZigBee in existing industrial automation networks", Industrial Embedded System Resource Guide, Networking: Technology, pp. 34-36, 2005.

[11] C. Herzog, "Creating value with ZigBee Networks", Industrial Embedded System Resource Guide, Networking: Technology, pp. 31-33, 2005.

[12] A. Koubâa, M. Alves, E. Tovar, “A Two-Tiered Architecture for Real-Time Communications in Large-Scale Wireless Sensor Networks.”, WIP Session on the 17th Euromicro Conference on Real-Time Systems (ECRTS’05), Palma de Mallorca, Spain, 2007.

[13] The ART-WiSe Framework, www.hurray.isep.ipp.pt/art-wise/, 2007

[14] M. Alves, A. Koubaa, A. Cunha, R. Severino, E. Lomba, “On the Development of a Test-Bed Application for the ART-WiSe Architecture”, In Euromicro Conference on Real-Time Systems (ECRTS 2006), (WiP Session), July 2006.

[15] IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Network - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, http://standards.ieee.org/

[16] J. Leal, A. Cunha, A. Koubâa, M. Alves, “A Gateway interface for a two-tiered architecture for wireless sensor networks”, to be presented in EFTA 2007 Work-in-progress session, September 2007

[17] IEEE 802.11e-2005, 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:

Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, http://standards.ieee.org/

[18] A. Cunha, M. Alves, A. Koubaa. "An IEEE 802.15.4 protocol implementation(in nesC/TinyOS): Reference Guide v1.2", IPP-HURRAY Technical Report, HURRAY-TR-061106, Nov 2006.

[19] A. Cunha, M. Alves, A. Koubâa, "Implementation of the ZigBee Network Layer with Cluster-tree Support", IPP-HURRAY Technical Report, HURRAY-TR-070510, May 2007.

[20] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, D. Culler, "The nesC language: A Holistic Approach to Networked Embedded Systems", in Proceedings of the Programming Language Design and Implementation, 2003.

[21] TinyOS, www.tinyos.net, 2007

[22] Crossbow, MICAz Datasheet, www.xbow.com, 2007

[23] Crossbow, TelosB Datasheet, www.xbow.com, 2007

[24] Open-ZB, "Opensource toolset for IEEE 802.15.4 and ZigBee", http://www.open-zb.net/, 2007.

[25] OPNET Technologies, Inc., Opnet Modeler Wireless Suite - ver. 11.5A, http://www.opnet.com

[26] Petr Jurčík, Anis Koubâa, The IEEE 802.15.4 OPNET Simulation Model: Reference Guide v2.0",www.open-zb.net, IPP-HURRAY Technical Report, HURRAY-TR-070509, May 2007

[27] A. Cunha, A. Koubâa, R. Severino, M. Alves, "Open-ZB – An Open-Source Implementation of the IEEE 802.15.4/ZigBee Protocol Stack", to be presented in Mobile Ad-hoc and Sensor Systems 2007 (MASS 2007), Pisa, Italy, October 2007

[28] A. Koubâa, A. Cunha, M. Alves, "A Time Division Beacon Scheduling Mechanism for IEEE 802.15.4/ZigBee Cluster-Tree Wireless Sensor Networks", in the proceedings of Euromicro Conference on Real-Time Systems (ECRTS 2007), Pisa(Italy), July 2007

[29] A. Cunha, M. Alves, A. Koubâa, "Implementation Details of the Time Division Beacon Scheduling Approach for ZigBee Cluster-Tree Networks", IPP-HURRAY TR-070102, 2007

[30] S. Ben Attia, A. Cunha, A. Koubâa, M. Alves, "Fault-Tolerance Mechanisms for ZigBee Wireless Sensor Networks", To be presented in the WIP Session of the Euromicro Conference on Real-Time Systems (ECRTS 2007), Pisa(Italy), July 2007

[31] A. Koubaa, M. Alves, E. Tovar, A. Cunha, "An Implicit GTS Allocation Mechanism in IEEE 802.15.4: theory and practice", to be published in the Journal of Real-Time Systems, 2007.

[32] A. Cunha, A. Koubâa, M. Alves, "Implementation of the i-GAME Mechanism in IEEE 802.15.4 WPANs", IPP-HURRAY! Technical Report, TR060702, July 2006.

[33] IETF, RFC 3561 Ad hoc On-Demand Distance Vector (AODV) Routing, 2003

[34] Ed Callaway, "MAC Proposal for the Low Rate 802.15.4 Standard", MOTOROLA, 2001.

[35] Texas Instruments, "MSP430x21x1 Microcontroler Datasheet", http://focus.ti.com/docs/prod/folders/print/msp430f149.html, 2004

[36] ATmega128L 8-bit AVR Microcontroller Datasheet, Atmel ref: 2467MAVR-

11/04, http://www.atmel.com
[37] Chipcon, Texas Instruments, "CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver," http://www.chipcon.com, 2004.
[38] CrossBow, MIB510 Datasheet, www.xbow.com
[39] CrossBow, MIB520 Datasheet, www.xbow.com
[40] CrossBow, MIB600 Datasheet, www.xbow.com
[41] Chipcon, Texas Instruments Incorporated, “Chipcon Packet Sniffer for IEEE 802.15.4”, www.chipcon.com, 2006
[42] Daintree Networks, "Sensor Network Analyser," www.daintree.net, 2006.
[43] Chipcon, Texas Instruments Incorporated, “SmartRF Studion User Manual 6.5”, 2006, http://www.chipcon.com.
[44] Daintree Networks, “2400E Sensor Network Adapter Datasheet”, www.daintree,net, 2006
[45] Ember, www.ember.com, 2007
[46] Ember, “EM250 Single-Chip ZigBee/802.15.4 Solution”, Datasheet http://www.ember.com/products_zigbee_chips_e250.html, 2006
[47] Ember, “EM260 ZigBee/802.15.4 Network Processor”, Datasheet http://www.ember.com/products_zigbee_chips_e260.html, 2006
[48] Freescale semiconductor, www.freescale.com, 2007
[49] Freescale, “MC13192 2.4 GHz Low Power Transceiver for the IEEE® 802.15.4 Standard”, Technical Datasheet, www.freescale.com, 2007
[50] Freescale, “MC13201 2.4 GHz Low Power Transceiver for the IEEE® 802.15.4 Standard”, Technical Datasheet, www.freescale.com, 2007
[51] Integration, “IA OEM-DAUB1 2400 - IEEE 802.15.4/ZigBee USB Dongle”, www.integration.com, 2006
[52] Integration Associates, www.integration.com, 2007
[53] Texas Instruments, “Z-Stack”, http://focus.ti.com/docs/toolsw/folders/print/z-tack.html, 2007
[54] Texas Instruments, “CC2431 System-on-Chip for 2.4 GHz ZigBee/ IEEE 802.15.4 with Location Engine”, Datasheet, http://focus.ti.com/docs/prod/folders/print/cc2431.html, 2007
[55] Atmel, ”Z-link”, http://www.atmel.com/products/AVR/z-link/Default.asp, 2007
[56] ZigBee Alliance, Compliant Platforms, http://www.zigbee.org/en/certification/compliant_platforms.asp
[57] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, “System Architecture Directions for Networked Sensors”, in Architectural Support for Programming Languages and Operating Systems, pages 93–104, 2000.
[58] TinyOS Network Protocol Working Group, http://tinyos.stanford.edu:8000/Net2WG
[59] A. Rowe, R.Mangharam, and R. Rajkumar, "Rt-link: A time synchronized link protocol for energy constrained multi-hop wireless networks", in Third Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), USA, 2006.
[60] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocols for wireless microsensor networks", in Proceedings of the Hawaii International Conference on Systems Sciences, Hawai, 2000.

[61] G. Gupta and M. Younis, "Fault-Tolerant Clustering of Wireless Sensor Networks", in IEEE Wireless Communication and Networks Conference (WCNC 2003), vol. 3. New Orleans (Louisiana), 2003.

[62] V. A. Kottapalli, A. S. Kiremidjiana, J. P. Lyncha, E. Carryerb, T. W. Kennyb, K. H. Law, and Y. Lei, "Two-Tiered Wireless Sensor Network Architecture for Structural Monitoring", in Proceedings of the 10th Annual International Symposium on Smart Structures and Materials, San Diego (USA), 2003.

[63] Cisco Systems, "Understanding Spanning Tree Protocol". // IEEE 802.1D MAC Bridges 2004

[64] S. Tardieu, L. Pautet, "Building Fault Tolerant Distributed systems using IP multicast", Ecole Nationale Supérieure des Télécommunications de Paris

[65] S-J. Park, R. Sivakumar, "Sink-to-Sensors Reliability in Sensor Networks", Georgia Tech, MobiHoc'03, 2003

[66] A. Willig & H. Karl, "Data Transport Reliability in Wireless Sensor Networks — A Survey of Issues and Solutions"

[67] D. Bein, V. Jolly, B. Kumar & S. Latifi, "Reliability Modeling in Wireless Sensor Networks", International Journal of Information Technology, Vol. 11 No. 2, 2005

[68] G. Gupka & M. Younis "Fault-Tolerant Clustering of Wireless Sensor Networks", University of Maryland Baltimore County, IEEE, 2003

[69] A. Koubaa, M. Alves, E. Tovar, "i-GAME: An Implicit GTS Allocation Mechanism in IEEE 802.15.4", in Proc. of Euromicro Conference on Real-Time Systems (ECRTS 2006), Dresden (Germany), July 2006.

[70] A. Koubaa, M. Alves, E. Tovar, "A Comprehensive Simulation Study of Slotted CSMA/CA for IEEE 802.15.4 Wireless Sensor Networks", In IEEE WFCS 2006, Torino (Italy), June 2006.

[71] A. Koubâa, M. Alves, E. Tovar, "On the Performance Limits of Slotted CSMA/CA in IEEE 802.15.4 for Broadcast Transmissions in Wireless Sensor Networks", IPP-HURRAY Technical Report, HURRAY-TR-060202, Feb. 2006.

[72] A. Koubaa, M. Alves, B. Nefzi, Y. Q. Song, "Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for Time-Critical Events in Wireless Sensor Networks",In Proc. of the Workshop of Real-Time Networks (RTN 2006), Satellite Workshop to (ECRTS 2006), July 2006.

[73] A. Koubâa, M. Alves, and E. Tovar, "GTS Allocation Analysis in IEEE 802.15.4 for Real-Time Wireless Sensor Networks," in 14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS 2006). Rhodes Island (Greece): IEEE, 2006.

[74] A. Koubaa, M. Alvès, and E. Tovar, "Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks," in Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS'06), Rio de Janeiro (Brazil), 2006.

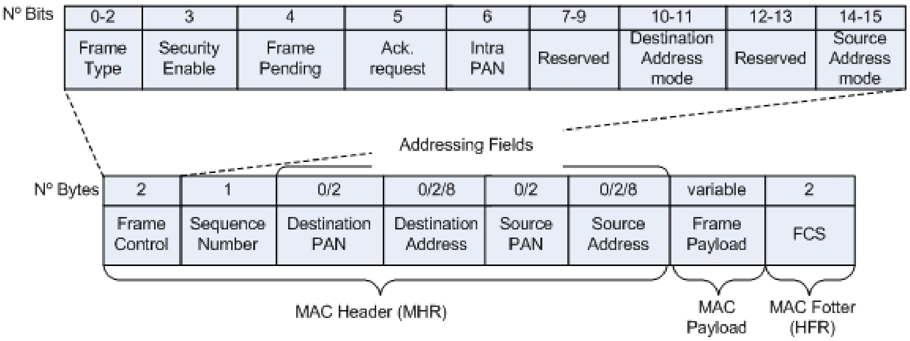
Annex A

IEEE 802.15.4 Frame Formats

This annex overviews the IEEE 802.15.4 frame formats. The standard defines four frame types: beacon frames, data frames, command frames and acknowledgment frames.

Annex A.1 – General IEEE 802.15.4 MAC Protocol Data Unit

The general IEEE 802.15.4 MAC frame format is depicted in Figure A.1.



Nº Bits	0-2	3	4	5	6	7-9	10-11	12-13	14-15
	Frame Type	Security Enable	Frame Pending	Ack. request	Intra PAN	Reserved	Destination Address mode	Reserved	Source Address mode

Nº Bytes	2	1	0/2	0/2/8	0/2	0/2/8	variable	2
	Frame Control	Sequence Number	Destination PAN	Destination Address	Source PAN	Source Address	Frame Payload	FCS

Figure A.1 – General IEEE 802.15.4 MAC frame format

The frame formats are composed of a MAC Header (MHR), a MAC payload and a MAC Fotter (MFR). The fields of the MHR appear in a fixed order; however, the addressing fields may not be included in all frames. The general MAC frame is formatted as illustrated in Figure A.1.

The MHR includes the following fields:

- *Frame control field* - depicted in Figure A.1 is a 16-bit field containing information of the following:
    - the frame type (e.g. data, beacon, command or acknowledge);
    - the frame is encrypted;
    - if there is any pending information (in the case of the source device is a Coordinator there can be pending data for the destination device to request);
    - the request for an acknowledgment by the recipient;
    - if the transmission is intra-pan, thus there is no need for the Destination PAN identifier in the addressing fields;

  - the type of destination address present in the addressing fields (e.g 16 bit short address or 32 bit extended address);
  - the type of source address present in the addressing fields (e.g 16 bit short address or 32 bit extended address).
- *Sequence number* – 8 bit field defining an unique sequence number identifier for the frame;
- *Destination PAN identifier* – if present defined the 16 bit short address of the destination PAN;
- *Destination Address* – if present can define the 16 bit short address or the 32 bit extended of the destination device;
- *Source PAN identifier* – if present defines the source 16 bit short address of the source PAN
- *Source Address* – if present can define the 16 bit short address or the 32 bit extended of the source device.

Table A.1 illustrates the possible combinations with the indication of the address field length in bytes.

Table A.1 – MAC Address fields - possible sizes and combinations

Address Field Size (Bytes)			**INTRA PAN**			**No INTRA PAN**		
			Source Address					
			No	Short	Long	No	Short	Long
Intra/No intra PAN	Destination Address	No	-	2	8	-	4	10
		Short	4	6	12	4	8	14
		Long	8	12	18	10	14	20

Annex A.2 – IEEE 802.15.4 Beacon Frame

The beacon frame format is depicted in Figure A.2.

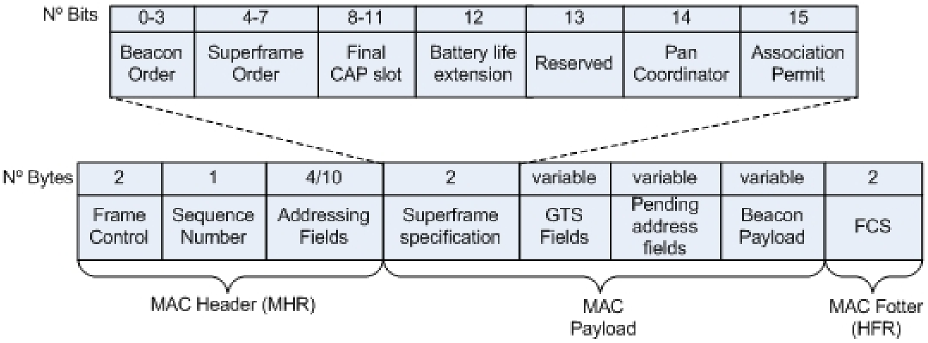


Figure A.2 – Beacon frame format

The beacon frame is divided into two parts: the MHR and the MAC payload. The MAC payload contains the following information:

- *Superframe Specification* is a 16-bit field that specifies different parameters related to the superframe such as the Beacon Order, the Superframe Order, the Final CAP Slot, the Battery Life Extension, the PAN Coordinator, the Association Permit subfields.
- The *GTS field* has a variable size and contains information on GTSs being allocated such as the GTS list and other control flags.
- The *Pending Address field* has a variable length and contains information of the devices that currently have messages pending with the Coordinator.
- The *Beacon Payload field* is an optional sequence of up to *aMaxBeaconPayloadLength* bytes specified to be transmitted in the beacon frame by the next higher layer. If *macBeaconPayloadLength* is nonzero, the set of octets contained in *macBeaconPayload* must be copied in this field.

The GTS fields (Figure A.3) include information about the allocated/dealocated GTS slots. The header of the GTS fields is composed by three fields: the GTS specification with the information of the number of GTS descriptors in the GTS list and the GTS permit; the GTS directions field with the information about the directions of the allocations, if there are GTS descriptor on the list; and the GTS list of descriptors. The GTS descriptors contain information about the short address of the devices allocation GTS, the start slot in the CFP and the number of slots allocated. Besides the allocated GTS slots, the GTS descriptors list include the deallocated GTS with the GTS descriptors containing information about the device address and with zero values in the start slot and length.

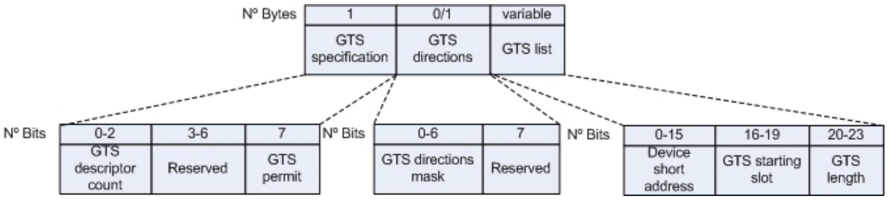


Figure A.3 - GTS field format

The pending addresses field (Figure A.4) contains information of the data stored in the Coordinator. This data must be requested by the receiver device in order to be sent by the Coordinator The header field of the pending address (or the pending address specification field) contains information about the number of number of short and extended addresses with pending data followed by the list of the devices organized with the short addresses first.

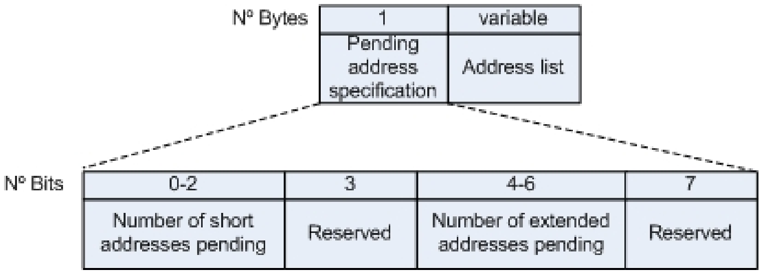

Nº Bytes
1
variable
Pending address specification
Address list
Nº Bits
0-2
3
4-6
7
Number of short addresses pending
Reserved
Number of extended addresses pending
Reserved

Figure A.4 – Pending addresses fields format

Annex A.3 – IEEE 802.15.4 Data Frame

The data frame format is depicted in Figure A.5. The data payload field contains the sequence of bytes that the Network Layer requested the MAC sub-layer to transmit.

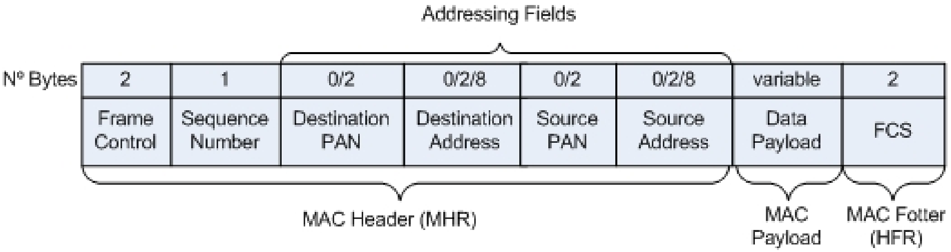


Figure A.5 – Data frame format

Annex A.4 – IEEE 802.15.4 Command Frame

The command frame format is depicted in Figure A.6.

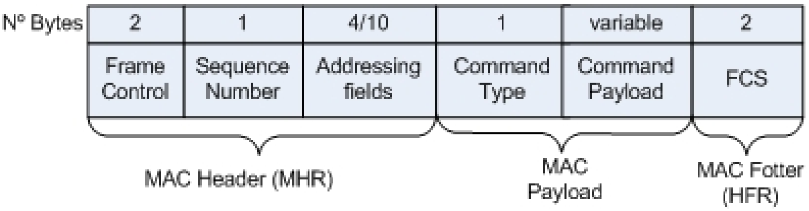


Figure A.6 – IEEE 802.15.4 command frame format

The command type field indicates the type of command. The possible commands are shown in Table A.2.

Table A.2 – IEEE 802.15.4 available command identifiers

Command Frame Identifier	Command Name
0x01	Association Request
0x02	Association Response
0x03	Disassociation Notification
0x04	Data Request
0x05	PAN ID Conflict Notification
0x06	Orphan Notification
0x07	Beacon Request
0x08	Coordinator Realignment
0x09	GTS Request
0x0a-0xff	Reserved

A.4.1 Association Request

The association request frame format is depicted in Figure A.7. The association request has the normal MHR fields, the command type (0x01 association request) and the capability information of the associated device.

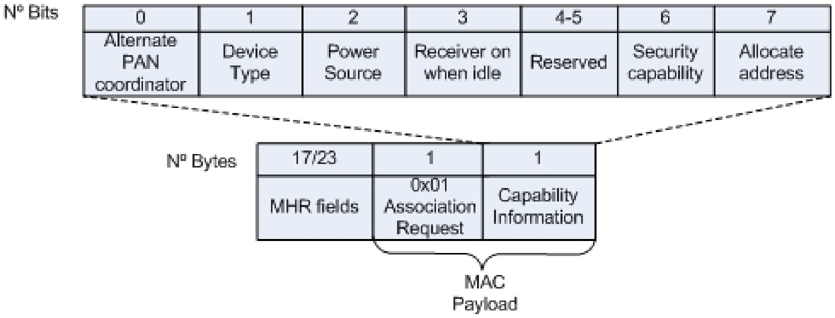


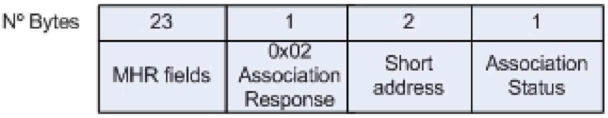
Figure A.7 – IEEE 802.15.4 association request command frame format

The capability information field (Figure A.7) contains the following information:

- *Alternate PAN Coordinator* – 1 if the device is capable of becoming a PAN Coordinator (assuming that the device can be a router);
- *Device type* – 1 – FFD; 0 –RFD;
- *Power source* – 1 if the device is main powered;
- *Receiver on when idle* – 1 if the receiver is on during the inactive period;
- *Security* – 1 if the device is capable of sending and receiving secured MAC frames with a security suite;
- *Allocate address* – 1 – if the device wants a short address; 0 if the device wants to communicate with the 64 bits extended address.

A.4.2 Association Response

The association response command frame is depicted in Figure A.8. This command is sent by one Coordinator confirming a request for an association. The association response command has an IEEE 802.15.4 MHR field (including the destination and source extended addresses), the command type (0x02 association response), the short address assigned and the association status with additional information about the association procedure.



N° Bytes	23	1	2	1
	MHR fields	0x02 Association Response	Short address	Association Status

Figure A.8 – IEEE 802.15.4 association response command frame format

The short address assigned can have the following values:

- *0x0000-0xfffd* – informing associated device that it shall use the assigned short address;
- *0xfffe* – informing the device that it shall use the 64bit extended address. In this case the device is associated to the network but it does not have a short address.
- *0xffff* – informing the device that the association procedure has failed and it shall not communicate on the PAN.

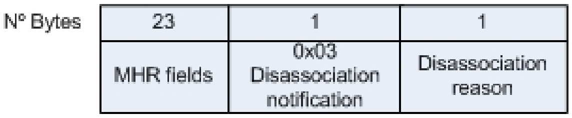
Table A.3 shows the possible association status information.

Table A.3 – Association response status

Association status	Description
0x00	Association successful
0x01	PAN at capacity
0x02	PAN access denied
0x03-0xff	Reserved

A.4.3 Dissociation Notification

Figure A.9 depicts the disassociation notification command frame.



N° Bytes	23	1	1
	MHR fields	0x03 Disassociation notification	Disassociation reason

Figure A.9 – IEEE 802.15.4 association response command frame format

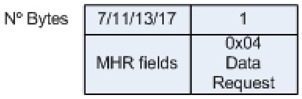
The disassociation notification has an IEEE 802.15.4 MHR field (including the destination and source extended addresses), the command type (0x03 disassociation notification) and the disassociation reason. Table A.4 shows the possible disassociation reasons.

Table A.4 – Disassociation reasons

Disassociation status	Description
0x00	Reserved
0x01	The Coordinator wishes the device to leave the PAN
0x02	The device wishes to leave the PAN
0x03-0xff	Reserved

A.4.4 Data Request

The data request command frame, depicted in Figure A.10 is sent to a Coordinator when a device has pending data. The Coordinator uses the pending data information field in the frame control (refer to Figure A.1) of the frames unicasted to a device to inform that the device can request or has stored pending data. The data request command frame includes the MHR fields and the command type (0x04 data request).

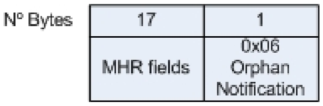


Nº Bytes	7/11/13/17	1
	MHR fields	0x04 Data Request

Figure A.10 – Data request command frame

A.4.5 Orphan Notification

The orphan notification command is depicted in Figure A.11. This frame includes the MHR fields and the command type (0x06 orphan notification).

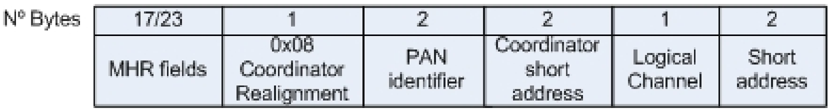


Nº Bytes	17	1
	MHR fields	0x06 Orphan Notification

Figure A.11 – Orphan notification command frame format

A.4.6 Coordinator Realignment

The Coordinator realignment command is depicted in Figure A.12. This command is sent by the Coordinator in response to an orphan notification command, in the case that the Coordinator is the corresponding parent of the orphan device.



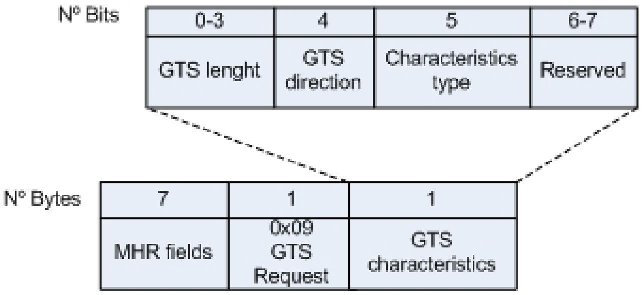
Nº Bytes	17/23	1	2	2	1	2
	MHR fields	0x08 Coordinator Realignment	PAN identifier	Coordinator short address	Logical Channel	Short address

Figure A.12 – Coordinator realignment command frame format

The Coordinator realignment frame includes the MHR fields, the command type (0x08 Coordinator realignment), the PAN identifier with the 16-bit address if the PAN, the Coordinator short address field with the 16-bit short address of the Coordinator, the logical channel field with the physical IEEE 802.15.4 channel where the Coordinator is operating and the short address field with the 16 bit short address of the orphan device.

A.4.7 GTS Request

Figure A.13 depicts the GTS request command frame. This frame includes the MHR fields, the command type (0x09 GTS request) and the GTS characteristics field. The command is used for both allocation and deallocation of GTS time slots.



Nº Bits	0-3	4	5	6-7
	GTS lenght	GTS direction	Characteristics type	Reserved

Nº Bytes	7	1	1
	MHR fields	0x09 GTS Request	GTS characteristics

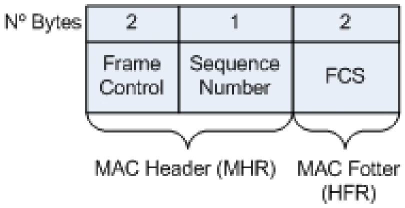
Figure A.13 – Guaranteed Time Slot request command frame format

The GTS characteristics have the following fields:

- GTS length - Number of superframe slots being requested for the GTS (valid range from 0 to 7);
- GTS Direction - GTS direction is defined relative to the direction of data frame transmissions by the device. Is 1 if the GTS is to be a receive-only GTS and 0 if the GTS is to be a transmit-only GTS;
- Characteristic Type - Is 1 if the characteristics refer to a GTS allocation or 0 if the characteristics refer to a GTS deallocation.

Annex A.5 – IEEE 802.15.4 Acknowledgement Frame

The acknowledgment frame format is depicted in Figure A.14. The sequence number field contains the number of the acknowledge message.



Nº Bytes	2	1	2
	Frame Control	Sequence Number	FCS

Figure A.14 - Acknowledgment frame format

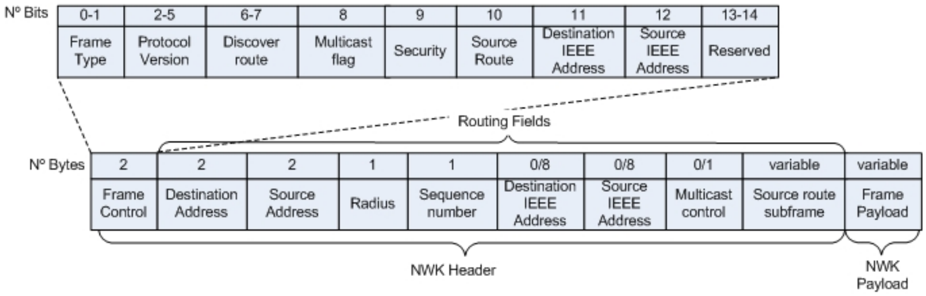
Annex B

ZigBee Network Layer Frame Formats

This annex overviews the ZigBee Network Layer (NWK) frame formats. The ZigBee standard defines two frame types: data frames and command

Annex B.1 – General ZigBee Network Layer Potocol Data Unit

The general ZigBee NWL frame format is depicted in Figure B.1.



Nº Bits	0-1	2-5	6-7	8	9	10	11	12	13-14
	Frame Type	Protocol Version	Discover route	Multicast flag	Security	Source Route	Destination IEEE Address	Source IEEE Address	Reserved

Nº Bytes	2	2	2	1	1	0/8	0/8	0/1	variable	variable
	Frame Control	Destination Address	Source Address	Radius	Sequence number	Destination IEEE Address	Source IEEE Address	Multicast control	Source route subframe	Frame Payload

Figure B.1 – General ZigBee Network Layer frame format

The general ZigBee frame formats (Figure B.1) are composed of a NWK Header and a NWK Payload.

The frame control field in NWK Header includes the following fields:

- *Frame Type* – Data Frame or NWK command frame (can assume de values of 0 for data frames and 1 for command frames);
- *Protocol version* – ZigBee protocol version used;
- *Discover route* – Option about the type of route discovery. This field can assume the following values;
  - 0x00 – Supress route discovery;
  - 0x01 – Enable route discovery;
  - 0x02 – Force route discovery.
- *Multicast Field* – 1 if it's a multicast frame, 0 if it's a unicast or broadcast frame
- *Security* – 1 if security is enable;
- *Source Routing* – 1 for source routing enabling the source route sub-fields;
- *Destination IEEE Address* – The presence of the extended IEEE address of the destination device in the routing fields;

- *Source IEEE Address* **–** The presence of extended IEEE address of the source device in the routing fields.

The Routing fields of the NWK Header include the following:

- *NWK Destination address* – short address of the destination device;
- *NWK Source address* – short address of the source device;
- *Radius* – Range of the radius transmission, this value is decremented by one on each relay. If it reaches 0 the frame is discarded;
- *Sequence number field* - NWK frames sequence number;
- *Destination IEEE Address* – Destination Extended address;
- *Source IEEE Address* – Source Extended address;
- *Multicast Control* – Multicast transmission flow control parameters;
- *Source route subframe* – This field includes a list of relay addresses. The fields included are the following:
    - *Relay count* – number of nodes in the relay list;
    - *Relay Index* – next relay;
    - *Relay list* –short addresses of the relay nodes.

Annex B.2 –ZigBee Network Layer Data Frame

The ZigBee NWL Data Frame is depicted in Figure B.2. The data payload field contains the sequence of bytes that the ZigBee NWK upper layer requested to transmit.

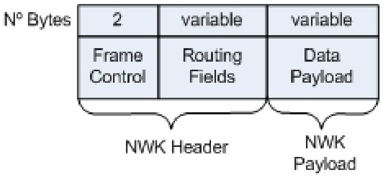


Figure B.2 – ZigBee Network Layer data frame format

Annex B.3 –ZigBee Network Layer Command Frame

The ZigBee NWL Command Frame is depicted in Figure B.3. These frames are mainly used for routing purposes.

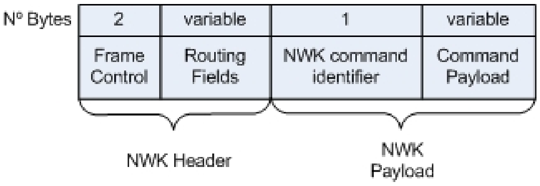


Figure B.3 – ZigBee Network Layer command frame format

The NWK command type field indicates the type of command. The possible commands are shown in Table B.1.

Table B.1 – ZigBee Network Layer available command identifiers

Command Frame Identifier	Command Name
0x01	Route Request
0x02	Route reply
0x03	Route Error
0x04	Leave
0x05	Route Record
0x06	Rejoin request
0x07	Rejoin response
0x00,0x08-0xff	Reserved

Annex C

IEEE 802.15.4 Reference Time Values

This annex presents the time durations in symbols, microseconds, backoff periods and number of clock tick of the timeslots and beacon intervals theoretical values as defined in the IEEE 802.15.4 protocol standard comparing them with the effective values used in the implementation, shown in Annex C.1. In Annex C.2 are the values obtained for the Crossbow MICAz mote and in Annex C.3 for the Crossbow TelosB mote.

Annex C.1 – IEEE 802.15.4 theoretical values of beacon intervals and time slot durations

Table C.1 - IEEE 802.15.4 theoretical values for the time slot durations

Time Slot Durations				
SO	**Symbols**	**Backoff Periods**	**Timeslot Duration (us)**	**Clock Ticks**
0	60	3	960	15
1	120	6	1920	30
2	240	12	3840	60
3	480	24	7680	120
4	960	48	15360	240
5	1920	96	30720	480
6	3840	192	61440	960
7	7680	384	122880	1920
8	15360	768	245760	3840
9	30720	1536	491520	7680
10	61440	3072	983040	15360
11	122880	6144	1966080	30720
12	245760	12288	3932160	61440
13	491520	24576	7864320	122880
14	983040	49152	15728640	245760

Table C.2 - IEEE 802.15.4 theoretical values for the beacon interval durations

MICAz Beacon Interval Durations				
BO	**Symbols**	**Backoff Periods**	**Duration (us)**	**Clock Ticks**
0	960	48	15360	240
1	1920	96	30720	480
2	3840	192	61440	960
3	7680	384	122880	1920
4	15360	768	245760	3840
5	30720	1536	491520	7680
6	61440	3072	983040	15360
7	122880	6144	1966080	30720
8	245760	12288	3932160	61440
9	491520	24576	7864320	122880
10	983040	49152	15728640	245760
11	1966080	98304	31457280	491520
12	3932160	196608	62914560	983040
13	7864320	393216	125829120	1966080
14	15728640	786432	251658240	3932160

Annex C.2 – MICAz effective values of beacon intervals and time slot durations

Table C.3 - MICAz time slot effective durations

MICAz Time Slot Durations				
SO	**Symbols**	**Backoff Periods**	**Timeslot Duration (us)**	**Clock Ticks**
0	60	3	1041,72	15
1	120	6	2083,44	30
2	240	12	4166,88	60
3	480	24	8333,76	120
4	960	48	16667,52	240
5	1920	96	33335,04	479
6	3840	192	66670,08	959
7	7680	384	133340,16	1917
8	15360	768	266680,32	3835
9	30720	1536	533360,64	7670
10	61440	3072	1066721,28	15340
11	122880	6144	2133442,56	30679
12	245760	12288	4266885,12	61359
13	491520	24576	8533770,24	122717
14	983040	49152	17067540,48	245435

Table C.4 - MICAz beacon interval effective durations

MICAz Beacon Interval Durations				
BO	Symbols	Backoff Periods	Duration (us)	Clock Ticks
0	960	48	16667,52	240
1	1920	96	33335,04	479
2	3840	192	66670,08	959
3	7680	384	133340,16	1917
4	15360	768	266680,32	3835
5	30720	1536	533360,64	7670
6	61440	3072	1066721,28	15340
7	122880	6144	2133442,56	30679
8	245760	12288	4266885,12	61359
9	491520	24576	8533770,24	122717
10	983040	49152	17067540,48	245435
11	1966080	98304	34135080,96	490870
12	3932160	196608	68270161,92	981739
13	7864320	393216	136540323,8	1963479
14	15728640	786432	273080647,7	3926958

Annex C.3 – TelosB effective values of beacon intervals and time slot durations

Table C.5 - TelosB time slot effective durations

TelosB Time Slot Durations				
SO	Symbols	Backoff Periods	Timeslot Duration (us)	Clock Ticks
0	60	3	1006,5	33
1	120	6	2013	66
2	240	12	4026	132
3	480	24	8052	264
4	960	48	16104	528
5	1920	96	32208	1056
6	3840	192	64416	2112
7	7680	384	128832	4224
8	15360	768	257664	8448
9	30720	1536	515328	16896
10	61440	3072	1030656	33792
11	122880	6144	2061312	67584
12	245760	12288	4122624	135168
13	491520	24576	8245248	270336
14	983040	49152	16490496	540672

Table C.6 - TelosB beacon interval effective durations

TelosB Beacon Interval Durations				
BO	**Symbols**	**Backoff Periods**	**Duration (us)**	**Clock Ticks**
0	960	48	16104	528
1	1920	96	32208	1056
2	3840	192	64416	2112
3	7680	384	128832	4224
4	15360	768	257664	8448
5	30720	1536	515328	16896
6	61440	3072	1030656	33792
7	122880	6144	2061312	67584
8	245760	12288	4122624	135168
9	491520	24576	8245248	270336
10	983040	49152	16490496	540672
11	1966080	98304	32980992	1081344
12	3932160	196608	65961984	2162688
13	7864320	393216	131923968	4325376
14	15728640	786432	263847936	8650752

Annex D

Superframe Duration Scheduling (SDS) algorithm

This annex presents the superframe duration scheduling (SDS) algorithm used in the Time Division Beacon Scheduling (TDBS) approach

The Superframe Duration Scheduling Algorithm [28]

```
$A = \{2^{BO_i}\}_{1 \le i \le N}$ the set of beacon intervals in the cluster-tree network
$\overline{BI}_{min} = 2^{BO_{min}}$ be the minimum beacon interval
organize the set $A = \{2^{BO_i}\}_{1 \le i \le N}$ in the increasing order of $BO_i$ such that
if (for a given i, j we have $\overline{BI}_i = \overline{BI}_j$ ) then
        if ( $\overline{SD}_i \ge \overline{SD}_j$ ) then put $\overline{BI}_i$ before $\overline{BI}_j$ in the set A
        else put $\overline{BI}_j$ before $\overline{BI}_i$ in the set A
Consider the slotted time line of length $\overline{BI}_{maj}$ where
the size of a slot is equal to $\min(SD_i)_{1 \le i \le N}$
for (each element i in the organized set A) do {
    search the first available consecutive time slots with a length at least equal to SDi
    write (i) in SDi consecutive time slot starting from the first available time slot
    repeat
if (write( i) in SDi consecutive time slots after each BIi interval) = false)
then return("the set is not schedulable")
until (end Major Cycle) }
Return ("the set is schedulable")
```

Annex D – Superframe Duration Scheduling (SDS) Algorithm

Annex E

i-GAME Implementation Code

This annex presents the algorithms and the nesC implementation of the i-GAME mechanism.

Annex E.1 – Management and Admission Control algorithms

i-GAME Management Algorithm [31]

```
1   type Flow = (id, b, r, D) //traffic specification and delay requirement
2   type FlowSetType = (Fi , where Fi requests a time slot in the CFP)
3   int N = 0; // the number of flow sharing a GTS
4   int k = 1; // the number of shared time slot
5   FlowSetType FlowSet; Flow F;
6   On (arrival of a new flow F) do {
7   N = N + 1;
8           if (admission_control (k, N, FlowSet, F) == false) {
9                   if (k == 7) { //the maximum number of GTSs is reached
10                          reject_request(F);
11                          N = N - 1; break;
12                  }
13                  else { // k < 7
14                          k = k + 1; //increase the length of the CFP
15                          goto line 8;
16                  }
17          }
18          else {
19          accept_request(F); //accept the new flow to share the GTS
20          FlowSet_Add(FlowSet, F); //add the new flow to the GTSset
21          }
```

i-GAME Admission control algorithm [31]

```
RTS = guaranteed bandwidth by one time slot
Ts = time slot duration
boolean admission_control (int k, int N, FlowSetType FlowSet, Flow F)
{
    boolean adm_crt = true;
    if (k <= N) {
        p = ceil (N / k);
        q = N – p * k – 1;
        for (int i = 1, i++; i<=N)
            if (( Di < (( bi / (k * RTS / N)) + ( p * Bi – q * Ts))) or (ri>k*RTS/N))
                adm_crt = false;
    } else { //the case (k>N) is considered as explicit allocation
        adm_crt = false;
    }
```

Annex E.2 – Management and Admission Control nesC implementation code

i-GAME Management nesC Implementation [32]

```
uint8_t iGAME_management_algorithm(Flow F)
{   N = N + 1;
    flowset[N - 1] = F;
    while(admission_control() == FLOW_REFUSED) {
        if (k == 7 ) {  //the maximum number of GTSs is reached
            //Flow rejected
            N = N - 1;
            return FLOW_REFUSED;
        } else {    k = k + 1;          }
    }  //accept the new flow to share the GTS
    return FLOW_ACCEPTED;  }
```

i-GAME Admission Control nesC Implementation [32]

```
uint8_t admission_control()
{
uint8_t adm_crt;
uint8_t p;
uint8_t q;
uint8_t i;
adm_crt=FLOW_ACCEPTED;
if ( k <= N)
{     if ( (N % k) != 0)
{     p = (N  / k) + 1;
                    }else{      p = (N  / k);             }
                    q = (p * k) + 1 - N ;
                    for (i=0; i < N;i++) {
                                        if ( ( GAME_get_delay_requirements_ms(flowset[i].delay_class)
<  (( iGAME_get_burst_size_byte(flowset[i].burst_class) * 8) / (k * (Rts / 1000) / N) ) + ( p *
Bi - q * Ts )) || (iGAME_get_arrival_rate_bps(flowset[i].rate_class) > (k * Rts / N)))
                           {       adm_crt = FLOW_REFUSED;     }
                    }
      }  else
      {  adm_crt = FLOW_REFUSED;          }
return adm_crt;
}
```