Evaluation assessment of energy-efficient duty cycling MAC protocols in ContikiOS

Elshal, Omar

Award date:
2017

Awarding institution:
Åbo Akademi University

ÅBO AKADEMI UNIVERSITY

MASTER'S THESIS

Evaluation Assessment of Energy-Efficient Duty Cycling MAC Protocols in ContikiOS

Author:
Omar ELSHAL

Supervisor:
Prof. Johan LILIUS

Advisor:
Dr. Sébastien LAFOND

November 9, 2017

Declaration of Authorship

I, Omar ELSHAL, declare that this thesis titled, "Evaluation Assessment of Energy-Efficient Duty Cycling MAC Protocols in ContikiOS" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Computer science is no more about computers than astronomy is about telescopes."

Edsger W. Dijkstra

ÅBO AKADEMI UNIVERSITY

Abstract

Faculty of Science and Engineering
Embedded Systems - Information Technologies Department

Master of Science

Evaluation Assessment of Energy-Efficient Duty Cycling MAC Protocols in ContikiOS

by Omar ELSHAL

As the field of internet of things and wireless sensor networks is expanding swiftly, the energy constraints of its networked embedded devices will increase significantly. This thesis tries to address the issues of this increase in the energy consumption of the wireless sensors. The thesis work includes an analysis of three different Medium Access Control (MAC) protocols which are ported into the Contiki operating system. The three protocols are the beacon-enabled mode of the IEEE 802.15.4 standard, the TSCH mode of the IEEE 802.15.4e standard, and the ContikiMAC protocol which is implemented by the ContikiOS developers. While the beacon-enabled mode is evaluated theoretically, the other protocols are evaluated through simulations using Cooja simulator. Although the results have demonstrated a more energy-efficient performance by the TSCH mode during normal operating modes, ContikiMAC results are more energy efficient during the transmission and reception operations of the sensor transceiver. The presented results can be utilized to pave the way for future development of a more energy-efficient MAC protocol. Therefore, the overall power consumption of a wireless sensor platform can be thoroughly exploited.

Acknowledgements

I would like to express my sincere gratitude to Professor Johan Lilius for all the guidance and support through this journey. His suggestions, encouragement and keen insight have been fundamental in getting this work to see the light. I would like also to thank Dr. Sébasiten Lafond for reviewing the thesis work and advising the writing process of the thesis. I am deeply grateful to the EIT Digital masters schools for giving me this invaluable opportunity to experience completely different atmospheres at two amazing European countries. The experience I got throughout this journey all over Europe is profound whether during technical courses, entrepreneurial events, winter and summer schools. I had unforgettable moments meeting incredible people from all over the world.

Furthermore, I would like to thank all my colleagues in 344E room. Special thanks to my friend Safayat Bin Hakim for all the inspiring talks we had through this year. These talks helped me quite a lot to during the thesis work. I would like to thank my friends Syed Usman and Haider Rizvi for the all great and fun moments we shared inside the Agora building and the delicious food you made at the Student village.

I would like to thank all my friends for supporting me from whether my home country or wherever they are pursuing their dreams currently. I am deeply grateful to my brothers and friends Omar Ghoniem, Abdelrahman Abdelkawy and Hisham Ramzy for their limitless support and encouragement during this journey.

Finally, I am extremely fortunate to have been blessed with such a loving and supportive family throughout my education and career pursuits. I could have never moved forward in life without the support and guidance from my father Ahmed and the love and affection from my mother Eman. I shall be forever in debt to you no matter what I do. I would like thank my sisters Alaa and Sondos, and my brother Hamza for all their love and support. I last but foremost would like thank my future family, my fiancée Shefaa for supporting me throughout the final stages of the thesis work and enduring all my ramblings and existential thoughts.

Contents

List of Figures

List of Tables

List of Abbreviations

WSN	**W**irless **S**ensor **N**etwork
ContikiOS	**Contiki O**perating **S**ystem
PAN	**P**ersonal **A**rea **N**etwork
IEEE	**I**nstitute of **E**lectrical and **E**lectronics and **E**ngineers
BS	**B**ase **S**tation
MAC	**M**edium **A**ccess **C**ontrol
IoT	**I**nternet **o**f **T**hings
6LoWPAN	IPv**6** over **Lo**w-power **W**ireless **A**rea **N**etworks
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IPV6	**I**nternet **P**rotocol **V**ersion **6**
TCP/IP	**T**ransmission **C**ontrol **P**rotocol over IP
BPSK	**B**inary **p**hase-**s**hift **k**eying
O-QPSK	**O**ffset **Q**uadrant **p**hase-**s**hift **k**eying
RF	**R**adio **F**requency
PSDU	**P**hysical layer **S**ervice **D**ata **U**nit
CSMA/CA	**Carrier**-**S**ense **M**ultiple **A**ccess with **C**ollision **A**voidance
FFD	**F**ull **F**unction **D**evice
RFD	**R**educed **F**unction **D**evice
OSI	**O**pen **S**ystems **I**nterconnection
RDC	**R**adio **D**uty **C**ycling
RPL	**R**ouing **P**rotocol for **L**ow-power and lossy networks
DODAG	**D**estination **O**riented **D**irected **A**cyclic **G**raph
UDP	**U**ser **D**atagram **P**rotocol
CoAP	**Co**nstrained **A**pplication **P**rotocol
CAP	**C**ontention **A**ccess **P**eriod
CFP	**C**ontention **F**ree **P**eriod
GTS	**G**uranteed **T**ime **S**lot
BO	**B**eacon **O**rder
SO	**S**uperframe **O**rder
BI	**B**eacon **I**nterval
SD	**S**uperframe **D**urtaion
SoC	**S**ystems-**o**n-**C**hip
MCU	**M**icro-**C**ontroller **U**nit
BLE	**B**luetooth **L**ow **E**nergy
ROM	**R**ead-**O**nly **M**emory
API	**A**pplication **P**rogramming **I**nterface

TSCH	**T**ime-**S**lotted **C**hannel **H**opping
DSME	**D**eterministic and **S**ynchronous **M**ulti-channel **E**xtension
ASN	**A**bsolute **S**lot **N**umber
RTOS	**R**eal-**T**ime **O**perating **S**ystem
DVFS	**D**ynamic **V**oltage **F**requency **S**caling
UDGM	**U**nit **D**isk **G**raph **M**edium

List of Symbols

a	Distance	m
P	Power	W ($\mathrm{J\,s^{-1}}$)
E_{switch}	Switching Energy	J
$E_{leakage}$	Leakage Energy	J
C_{total}	Total Capacitance	F
V_{dd}	Supply Voltage	V
I_{leak}	Leakage Current	A
Δt	Total duration	s
SO	MAC Superframe Order	$-$
BO	MAC Beacon Order	$-$
SD	Superframe Duration	$-$
BI	Beacon Interval	$-$
DC	Duty Cycle	$-$
ω	angular frequency	rad

Dedicated to my family, and my future one

Chapter 1

Introduction

The fields of sensor design, communication and information technology are constantly evolving at a very high pace resulting in the manifestation of Wireless Sensor Networks (WSNs). This type of network has special characteristics and features that can potentially enable these networks to connect the physical world with the virtual (computing) world and provide limitless numbers of practical applications in various fields. The applications of WSNs include cattle monitoring, supply chain management, precision agriculture, health care and military applications. However, some of these applications have encountered a huge number of constraints and challenges. Since the WSN field comprises various technical domains, the integration of all these technologies in a single embedded system can be a very challenging process especially when it has limited resource constraints.

A number of protocols have been introduced to provide communication between the nodes of a sensor network. Initially, the traditional IEEE 802.11 family of standards was used in the early versions of WSNs [1]. This type of WSN can be found in the networks that have high bandwidth requirements such as multimedia. The IEEE 802.11 standards were introduced in 1997 [2] providing wireless network technology for the majority of mobile systems. The IEEE 802.11 standards have two frequency bands: 2.4 GHz and 5 GHz which cover different protocols and usages.

Nevertheless, the low-power WSNs do not require the high data rates of the IEEE 802.11 standards due to the high-energy overhead imposed by these high data rates. As the bandwidth requirements in the low-power WSNs are typically low, a number of protocols have been developed to meet the low-power consumption and low data rate requirements. The IEEE 802.15.4 standard [3] was first published in 2003 to satisfy the low-power WSN requirements mentioned earlier. It defines the Medium Access Control (MAC) and Physical (PHY) layers of the wireless sensor network stack. These two bottom layers are very crucial for a wireless sensor optimization, as they can potentially enhance the performance and power consumption of the sensors in a WSN.

1.1 Contribution of the Thesis

The major aim of this thesis is to analyze the energy consumption of different Radio Duty Cycling (RDC) and Medium Access Control (MAC) protocols within the network stack of the Contiki operating system. A number of RDC/MAC protocols have been introduced to improve the power consumption of a wireless sensor, and thus we can achieve a longer lifetime for the network as a whole. These protocols provide different algorithms that optimize the activation of the sensor's radio transceiver, as it is the most power-consuming device of a sensor.
The thesis provides energy consumption evaluation of three different MAC behaviour modes which are beacon-enabled mode, TSCH mode and ContikiMAC. The beacon-enabled mode is analyzed theoretically, while the TSCH mode and ContikiMAC are analyzed through simulations using the Cooja simulation program based on the ContikiOS architecture. Furthermore, the power consumption for the nodes of the network is extensively analyzed to determine the energy efficiency of both protocols.

1.2 Structure of the Thesis

The structure of the thesis is explained in this section. **Chapter 2** starts by introducing the main concepts behind wireless sensor networks. Then, it continues by discussing the different communication protocols of WSNs, the IEEE 802.15.4 and IEEE 802.15.4e standards. Moreover, chapter 2 presents the structure of the Contiki network stack and the current supported network stacks. Additionally, the IEEE 802.15.4 beacon-enabled mode and the IEEE 802.15.4e TSCH mode are both explained. Finally, an overview of the software and toolchain used to implement the required tasks is presented.

The literature review is discussed in **chapter 3** mentioning the previous work that has been done on the beacon-enabled mode and the TSCH mode. Furthermore, a motivation of the thesis topic is discussed at the end of the chapter.

Chapter 4 explains the hardware and software implementations of the beacon-enabled mode and TSCH behaviour modes. Additionally, the configurations of the simulation work are discussed in detail in order to evaluate the software implementations of the protocols. Finally, the results of the simulation are shown and the main findings of this evaluation are presented.

The final conclusions of the thesis work are mentioned in **chapter 5**, along with the different directions for future work.

Chapter 2

Background

This chapter discusses the main concepts behind wireless sensor networks, their main communication protocols, and the IEEE 802.15.4 standard along with its behaviour modes: beacon and non-beacon enabled modes. The 2012 IEEE 802.15.4e standard and its new MAC behaviour modes such as TSCH are explained. Furthermore, an overview of the tools and software used is briefly presented.

2.1 Wireless Sensor Networks

The main objective of a Wireless Sensor Network (WSN) resides in transmitting the collected data from the sensors to a centralized processing station wirelessly. A sensor node may have more than the sensing component. It can also have an on-board processing to minimize the communication bandwidth, a communication transceiver for sending the data, and a storage component that can store the data until they are safely transmitted to a base station. As a result, the sensor nodes can provide functions other than data collection such as in-network analysis, correlation, and management of their own data and the data from other sensor nodes.

An example of a WSN network topology is described in Figure 2.1. The network has a tree-like topology where a base station (BS) acts as the root of the tree, relay nodes act as branches of the tree and sensor nodes are the children of each tree's branch. The base station (BS) gathers all the data from the relays typically to be sent to the cloud via internet. Furthermore, based on the application type, a number of relays can be deployed to receive the data from the sensor nodes within its transmission range which might be analyzed and processed to be sent to the base station depending on the routing protocol used.

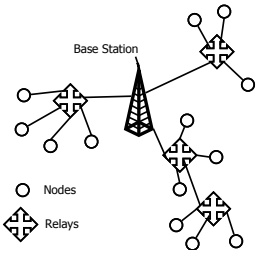

FIGURE 2.1: An example of a WSN network topology

Applications of WSNs with the aforementioned requirements might seem very limited to only high-end applications like radiation, land mine, and nuclear-threat detection systems. However, the number of applications and fields are quite huge and continue to increase along with the new advancements in technology. An overview of the current and potential applications is mentioned below [4].

- Environmental applications:
  - Precision agriculture
  - Natural disaster detection in remote areas
- Home-related applications:
  - Home Automation and monitoring
  - Automated meter reading
- Health-related applications:
  - Supporting and assisting the elderly
  - Drug administration
  - Remote monitoring patients' health and movement inside a hospital
- Other applications:
  - Cattle monitoring
  - Traffic flow surveillance
  - Inventory management
  - Battlefield and military applications

2.2 Challenges and Constraints

A number of factors and constraints affect the design of a WSN depending on its application area. This, in turn, lead to having more specific protocols and algorithms to satisfy the needs and requirements of each application domain. This section discusses the key challenges and design constraints of wireless sensor networks.

2.2.1 Energy Efficiency

One of the main constraints when designing any embedded system is the energy consumption along with its area and price. In general, most WSNs are powered by batteries [1], which have to be either replaced or recharged when their charge is empty. However, some sensor nodes can be simply discarded once their battery is drained, as their reactivation might affect the total energy consumption scheme of the whole network. Nevertheless, the most important factor is that a sensor node must be able to operate until either its *mission time* is over or the battery can be

replaced or recharged. The duration of a node's *mission time* depends on the application's nature. For instance, natural disaster applications may use sensors that need to operate for at least a couple of years while mission critical applications might only use sensors that can operate for a few hours.

It is clear from the points mentioned earlier that energy efficiency is one of the core design constraints for a WSN. As a result, the energy constraint design can potentially affect every aspect of a sensor node's design which is why it is discussed and analyzed thoroughly in the following chapters and throughout this work. It can be noticed from equation 2.1 below that the energy consumption of CMOS-based processors mainly depends on two factors, namely the switching energy (E_{switch}) and the leakage energy ($E_{leakage}$) [5].

$$E_{CPU} = E_{switch} + E_{leakage} = C_{total}V_{dd}^2 + V_{dd}I_{leak}\Delta t \ , \tag{2.1}$$

Where C_{total} is the total capacitance, V_{dd} is the supply voltage, I_{leak} is the leakage current, and Δt is the duration of operation. While the major factor that used to affect the total energy consumption is the switching energy, the latest developments resulted in making the leakage energy the dominant part of the equation taking up more than half of the total energy consumption [1]. Consequently, a number of techniques have been introduced to reduce the leakage energy such as dynamically switching off idle components and software-based techniques such as Dynamic Voltage Frequency Scaling (DVFS). As the energy efficiency is one of the cornerstones of this work, it is thoroughly discussed in the next chapters.

2.2.2 Design Constraints

Typically, WSNs deal with space and time where locality, coverage, data reliability and synchronization are of extreme importance, as they constitute the core purpose of wireless sensors. While the advancement in hardware and software development is rapidly evolving which suggests adding more processing power and complex features, it is extremely challenging to integrate such features in a simple board with constrained energy consumption. Consequently, creating simpler, cheaper, and more optimized and efficient devices forms the main design goal of wireless sensors of this type, in which they share limited processing power and storage capacity.

Moreover, these hardware constraints play a fundamental role in software development at different levels. For instance, WSN's operating systems must be small and simple meaning they should have limited memory footprints and efficient resource/task management like the Mini Operating System (MOS) used by Li-chun Ko et al. in [6]. In addition to software constraints, hardware constraints also impact the development of applications and protocols that are executed on the sensor nodes. For example, a number of data fusion and aggregation protocols can process the data before they are sent to eliminate the redundant data. However, these algorithms can potentially require processing power and memory footprints more than

the capabilities of the current sensor boards of a low cost. In conclusion, the whole software and hardware development cycle of WSNs must be designed in an efficient way where each layer can complement one another and make efficient use of such high resource-constrained system.

2.2.3 Security and Reliability

Data are considered the intrinsic currency of WSNs; therefore, the security and the reliability of the data transmitted across the network are extremely crucial in most WSNs applications. Security deals with different aspects such as data confidentiality [4] in which encryption ensures giving the correct access privileges to the right users, data integrity which assures the identity and digital signature of the data, and data availability which ensures that the data are always available when needed (preventing denial of service attacks). Moreover, reliability guarantees that the transmitted data are fully received whether there is no packet loss or resending the lost packets if a packet loss is found.

The remote and unattended nature of WSNs makes them vulnerable to different kind of malicious attacks and threats. For example, *denial-of-service* attack tries to disrupt the flow of communication between the nodes of the network making them unable to function properly [1]. This can be achieved by sending high-powered wireless signals to prevent a successful reception of packets between the nodes which is called a *jamming* attack. Based on the nature of the application under attack, the damages can be catastrophic. Therefore, a number of algorithms and techniques have been introduced to protect the sensor networks from these attacks. At the same time, it is really challenging formulate solutions that typically require low computational power, communication and memory footprints to satisfy the resource-constrained nature of the sensor boards in WSNs.

2.2.4 Autonomous Maintenance

Sensor networks are typically self-configuring systems. The goal of most WSN simulators is to be able to mimic unpredictable situations and states, as they generally must operate in remote areas or tough environments. Consequently, the sensor nodes must be able to reconfigure themselves, operate and cooperate with other nodes in the network. Additionally, the nodes should adapt to potential failures in other nodes such as mobile ones, changes in the environment and unstable situations without human interference.

The self-management of the sensor nodes includes autonomous configuration, maintenance, and adaptation of the nodes within the network. Therefore, these design goals are handled on different levels in software and hardware development, in addition to the network topology of the system.

2.3 Network Standards of WSN

During the last couple of decades, a variety of protocols and standards have been introduced to provide communication and network functionalities to wireless sensor networks. The **IEEE 802.15.4** standard is a low data rate protocol for short-range wireless networks. It was first published [4] in 2002 and is discussed in detail in the following section. While IEEE 802.15.4 covers the physical layer and MAC layer of the network stack, ZigBee and 6LoWPAN protocols cover the network and application layers on top of IEEE 802.15.4 layers.

ZigBee was initially introduced in 2003 by the ZigBee Alliance group [1] with the aim of providing low-cost communication technology to networks that require low data rates and power consumption. Furthermore, the IEEE and ZigBee Alliances collaborated together to build upon IEEE 802.15.4 low-level network layers in 2004 where ZigBee became the commercial name for the IEEE 802.15.4 technology.

Finally, another protocol which was developed during the inception of Internet of Things (IoT) is IPv6 over Low-power Wireless Area Networks (**6LoWPAN**). It is a set of standards developed by the Internet Engineering Task Force (IETF) in 2007 [7] with the aim of enabling efficient use of IPV6 over IEEE 802.15.4 low-power wireless networks for devices with limited area, data rate, and memory footprints such as WSN. The 6LoWPAN protocol applies IP communication capabilities to the nodes of the network by building an adaptation layer on top of the IEEE 802.15.4 MAC layer, and thus providing these nodes with TCP/IP communication above the adaptation layer. The primary motivation behind the 6LowPAN project was to connect all IP-based devices easily to other IP networks without any kind of translation gateways or proxies and to have a fully compatible and connected IoT world.

2.4 IEEE 802.15.4 Standard

IEEE 802.15.4 [8] is a short-range wireless technology that aims to enable applications with relatively low throughput and latency requirements within wireless personal area networks (PANs). The task group 4 of the IEEE 802.15 working group published the first version of the standard in 2003. A number of revisions were issued later in 2006 and 2011 [3]. The main features include low cost, low complexity, low data rate and most importantly, low-power consumption. Moreover, the IEEE 802.15.4 protocol [1] has been specifically developed to address an increasing demand for short-range communications between the nodes in low-power sensor networks and is supported by a large number of academic and commercial sensor nodes.

The core goal of the standard was to define the physical (PHY) and Medium Access Control (MAC) layer of the network stack for low-power WSNs. As shown in Figure 2.2, the IEEE 802.15.4 standard has lower data rates, lower complexity, lower

cost and lower power consumption than most of the other standards. It has approximate data rates of 250 Kb/s, 40 Kb/s and 20 Kb/s depending on the application type and it supports multiple Radio Frequency (RF) bands and digital modulation mechanisms, which can be seen in Table 2.1.

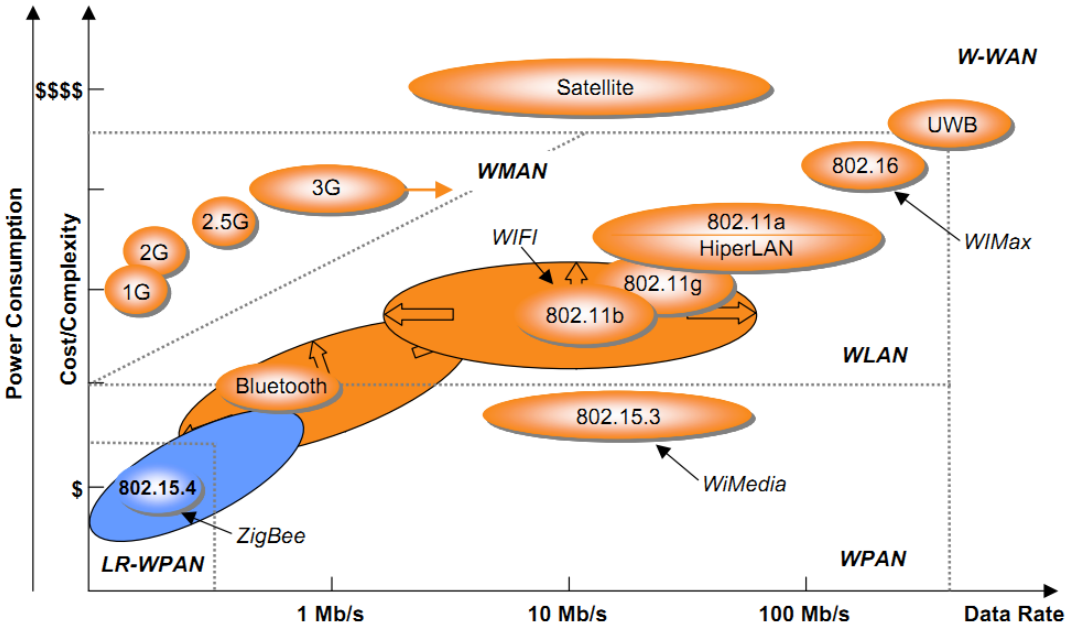

FIGURE 2.2: A map of wireless standards and their key features [9].

Freq. Band	Region	Channel num.	Modulation	Bit rate
(868-868.8) MHz	EU, Japan	1	BPSK	20 kb/s
(902-928) MHz	USA	10	BPSK	40 kb/s
(2400-2483.5) MHz	Worldwide	16	O-QPSK	250 kb/s

TABLE 2.1: IEEE 802.15.4 frequency bands and data rates.

The communication range of the protocol is about 10 meters [8]. Moreover, both star and mesh-based topologies are supported by the standard. In the star topology networks, a central node named PAN coordinator plays the role of a PAN controller where other devices can only communicate though it. However, mesh/peer-to-peer topology nodes can communicate with any other node within their radio range. The protocol uses different mechanisms to access the channel such as Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA). Moreover, the protocol has four types of MAC frames: data frame, beacon frame, acknowledgement frame and MAC command frame. As shown in Figure 2.3, the packet structure of IEEE 802.15.4 consists of four main fields: a preamble of four octets (32 bits) used for synchronization, a packet ID acting as a start of packet delimiter (8 bits), a PHY header which contains the Physical layer Service Data Unit (PSDU) length (8 bits), and finally a PSDU field which contains the actual data.

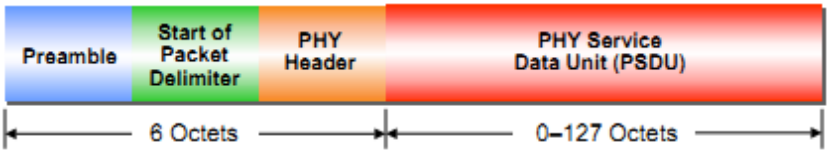

FIGURE 2.3: A general structure of IEEE 802.15.4 packet [9].

Finally, there are two network device classes of the 802.15.4 standard: full-function devices (FFDs) and reduced-function devices (RFDs). An FFD has the capability to operate in three different modes: it can act as a PAN coordinator, a simple coordinator, or a device. As a PAN coordinator, it can communicate with any other node in the network, and thus it has higher processing power than other devices in the network. However, an RFD has limited processing power enough to enable its communication with a single FFD at most at any instant.

2.5 Network Stacks in Contiki

Contiki [10] is an open source, lightweight, and multi-tasking operating system built with memory-constrained networked embedded systems and wireless sensor networks in mind. However, one of its main issues is the lack of proper documentation apart from its source code comments and examples which made it challenging to acquire sufficient information about its network stack.

Three types of network stacks [11] can be used in Contiki: µIP's (IPv4 and IPv6) stacks and rime stack. The uIP stack is a concise implementation of the TCP/IP network suite which provides IPv4 networking capabilities and later on, it was extended to provide IPv6 capabilities. The rime stack offers a set of custom networking primitives to enable communication for low-power wireless networks using lightweight layering and the ability to build complex abstractions.
Contiki adopts a five-layer network stack which is roughly similar to the TCP/IP model but simpler considering the computation and memory constraints of most networked embedded systems. At the same time, it also covers the traditional seven layers of the Open Systems Interconnection (OSI) model as shown in Figure 2.4. The description of each layer is briefly explained in the following subsections.

2.5.1 Radio Layer

The radio or physical layer is the first layer at the bottom of the Contiki model. This layer defines how the input data are structured and built to be transmitted to the upper layers of the network. When the data arrive via interrupt handlers in bytes or as a full packet, they are copied into packet buffers. Furthermore, the data in these packet buffers are ready to be sent to the upper layers of the network through a polling process.

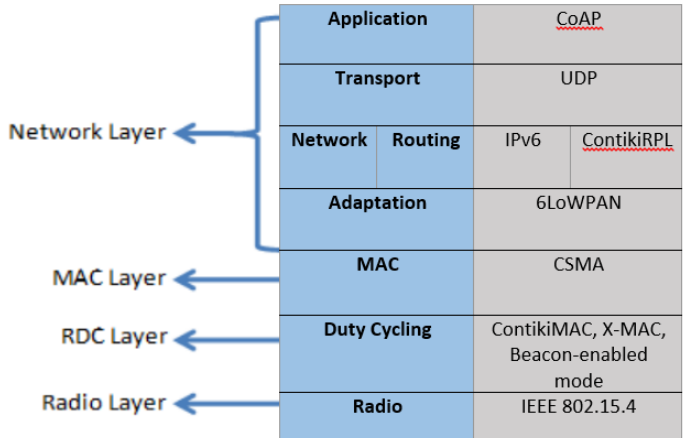

FIGURE 2.4: Contiki network stack and its protocol examples.

2.5.2 Framer Layer

The framer layer is not shown in the provided netstack figure but it is located between the physical/radio layer and the RDC layer. The framer layer does not have a regular layer implementation like the rest of the layers, as it consists of a set of auxiliary functions which are used for either creating a frame with data to be transmitted or parsing the frame's data upon reception. There are two types of framer layer that can be used in Contiki: *framer-nullmac.c* and *framer-802154.c*.

2.5.3 RDC Layer

The Radio Duty Cycling (RDC) layer plays a crucial role in the Contiki netstack as it significantly determines the energy consumption of the nodes by allowing the nodes to turn their radio transceiver off as long as they are not in use and making sure their radio transceivers are awake during packet reception.

Currently, Contiki offers three defined RDC protocols [11]: LPP, X-MAC and ContikiMAC. The LPP protocol was developed based on the original Low-Power Probing protocol while improving the power consumption at the same time. Contiki's X-MAC is similarly based on the X-MAC protocol while improving certain networking and power usage aspects. Finally, ContikiMAC was developed to enhance the low-power listening mechanisms used by the subsequent RDC protocols while improving the energy efficiency at the same time.

While the aforementioned protocols are already defined to be instantly used, Contiki offers the ability to implement a new RDC mechanism and evaluate its performance, power consumption and network capabilities. Therefore, the goal of this study is to show the analysis of using a protocol which is compliant with the IEEE standards such as the beacon-enabled mode of IEEE 802.15.4 and prove that it has similar or lower power consumption than the already implemented duty cycling protocols such as ContikiMAC and X-MAC which are not standard-compliant.

2.5.4 MAC Layer

The Medium Access Control (MAC) layer resides on top of the RDC layer. It also plays a vital part of the Contiki netstack as it defines how the nodes can communicate when the network is congested. The MAC layer is responsible for avoiding collisions and retransmitting packets in case of collisions. Contiki provides two MAC protocols to use: Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and nullmac. CSMA provides various functionalities such as sensing the channel/radio medium before transmitting to back-off if another node is transmitting, waiting for a specific time depending on the RDC protocol used, and retransmitting the dropped packets during collisions. Alternatively, nullmac does not offer any MAC-level processing, as it only forwards the packets from the radio driver to the upper layer and vice versa, and thus it has potentially more packet loss ratio than CSMA.

2.5.5 Network Layer

The network layer is the topmost layer in Contiki where it covers various sub-layer tasks as shown in Figure 2.4. It is mainly responsible for preparing the packets before they are sent. In other words, it provides different networking functionalities and routing the received packets while adapting these packet frames to match the upper sub-layers format such as IPv6 before they are sent to other nodes. The routing protocol used in Contiki is RPL (Routing Protocol for Low-power and lossy networks). It is responsible for finding the optimal route, in which the transmitted packets can take by forming a routing acyclic graph starting from the root node which is called Destination Oriented Directed Acyclic Graph (DODAG). Finally, the last two uppermost sub-layers are transport and application sub-layers. A transport sub-layer protocol such as User Datagram Protocol (UDP) defines the way of communication between source and destination nodes. At the top, the application sub-layer acts as an interface between host applications and lower layers and vice versa. One of the current protocols in the application sub-layer is the IETF Constrained Application Protocol (CoAP) which is a low-power program implementation that tries to leverage any generic duty cycling protocol and to achieve low-energy consumption.

2.5.6 A note on the Contiki layerization scheme

While most of the current MAC protocols cover both of Contiki's RDC and MAC layers, they aim to dynamically adapt the network's efficiency and energy consumption according to the current traffic [12]. ContikiOS developers have decided to separate the MAC layer into two layers for simplicity. However, this separation adds more unnecessary complexity to the MAC protocol development [13]. For example, ContikiMAC protocol acts only as an RDC protocol which have to be used with a MAC protocol such as CSMA or nullmac protocol which adds more testing scenarios to analyze which MAC driver can be utilized with it to achieve an optimal performance.

Furthermore, the lack of sufficient documentation about the system as a whole plays a key role in making it difficult to understand the design decisions behind various implementations within its network stack.

2.6 802.15.4 Beacon- and Non Beacon-Enabled Modes

The IEEE 802.15.4 MAC protocol provides two operating mechanisms: non beacon-enabled and beacon-enabled modes which are explained in Figure 2.5. In the non beacon-enabled mode, the nodes use the unslotted Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism to access the channel and a Clear Channel Assessment (CCA) is executed before using the channel. In a beacon-enabled network, the basic medium access is the slotted CSMA/CA mechanism where a superframe structure is managed by the PAN coordinator for synchronization and network management functionalities. While devices that use the beacon-enabled mode are more complex than non beacon-enabled devices [6], the beacon-enabled networks' support of synchronization and duty cycling to preserve the nodes' energy consumption make them more favourable to use considering their design complexity trade-off at the same time.

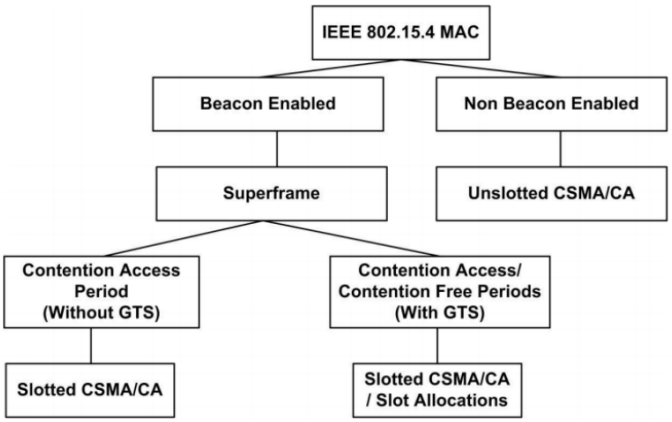

FIGURE 2.5: IEEE 802.15.4 operation modes [3].

2.6.1 IEEE 802.15.4 Non Beacon-Enabled Mode

As mentioned earlier, the non beacon-enabled (unsynchronized) mode uses the classical (unslottted) CSMA/CA mechanism where each time a node wants to access the channel, it waits for a random backoff duration and senses the channel to transmit the data only if the channel is free; otherwise, the node waits for another random period before trying to sense the channel again.

2.6.2 IEEE 802.15.4 Beacon-Enabled Mode

In the beacon-enabled mode or synchronized mode, the PAN coordinator periodically broadcasts a train of beacon signals (superframes) with information that manages and synchronizes the nodes' communications within the network. The main purpose of the synchronization is to enable organized channel access of the slotted CSMA/CA mechanism adapted. Therefore, whenever a device wants to transmit, it performs a random backoff before sensing the channel. If there is no channel activity, the node waits until the next slot before sensing the channel again where no activity has been detected for two successive slots after the initial random backoff period. If there is an activity within the channel, the backoff procedure is repeated as mentioned earlier. The core difference in the non beacon-enabled mode lies in nodes' ability to access the channel instantly if there is no channel activity detected after the first backoff duration which clearly shows the unsynchronized nature of this mechanism.

Algorithm Description

According to the IEEE 802.15.4 MAC standards for the beacon-enable mode [3], a superframe is used in order to access the channel where it is initiated by the PAN coordinator. The superframe structure, which is shown in Figure 2.6, is divided into the following three parts: 1) an inactive part; 2) a Contention Access Period (CAP), where nodes can utilize the slotted CSMA/CA mechanism; and 3) a Contention Free Period (CFP), which contains a number of time slots called Guaranteed Time Slots (GTSs) which are allocated by the PAN coordinator to access the channel without any contention. Each GTS contains one or more slots within the CFP part in a superframe. CFP and CAP together constitute the active part of the superframe which is divided into 16 time slots. The PAN coordinator can allocate up to seven GTSs out of the 16 time slots while the rest of GTSes are reserved for contention-based access. However, there is no communication activity during the inactive part, so the nodes are able to switch off their radio transceivers and activate the power saving mode to save the network's energy consumption.

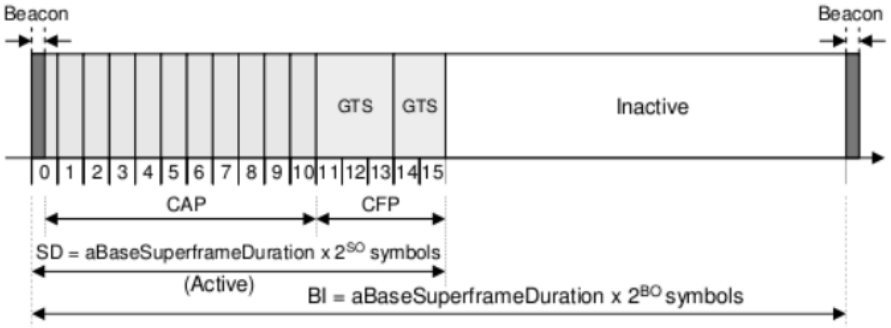

FIGURE 2.6: IEEE 802.15.4 superframe structure [6].

As shown in Figure 2.6, a superframe period is the duration between two consecutive beacons. The timing of the superframe is determined by these two parameters:

- MAC Superframe Order (SO), which determines the active part duration of the superframe called superframe duration (SD) as shown in equation 2.2.
- MAC Beacon Order (BO), which defines the whole superframe duration called Beacon Interval (BI) as shown in equation 2.3.

The values of SO and BO for the beacon-enabled mode are defined in the standards as follows: $0 \leq SO \leq BO \leq 14$. However, in the non-beacon enabled mode, the $macBeaconOrder$ BO = 15, meaning that the timeout value is just a simple backoff. Moreover, the parameter $aBaseSuperframeDuration$ is a standards-defined constant which is equal to 960 symbols, with a symbol time of 16 microseconds.

$$SD = aBaseSuperframeDuration * 2^{SO} \tag{2.2}$$

$$BI = aBaseSuperframeDuration * 2^{BO} \tag{2.3}$$

The inactive part is very essential in saving energy as the nodes sleep during this period of the superframe, the Duty Cycle (DC) of nodes depends only on the superframe structure as shown in equation 2.4, where Inactivity Order (IO) is equal to the difference between the superframe order and the beacon order $IO = SO - BO$.

$$DC = \frac{SD}{BI} = 2^{IO} \tag{2.4}$$

Data transfer mechanisms in the beacon-enabled mode

Two different data transfer modes can be used within the beacon-enabled mode to achieve the communication between a PAN coordinator and its nodes using low-power capabilities. The *direct transfer* mode is used in the case a device wants to send a packet to the coordinator. In direct mode, the device initially waits for the synchronization beacon frame from the coordinator. Moreover, it starts transmitting the data using one of the superframe slots and the coordinator sends an optional acknowledgement upon receiving the packet and end this communication process. The *indirect transfer* mode is used when the PAN coordinator tries to send a packet to one of its associated devices. Initially, the coordinator sends a beacon frame to the destination node. Furthermore, after the node receives the beacon frame, it sends a data request frame and upon receiving the data request, the coordinator sends the data packet and an acknowledgement of the data request sent by the device. At the end, an acknowledgement is sent to the coordinator to end the transmission process.

Implementation Challenges

The beacon-enabled networks have more complex design and time-constrained requirements compared to non beacon-enabled networks. However, the low-power

consumption advantages account for this level of complexity in contrast to the inefficient contentious listening of non beacon-enabled devices. Moreover, the implementation of a standard-compliant protocol that should support a wide range of platforms has a number of challenges. Initially, the implementation should be platform independent and portable to most of the IEEE 802.15.4-supported platforms which is hard considering the different hardware dependencies of each platform. Therefore, the implementation should be implemented in a real-time operating system (RTOS) to satisfy the time-critical operations of the beacon-enabled mode which is still challenging to implement. The other solution is to shift these time-critical operations to the physical (PHY) drivers layer in order to accurately meet these timing requirements. At the same time, the different hardware dependencies of each platform add another layer of complexity and put more restrictions and challenges during the implementation development.

2.7 IEEE 802.15.4e Standard

While the IEEE 802.15.4 standard was initially published in 2003, other revisions and improvements for different aspects of the standard were introduced in 2006 and 2011 [3]. The first version of the standard defined the bottom two layers of the network stack, namely PHY and MAC layers for low-power, low-rate and low-cost WPANs and introduced two operating modes: non beacon-enabled mode and beacon enabled mode. Furthermore, a new amendment to the 802.15.4-2006/2011 revisions was published in 2012 titled IEEE 802.15.4e [14] in order to enhance the previous MAC protocols and communication modes while addressing the emerging needs of time-critical embedded applications in industrial environments at the same time. It introduced a number of general functional improvements and the following new MAC behaviour modes:

- AMCA (Asynchronous Multi-Channel Adaptation) for infrastructure monitoring networks.
- DSME (Deterministic and Synchronous Multi-channel Extension) for deterministic latency and scalability requirements.
- LLDN (Low Latency Deterministic Network) for high reliability and low latency.
- TSCH (Time-Slotted Channel Hopping) for high throughput requirements, bounded latency, and high reliability.

Furthermore, the standard provided a number of general functional improvements [15] in various aspects such as low energy, enhanced beacons, multipurpose frame, fast association and multiple MAC performance metrics. Out of the new MAC behaviour modes mentioned earlier, the related ones to beacon and non beacon-enabled modes are DSME and TSCH behaviour modes.

DSME enhances the performance of beacon-enabled networks where all the devices are synchronized with the DSME coordinator through enhanced beacons using a multi-superframe structure (improvement to 802.15.4 superframe). The multi-superframe consists of a number of repeated superframes which contain the normal CAP and CFP periods used in the beacon-enabled mode. Moreover, the multi-superframe enhances the 802.15.4 GTS mechanism to support more channels by grouping a number of superframes and extending their single channel operation to multi-channel ones through adaptive switching channels or hopping channels according to certain parameters such as the link quality. Nevertheless, the TSCH protocol is the MAC behaviour mode of interest to this study which is explained thoroughly in the next subsection.

2.7.1 TSCH MAC Behaviour Mode

The Time Slotted Channel Hopping (TSCH) MAC protocol [15] is one of the prominent IEEE 802.15.4e MAC behaviour modes which was developed to satisfy industrial and vehicular sectors. The TSCH behaviour mode combines time slotted access (that was previously defined in the beacon-enabled mode), multichannel communication and channel hopping which suits multi-hop networks in particular. In addition, TSCH supports star, tree and partial/full mesh network topologies. Dedicated and shared links are supported by TSCH in which the latter represents special communication slots that can be assigned to more than one transmitter, and thus enabling concurrent access by multiple nodes at the same time. The protocol's core goals are to support larger network capacity, predictable latency and most importantly to achieve high reliability and low-power consumption through time slotted access mechanism. TSCH supports multichannel based on channel hopping through 16 different channels which are defined by *channelOffset* which is an integer value that ranges from 0 to 15. Multichannel communication allows more nodes to communicate at the same time (timeslot) using different channels which are identified by their channel offset.

Slotframe structure

The traditional 802.15.4 superframe structure is replaced by a slotframe structure which consists of a number of timeslots of 10ms duration typically. The nodes use the periodic slotframe for synchronization through their timeslots. Each timeslot enables a pair of devices to exchange either a maximum-size data frame or acknowledgement for this frame through the duration of the timeslot. If the acknowledgement is not received, a retransmission of that frame is deferred until the next assigned transmit timeslot for the same (sender-receiver) devices. The Absolute Slot Number (ASN) is the total number of slots elapsed since the network was deployed. A TSCH link is defined as a pairwise assignment of a directed communication between devices in a specific timeslot on a specific channel offset. Therefore, a link

between two nodes is denoted by [n, *channelOffset*] and the frequency f that is used for communication in timeslot n of the slotframe is derived in the following equation.

$$f = F[(ASN + channelOffset\%N_{ch})] \ , \tag{2.5}$$

Where N_{ch} is the number of channels in use and F is the lookup table function containing the sets of available channels.

Channel hopping

Equation 2.5 represents the channel hopping mechanism in TSCH, where multiple frequencies can be returned for the same link at different timeslots. Hence, channel hopping enables mitigating the effects of interference and multipath fading by ensuring that all the available channels are used for a specific link during their specified timeslot, and thus improving the network reliability.

TSCH scheduling

Regarding network scheduling, the IEEE 802.15.4e standard [14] does not specify how the communication schedule is built, optimized and maintained. It only explains the mechanism of how the MAC layer can execute the schedule. Initially, a number of scheduling techniques were proposed such as centralized scheduling and distributed scheduling. Centralized scheduling assigns a manager node that is responsible for building and optimizing the network schedule while the nodes regularly updates the manager node with their neighbours list and the data size transmitted/received. However, the nodes which use distributed scheduling have no central entity and take decisions locally based on which links to schedule with their list of neighbours. Alternatively, a number of studies in the literature proposed new scheduling mechanisms to enhance the network performance and its energy efficiency.

Compliance with the IEEE 802.15.4 standard

The TSCH MAC behaviour mode does not completely amend the physical layer structure. In other words, it can operate on any hardware that is compliant with the previous IEEE 802.15.4 standard which is really crucial for a sustainable development of the standard without spending too much resources during the research cycle.

2.8 Software Overview

A lot of software and operating systems have been introduced to address different requirements in the WSN field. This section describes the software and toolchain used to accomplish the practical work and evaluate the required implementations.

2.8.1 Contiki Operating System

The operating system (OS) in a wireless sensor network differs from the traditional operating systems used in personal computers. It is a relatively smaller piece of software that enables basic programming abstractions on embedded systems such as sensor nodes to be utilized by application developers. A wide range of WSN operating systems are available for usage and testing such as RIOT, OpenWSN, TinyOS and Contiki.

ContikiOS [10] is a lightweight WSN operating system designed for resource constrained platforms. Contiki does not fully support Real-Time OS (RTOS) functionalities, as it uses a hybrid model to combine the advantages of event-driven processes and preemptive threads. Contiki introduced *protothreads* [16] which provide event-driven services while enabling optional preemptive multithreading through an application library that can be linked only with the applications that require this preemptive feature explicitly in a specific process.

2.8.2 COOJA Simulator

Cooja [17] is a Java-based simulator designed for simulating the sensors of a WSN that run ContikiOS. It provides a set of functionalities to track the performance of the sensor nodes. A number of applications are provided along with the simulator such as MSPsim device emulator, mobility plug-ins and powertrace tool. MSPsim emulator can be used through Cooja to provide emulation of the sensor nodes such as Tmote Sky and Zolertia Z1 based on the MSP430 microcontroller. The simulated motes in Cooja have three essential properties: data memory containing the program code required to test, mote type which can be shared between multiple motes while using the same source code, and hardware peripherals.

2.8.3 Powertrace Profiling tool

Powertrace [18] is a run-time and network-level power profiling tool for low-power wireless sensors. It utilizes power state tracking mechanism in order to roughly estimate the mote's power usage and store their energy consumption in duty cycling entities called energy capsules. The powertrace tool is included within ContikiOS set of applications and requires no additonal hardware for profiling. The tool showed through multiple experimentations to have a 94% accuracy in estimating the power consumption compared to hardware-based power measurements [18]. The tool was used to empirically evaluate the power consumption of the MAC behaviour mode

implementations that run on a number of emulated motes in Cooja and estimate the expected network's lifetime.

2.8.4 MSP430-GCC toolchain

MSP430-GCC is an open-source C/C++ compiler toolchain designed to help flashing and debugging the embedded source code on MSP430-based microcontrollers. The toolchain normally is periodically updated in order to provide better optimizations such as modifying the platform's memory footprints whether ROM, RAM or other peripherals in order to accommodate larger bin files (generated code) in the platform's memory.

Chapter 3

Literature Review

In this chapter, the latest contributions and developments achieved on different parts and modes of the IEEE 802.15.4 protocol are discussed along with the core challenges and constraints as well as an analysis of the protocol. Although the main focus of the literature review is on the beacon-enabled mode of IEEE 802.15.4 MAC layer, a number of different IEEE 802.15.4 studies are presented due to two main reasons: the shortage of studies done on the beacon-enabled mode specifically, and the indirect relation of these studies on the beacon-enabled networks as a whole.
Finally, a motivation of this study is discussed mentioning the reasoning behind choosing this topic.

3.1 Previous work on IEEE 802.15.4 MAC protocols analysis

Previous studies in the literature have covered different aspects of IEEE 802.15.4 networks. The studies vary from performance evaluations and analysis of the protocol to comparative assessments with other protocols using practical measurements, simulations and/or analytical models.

On the one hand, a performance evaluation of the IEEE 802.15.4 MAC protocol - *partial aspects of the protocol or as a whole* - in which simulations and/or practical experimentations were used is discussed in [19–25]. On the other hand, analytical and/or mathematical models were used to theoretically analyze and evaluate certain aspects of the protocol's performance in [26–29]. However, a number of the aforementioned studies provided simulation results along with mathematical models to validate its accuracy.

3.1.1 Practical-based studies on the IEEE 802.15.4 beacon-enabled mode

A new adaptive duty cycle algorithm for beacon-enabled networks is presented in [19]. The algorithm tries to efficiently exploit the network traffic to adjust the duty cycle dynamically for minimum energy consumption. The simulation was done using OMNeT++ (an extensible and component-based C++ network simulator) customized for IEEE 802.15.4 networks. The authors optimized their energy-efficient algorithm for star topology beacon-enabled networks where an extensive reduction in energy consumption can be seen when compared with other recently proposed

algorithms and studies such as AAOD, DSAA, DCA and IEEE 802.15.4.
The measurements and simulations done in [24] aim to pave the way for integration of future applications with the IEEE 802.15.4 technology and analyze any potential interference by other wireless technologies operating in the same band such as IEEE 802.11 and Bluetooth. The simulation was done using ns-2 simulator where it was calibrated using the measurements done earlier for providing a more realistic simulation and a reliable way to evaluate the IEEE 802.15.4 MAC protocol.

The authors in [20] proved through comparative experiments of the IEEE 802.15.4 /ZigBee and 6LoWPAN protocols that the IEEE 802.15.4/ZigBee protocol can support smaller duty cycles and minimize the maximum end-to-end delay compared to 6LowPAN. Nonetheless, the results also showed that the 6LoWPAN protocol can provide smaller mean end-to-end delays and lower packet loss rates, hence enhancing the network reliability. The experimentation was done after a theoretical analysis of the low-power characteristics of both protocols had been done in order to tune the low-power related parameters and configurations of both protocols.

While most of the preceding studies explored the protocol through measurements and simulations, the study by Li-chun et al. [6] focuses on the design and implementation of 802.15.4 beacon-enabled devices. The authors introduced a priority-based scheduling kernel to handle time-critical beacon events for beacon-enabled devices. However, the evaluation of their implementation was very abstract and lacked sufficient simulation results to support their implementation.

3.1.2 Analytical-based studies on the IEEE 802.15.4 beacon-enabled mode

A detailed analytical evaluation of IEEE 802.15.4 with (CSMA/CA) MAC scheme in a star topology is presented in [28] where they investigated whether the low-power design constraints of WSN are met by using this MAC scheme or not. The model is used to estimate the energy consumption and data rate of 802.15.4 networks where both saturated and unsaturated periodic traffic scenarios are considered.

A more in-depth study provided a mathematical model for the beacon-enabled mode of the IEEE 802.15.4 MAC protocol [26], where both star and tree-based topologies are inspected. Moreover, a validation of the model is provided through simulation using a dedicated C simulation tool. Finally, a suitable comparison between the simulation results is provided in which tree-based topologies performed better in terms of packet success probability but with larger delay than their respective star-based topologies.

The analytical model provided in [27] accommodates a general traffic distribution to capture the behaviour of the IEEE 802.15.4 low-power mode. Furthermore, a simulation of the model was done using ns-2 simulator where it shows a number of performance evaluation insights that could be utilized for the deployment of low-power IEEE 802.15.4/ZigBee networks.

3.2 Previous work on the IEEE 802.15.4e TSCH mode

The TSCH MAC behaviour mode has received strong consideration and interest among the WSN research community after the IEEE 802.15.4e standard [14] had been released in 2012. The new standard introduced various MAC behaviour modes in order to meet the emerging needs of different fields and markets. After the TSCH protocol was introduced, a number of studies in [30–34] were done on the protocol to evaluate and analyze its performance and/or provide different mechanisms to enhance different aspects of it or build on top of its MAC layer.

Domenico et al. in [30] evaluated the performance of TSCH nodes which use TSCH shared links by providing an analytical model of the new TSCH CSMA-CA algorithm using discrete time markov chains. The performance metrics include packet delivery ratio, latency and network power consumption. Furthermore, their model was validated through simulations and experimentations using cc2420 radio-based Tmote-sky nodes. Their evaluation showed that the CSMA-CA parameter values strongly determine the algorithm's performance. Moreover, it was noticed that the network performance can be improved in some cases using channel hopping.

The TSCH CSMA-CA mode was evaluated and compared with the non beacon-enabled mode with unslotted CSMA-CA of IEEE802.15.4 in [32]. The comparison was based on an analytical markov chain model for the TSCH mode in terms of packet loss rate, throughput and energy consumption. The analytical results proved that the TSCH mode can provide a better deterministic access, lower energy consumption and more network capacity compared to the IEEE 802.15.4 non-beacon enabled mode.

The authors in [33] compared the performance of the IEEE 802.15.4e behaviour modes TSCH and DSME in terms of their energy consumption, throughput and delay. The comparison model used was based on an energy consumption model which was previously developed for another low-power design platform. The results acquired from the analytical model showed that TSCH performs better than DSME when low duty cycles are used in terms of network delay and throughput. However, DSME is slightly more energy efficient during data transmission due to the more complicated synchronization mechanism used by TSCH.

Since the IEEE 802.15.4e standard did not define how a TSCH schedule is built and optimized, the authors in [34] proposed two new schedulers which focuses on energy efficiency using a centralized node (gateway) to allocate frequency and timeslots. An energy consumption model was derived from a TSCH node initially to determine the expected network lifetime. The first scheduler which is called low-complexity energy-efficient scheduler has low computational complexity for node management and optimization. Nevertheless, the second scheduler which is called Vogel's Approximation Method Heuristic Scheduling Algorithm is more complicated computationally to address the pitfalls of the nodes' greedy allocation. The

two schedulers were compared with the Round Robin Scheduler (RRS) in terms of packet delivery ratio and power consumption. Their results proved that both schedules were more energy efficient than RRS while providing a better throughput at the same time.

3.3 Motivation of the topic

One of the core goals of most WSN applications, especially the ones with resource constrained requirements is to save their power consumption as much as possible. There are various techniques to exploit the power consumption of the wireless sensors either at the hardware level or the software level. The main objective of this implementation is to exploit the power consumption at the software level of a wireless sensor by trying to demonstrate the effects of changing the MAC and RDC protocols used in Contiki network stack. To the best of our knowledge, the most energy-efficient MAC protocol used in Contiki currently is ContikiMAC [35]. However, the ContikiMAC protocol is not compliant with the current IEEE standards, so it has low portability to various standard compliant platforms. Therefore, this explains the need for another MAC protocol which is complaint with the standards and has at least the same energy efficiency of ContikiMAC or better. The beacon-enabled mode of IEEE 802.15.4 is one good candidate MAC protocol mode to be utilized within the Contiki network stack. The reasoning behind choosing the beacon-enabled mode is that it is standard-compliant which would potentially ease the development of a platform independent implementation. Moreover, it has low-power mechanisms using its low duty cycling techniques during the inactive part of its superframe structure which is explained in Chapter 2.

According to the previous work discussed earlier, there is no study that introduced an implementation for the beacon-enabled mode of the IEEE 802.15.4 standard on ContikiOS or a comparative assessment between the beacon-enabled mode and the current MAC and RDC protocols in the Contiki network stack.

After months of work on porting the beacon-enabled mode into Contiki netstack, a number of blocking issues and limitations (discussed in chapter 4) hindered the development process of the protocol. As a result, the evaluation direction was changed to the TSCH MAC behaviour mode which was introduced in the IEEE 802.15.4e-2012 standard. The TSCH protocol mitigates the core issues faced when using the MAC behaviour modes of the previous IEEE 802.15.4-2006 standard such as the beacon-enabled mode. The limitations of the beacon-enabled mode include reliability problems mainly due to the CSMA/CA algorithm to avoid collisions. Furthermore, the periodic beacons which are sent to synchronize the channel access increase the contention, and thus increase the collision probability and the overall network latency. All the previous limitations are mitigated in the TSCH mode in addition to introducing new features such as multichannel communication and channel hopping in order to alleviate interference and multi-path fading of the network.

Chapter 4

Implementation

This chapter discusses the hardware and software implementations and configurations in order to elaborate and accomplish the objectives of the thesis work. The initial goal of the implementation was to test an existing implementation of the beacon-enabled mode of the IEEE 802.15.4 standards and try porting it to the Contiki network stack along with the configurations needed for the testbed architecture. However, due to some limitations and blocking issues, the research work is shifted to the new IEEE 802.15.4e TSCH MAC protocol. Finally, a number of simulations are done to assess and evaluate the RDC and MAC protocols under investigation within ContikiOS. Additionally, the results of the simulation are shown along with the major concluding remarks based on this analysis.

4.1 Hardware Implementation

The hardware setup and experiments are explained in this section along with the configuration and toolchain required to test the software implementation on the testbed. In the beginning, a number of simulation experiments were done to emulate the hardware provided in order to analyze its characteristics and behaviour in contrast with other platforms.

4.1.1 Hardware and toolchain setup

The hardware setup which was planned to be used for testing and evaluating the software's implementations and the toolchain required to test the hardware are described in the following subsections.

CC2650STK SensorTag

The provided testbed for the experiment is a Systems-on-Chip(SoC) developed by Texas Instruments based on the low-power CC2650 wireless MicroController Unit (MCU). The sensorTag provides support to multiple wireless standards such as Bluetooth Low Energy (BLE), 6LowPAN and ZigBee. The CC2650 wireless MCU [36] is equipped with a 2.4 GHz RF transceiver, which can be easily configured to operate either BLE 4.2 or IEEE 802.15.4 PHY and MAC layers. Moreover, ten low-power

sensors are provided within the sensorTag collecting distinct analog and digital information. The main processor used is a 32-bit ARM Cortex-M3 with processing power of 48 MHz that operates the low-power sensor controller and has 20 kB RAM and 128 kB of flash memory. However, a separate ARM Cortex-M0 processor powers both the BLE controller and the IEEE 802.15.4 MAC protocol which are embedded into a Read-Only Memory (ROM) chip. For debugging and burning new code images, a 2-pin cJTAG and JTAG (Joint Test Action Group) debugging standards are provided.

SensorTag Debugger DevPack

The debugger development package is a debugging hardware that enables debugging the applications executed on the CC2650 sensortag. It consists of a small XDS110 JTAG debugger with a Universal Serial Bus (USB) connection to make sure the sensortag does not power off during the debugging process.

SmartRF06EB

The SmartRF06EB is a full development kit which contains all the hardware required to evaluate and debug the CC2650 sensortag. It consists of a motherboard designed for testing radio performance and software development, a 128x64 LCD screen, an XDS100vs debugger and a set of peripherals and sensors that enable testing various functionalities of the sensortag.

SmartRF Flash Programmer 2

SmartRF Flash Programmer 2 is a windows-based software which can be used with the aforementioned debuggers XDS110 and XDS100Vs to program the flash memory of the CC2650 wireless MCU of the sensortag board. The software includes a graphical interface and a command-line interface in order to analyze the process of flashing the target and track its success.

InstantContiki

InstantContiki is a linux-based ubuntu virtual machine equipped with the required compiler and toolchain configurations to develop and test Contiki applications. The current version (V 3.0) contains all the updated toolchain packages needed to build multiple targets such as the CC2650 MCU provided. Moreover, it also has the Cooja simulation software installed which is used to simulate a specific Contiki application in small or large networks by emulating different types of sensor nodes.

4.1.2 Blocking Issues

In the beginning, we tried simulating the sensortag nodes using the Cooja WSN simulator by running a simple Contiki example to analyze the network activity and

nodes/motes performance. However, the simulation motes for the CC2650 sensortag are not supported yet in Cooja. There was another suggestion to use emul8 framework [37] which is an open-source embedded systems emulator for different processor architectures such as ARM Cortex-M that is used in the CC2650 sensortag. Unfortunately, the emulator supports older Cortex-M platforms and running the current Contiki binaries for the CC2650 sensortag did not work successfully.
The next solution was to start testing using hardware directly by burning specific Contiki example binaries into the CC2650 sensortag MCU and debug the application to analyze and evaluate its performance.

Flashing the CC2650 MCU can be done in two ways, using either the SensorTag Debugger DevPack or the SmartRF06EB development kit. The cc26xx drivers are provided in the Contiki platform drivers' folder to create the bin file related to each platform. Furthermore, the *SmartRF Flash programmer 2* software is used to flash the generated bin file to the CC2650 MCU using any of the aforementioned debuggers connected to its JTAG pins. However, the main problem is that the required hardware for debugging the SensorTag board was not available during the experiment. Therefore, the remaining solution was to try the simulation using native or AVR architecture platforms which are supported in Contiki and to do a number of microbenchmark comparisons between the architecture used and the CC2650 architecture in order to obtain simulation results which are relative to the sensortag node.

4.2 Software Implementation

This section discusses the core challenges and constraints of the software implementation especially for resources constrained systems like sensor nodes in WSN. Moreover, it explains different software implementations for the IEEE 802.15.4 MAC layer and then summarises the discussion with an analysis of the mentioned implementations and the reasoning behind moving to the IEEE 802.15.4e TSCH MAC protocol.

4.2.1 Software Design Constraints

The software development process usually starts with setting a set of design constraints that need to be met at the end of the project. In order to implement an IEEE 802.15.4 MAC protocol stack, a number of challenges and design constraints have to be resolved to achieve an implementation which is complaint with the standards and an architecture that can be ported to different platforms.
Modularity is one key design constraint that should to be met, considering the huge amount of features provided by 802.15.4 MAC protocol. In other words, the implementation should be modular by allowing the selection of only specific subset of functionalities and customize it according to the platform resource constraints and application requirements.

Another constraint that should be checked for the MAC protocol implementation is portability or platform independence. Considering the platform meets the IEEE 802.15.4 standards [3] precision and accuracy requirements (62.5 kHz and ± 40 ppm for the 2.4 GHz frequency band) and contains a compliant radio chip, the implementation should be portable to any platform with these characteristics. Finally, the MAC implementation should provide extensibility to enable integrating new features, extensions and/or modifications to the standard as it is continuously evolving to match the new requirements and challenges that could emerge.

4.2.2 Porting current 802.15.4 MAC implementations to Contiki

A number of software implementations for the IEEE 802.15.4 MAC layer are currently available. However, none of the implementations are currently supported in Contiki. As a result, to ensure not wasting time and effort in reinventing the wheel, it was easier to try porting one of the existing implementations to Contiki. Then, we can assess the performance of at least the beacon-enabled mode individually and compare it with current Contiki RDC and MAC protocols. Initially, these implementations were evaluated to define the trade-offs and which version is the most suitable version to be ported to Contiki network stack.

The following four software implementations are investigated: open-ZB, TKN15.4, OpenMAC and TIMAC which are explained concisely in the next part.

Open-ZB

Open-ZB [38] is an open-source implementation of the IEEE 802.15.4/ZigBee network stack for TinyOS [39] WSN operating system version 1.1.15. It has been developed by a research unit from CISTER (Research Centre in Real-Time and Embedded Computing Systems), in collaboration with the TinyOS 15.4 and ZigBee working groups. The implementation is currently supporting two hardware platforms: the MICAZ and the Tmote sky motes. Open-ZB's architecture is monolithic, which means the entire MAC implementation is done within a single component.

There has been a number of revisions published later on to add support to the missing features such as security mechanisms and the GTS mode. However, PAN coordinator management is not implemented yet, so the implementation scope is only limited to RFD devices or clients. The Open-ZB implementation comprises three main components: the PHY layer which contains the required auxiliary functions and drivers to control the hardware, the MAC layer which provides the main functionalities of the protocol, and network (NWL) layer where high level functions such as ZigBee addressing schemes and routing of the nodes occur.

TKN15.4

TKN15.4 [40] is a platform independent and open-source implementation of the IEEE 802.15.4-2006 MAC layer. It has been developed at the Technical University of Berlin. The implementation is written in nesC programming language for the 2.1 version of TinyOS. One of nesC main features is producing component-based and event-driven software, which enables modularity and reusability in other architectures. Although the development is tested only on the TelosB/Tmote Sky sensor node, the implementation is platform independent which enables the protocol to be used on any platform that has a compatible execution environment.

Since TinyOS does not fully support real-time system functionalities, a number of functions and time-critical tasks have been moved from the MAC layer down to the PHY/radio layer of the hardware platform to meet the tight timing constraints in beacon-enabled networks. However, these modifications increases hardware dependencies at the same time. Moreover, the Guaranteed Time Slots (GTS) services were not supported at first, but this feature was implemented later in 2011. One of the main features of TKN15.4 implementation is being able to assign device roles (FFD or RFD) at runtime on software level. Thus, it improves the application's flexibility to adapt the network nodes according to any potential changes, but it also add more complexity on the software level which may not satisfy certain type of very resource constrained platforms.

MeshNetics OpenMAC

OpenMAC [41] is another TinyOS-based and open-source implementation of the IEEE 802.15.4 MAC protocol by MeshNetics. The project was developed to complement their ZigBee stack implementation providing basic wireless networking in star and peer-to-peer network topologies. The implementation supports various AVR-based platforms such as Atmel's RF development kit (RZ502) and their own ZigBit and MeshBean2 development board. One of the main features of the project resides in providing an easy to use C Application Programming Interface (API) to developers who are not familiar with nesC programming language of TinyOS. However, the project's documentation and source code page are no longer supported by the MeshNetics group as they were acquired by Atmel Corporation. Moreover, the implementation comprises only the non beacon-enabled mode and provides limited security functionalities.

TIMAC

The TIMAC implementation [42] is a software stack developed by Texas Instruments to support their IEEE 802.15.4-compliant transceivers and System-on-Chips (SoS) such as the CC2630 wireless MCU. The implementation is standard-compliant with the IEEE 802.15.4-2006 standard. It supports beacon-enabled and non beacon-enabled modes of IEEE 802.15.4 and provides network security features. Since TIMAC is a commercial implementation, it is only available for download as object code. Therefore, it is not possible to port this implementation to Contiki or even analyze the internal software layer structure. Another implementation is provided by NXP [43] for the IEEE 802.15.4 MAC protocol stack, but the source code is not available for download as well due to the copyrighting nature of these implementations.

4.2.3 Analysis of the implementations

Out of the four implementations discussed earlier, the TKN15.4 software architecture is chosen to be ported to Contiki. The main reasoning behind choosing it is based on an analytical study made by Basmer et al. [44] which is explained in the following points:

- The static code analysis done by Basmer et al. showed that TKN15.4 is the most runtime efficient implementation. The profiling analysis tools measured the runtime of a set of MAC operations where TKN15.4 had the shortest runtime excluding association request response operation which is faster in the Open-ZB implementation.
- The Open-ZB implementation only covers network clients with partial support to network coordinators which limits its functionalities and range of applications. Their RFD implementation is missing the activation and deactivation

of the radio module while their FFD implementation does not support active scanning for beacon requests.

- TKN15.4 provides an elegant solution to mitigate the time-critical operations of the beacon-enabled mode by shifting these operation from the MAC layer down to the radio drivers layer as they operate in a timing constrained context through interrupt service routines.
- Although the TKN15.4 implementation is not certified yet according to the investigation conducted on this regard, the research work done by Basmer et al. shows that TKN15.4 meets all the mandatory and optional functionalities required by FFD and RFD devices in the standard. In other words, this means that TKN15.4 is theoretically standard-compliant but requires a set of interoperability practical tests with certified MAC implementations to practically prove its standard compliance.

4.2.4 Porting TKN15.4 to Contiki

The process of porting the TKN15.4 implementation to ContikiOS is a challenging and time consuming task as the language used by TinyOS in their TKN15.4 implementation is nesC which requires a lot of changes to be translated to C programming language used by Contiki. The only possible solution was to manually translate the nesC application source code to Contiki compatible C code which requires huge amount of time and effort for just reinventing the wheel of translating the source code. Nevertheless, Lahiru [45] provided a translator project that can automate the porting process from nesC to C which that can be compiled successfully in Contiki. The translation process entails different procedures such as lexical analysis of the input source code to a stream of tokens, then parsing these tokens to generate a tree parser that can be finally used for code generation.

Unfortunately, this project covered only a small subset of the whole nesC grammar since it was done as a thesis project with the aim of achieving higher translation accuracy rather than examining the efficiency of the translation. Therefore, it is not possible to use this tool for the porting process.

A number of manual experiments were done to port the existing implementation to Contiki, though it did not produce any fruitful results which have lead the research direction to test different and newer IEEE 802.15.4e MAC protocols for a number of reasons that are explained briefly in the next subsection.

4.2.5 Transitioning to the IEEE 802.15.4e TSCH mode

There are a number of challenges encountered while porting the beacon-enabled mode. However, the most critical issue is that even after successfully porting the implementation on ContikiOS. A set of hardware dependencies have to be resolved only through hardware, since the beacon-enabled mode entails some operations that

have to be accurately timed. This problem can not be solved in software by ContikiOS, as it does not fully support real time operating system functionalities.
During the research work done on the beacon-enabled mode, it is noticed the majority of the recent studies have shifted their focus to evaluate and analyze the novel MAC behaviour modes introduced in the new amendment to 802.15.4-2006/2011 standards named IEEE 802.15.4e-2012. The standard [14] introduced a number of behaviour modes to alleviate the performance and reliability problems encountered throughout using the older beacon-enabled and non beacon-enabled modes.

The Time-Slotted Channel Hopping (TSCH) protocol was introduced to provide high reliability and low-power consumption to a diverse number of industrial applications and to ensure predictable network performance. It combines the old mechanism time slotted access used in the late 802.15.4 modes with multi-channel and channel hopping functionalities. Therefore, it provides predictable latency, communication reliability and energy-efficient performance.

4.3 Simulations

This section discusses the main configuration and setup required to conduct the simulations using ContikiOS in order to evaluate the implementations under investigation and to test them using the emulated motes.

4.3.1 Simulation Setup

In the simulation, Zolertia Z1 mote [46] is considered as the emulated testbed of the experiment. A total of 11 nodes are emulated in cooja where node 1 acts as the PAN coordinator and the remaining 10 nodes (Z1- 2:11) act as clients or devices in a star topology. Each simulation is run for only ten minutes due to time and resource constraints, in addition to testing several simulation scenarios. The application example that was tested on the motes is a modified TSCH implementation [47] for ContikiOS based on the IPv6 network stack and the RPL routing protocol for transmitting UDP packets. Nevertheless, another application is provided to assess the TSCH evaluation process. The other example is a similar IPv6 and RPL implementation which uses ContikiMAC as its RDC protocol. The simulation motes are randomly distributed with a Unit Disk Graph Medium (UDGM) radio medium in an area of 100m by 100m as shown in Figure 4.1.

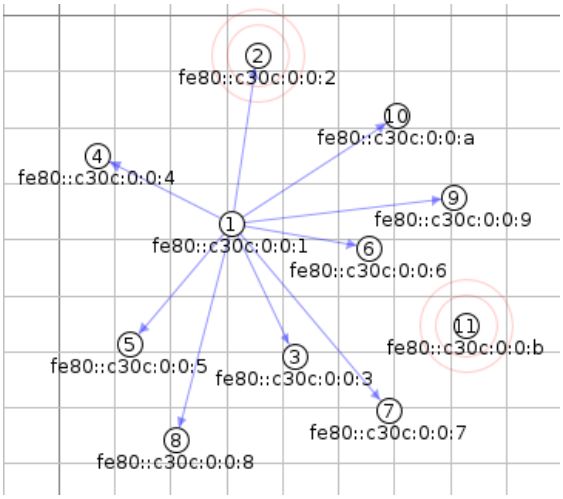

FIGURE 4.1: Network structure of the simulation experiment in Cooja

4.3.2 TSCH implementation in Contiki

The TSCH Contiki implementation was originally tested on multiple testbeds such as Tmote Sky, CC2650 and Zolertia zoul while some of the testbeds are successfully emulated in Cooja. Initially, there has been a number of issues when trying to flash the bin file to the emulated motes such as Tmote Sky and Zolteria z1. The main problem was caused by an insufficient ROM size which is required for flashing the software implementation. However, it turned out that the *MSP430-GCC* toolchain

needed to be updated to at least *V.4.7.0* in addition to modifying certain dependencies. Finally, after making the changes discussed earlier, the implementation was flashed to the emulated Z1 motes successfully.

The *rpl-tsch* example was initially tested using cooja motes to analyze its working scenarios. Furthermore, after fixing the dependencies problems and updating the toolchain, it was tested using Z1 motes as shown in Figure 4.1. There are two implementation examples which are tested: *rpl-tsch* implementation which uses IPV6 and RPL upper layers and a simple *rime-tsch* implementation that uses Contiki's default network stack *RIME*.

4.3.3 Configuration of Contiki network stack

Contiki netstack (network stack) can be configured in two ways, either through modifying the project's Makefile to use the desired protocols, or by creating *project-conf.h* within the project directory with all the required configuration and referencing it in the Makefile. Otherwise, the default network stack Rime will be used.

One of the advantages of the TSCH implementation is that it is fully independent from the upper layers that exist above the MAC layer, meaning that it can be tested using different network topologies with the least amount of changes. In the MAC layer of the implementation, an enhanced TSCH CSMA-CA mechanism is used along with the remaining features of the TSCH behaviour mode. Moreover, The lower layers of the configured netstack which is shown in Table 4.1 show that the *nordc_driver* RDC protocol is used, because the radio duty cycling mechanisms is handled in the *tschmac_driver* protocol, specifically in the *tsch-slot-opertion* files. Thus, there is no need to use Contiki's RDC protocols such as *ContikiMAC* or *X-MAC* which will only add unnecessary overhead.

TSCH netstack	Protocol
MAC	*tschmac_driver*
RDC	*nordc_driver*
Framer	*framer_802154e*

TABLE 4.1: The lower layers of Contiki netstack used in the *TSCH* example

4.3.4 Evaluation Parameters

The simulations evaluate two aspects of the TSCH behaviour mode implementations. First, profiling the network power consumption of the PAN coordinator and the nodes of a TSCH-based network. The second aspect of the evaluation is measuring the power consumed during different activities such as packet transmission/reception of the TSCH MAC behaviour mode. Furthermore, another RDC protocol implementation is evaluated which is explained in the next subsection.

4.3.5 Comparison with ContikiMAC

In order to evaluate the energy efficiency of the TSCH behaviour mode protocol, ContikiMAC is used as a comparison parameter for the experiment. To the best of our knowledge, ContikiMAC is currently the most energy-efficient RDC protocol in Contiki combined with CSMA as a MAC protocol [35]. The ContikiMAC implementation uses the same simulation setup mentioned earlier. Moreover, The ContikiMAC implementation under test is similar to the TSCH implementation in the sense that they both have the same higher layer topology. In other words, IPv6 network topology is utilized in this implementation along with RPL protocol for routing and UDP transport protocol on top of the RPL. It is worth mentioning again the difference between the default MAC protocols used outside contiki network stack and within the Contiki network stack. ContikiMAC is implemented by Contiki to be utilized in their RDC sub-layer along with CSMA in the MAC layer. Alternatively, the TSCH mode is implemented as a MAC behaviour mode protocol which is responsible for the tasks of both RDC and MAC layers, thus it does not use RDC protocol within Contiki netstack. Therefore, the energy efficiency assessment can be quantified based on the radio duty cycling mechanism by both of the protocols TSCH and ContikiMAC while the rest of the sub-layers of the Contiki netstack are the same.

ContikiMAC netstack	**Protocol**
MAC	*csma_driver*
RDC	*contikimac_driver*
Framer	*framer_802154*

TABLE 4.2: The lower layers of Contiki netstack used in the *ContikiMAC* example

4.4 Results

This section demonstrates the results of the different simulation set-ups that were discussed in the previous section. In addition, it also discusses the software tools and configurations used in order to generate these results.

4.4.1 TSCH packet frame sequence

The frame sequence of the devices and the PAN coordinator of a TSCH network along with its different message data types are described in the following part.

Frame sequence of the PAN coordinator

Cooja simulator has a number of tools that aim to help visualizing the network traffic of the sensor nodes during a particular simulation. Cooja's log listener is a visualization tool that prints all the packets transmitted or received by the different nodes of the network. For example, during the initiation process in the TSCH example, the first node is assigned as the PAN coordinator of the network where it starts sending enhanced beacon frames to the rest of the nodes in the network to manage their communication. The initiation frames of the PAN coordinator can be viewed in Figure 4.2, which shows the sequence of the messages to acknowledge their role and IPv6 address in the network.

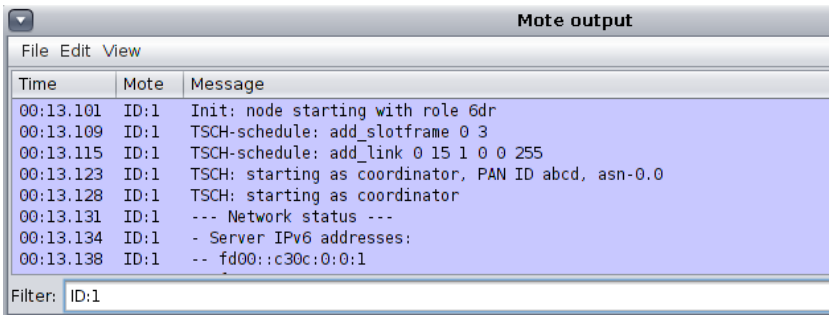

FIGURE 4.2: Packet initiation sequence of the network coordinator

Frame sequence of the nodes

In the beginning of a TSCH joining procedure, the nodes start the communication by advertising their role and IPv6 address. Their default route is initially set to none until they receive the association information packets such as slotframe, links and multi-hopping data. The slotframe mechanism is fully interrupt driven, as it wakes up the node only during its dedicated active slot and link configuration.

Figure 4.3 shows a snippet of the packets transmitted or received by node 2. For example, after node 2 receives a packet from node 6, another packet is advertised mentioning the current state of the node. This packet mentions a number of data

such as the *asn* which shows the total number of elapsed slots from the network initiation. In addition, the links attached to the node and the channel number used are mentioned in this packet.

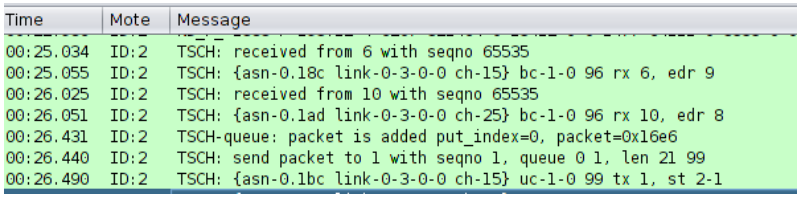

FIGURE 4.3: A set of transmit/receive communication packet sequence of node 2

4.4.2 Power Consumption analysis of a TSCH network

ContikiOS provides a set of applications to evaluate the performance of the nodes used during the simulations using Cooja. The powertrace profiling tool is used to track the node's power consumption values by tracking its power states to estimate its power consumption during different activities and store in energy capsules. In other words, these values represent the energy consumption for a specific cycle by the transceiver and MCU during different power states (e.g. transmit, receive, idle).

The power consumption data are saved using Cooja's log-listener to a log file. A snippet of the data printed to the log-listener can be shown in Figure A.1 of Appendix A. Then, the log file is parsed with a number of MATLAB scripts in order to calculate and plot the power consumption values after extracting the duty cycle values from the time capsules of the powertrace profiling tool.

The power consumption values are plotted using MATLAB in Figure 4.4 which shows the power consumption of the PAN coordinator (node 1) during different power states such as normal mode (CPU), transmission (Tx), reception (Rx) and low-power mode (idling) (LPM). The power consumption values are shown for the whole ten-minute duration of the simulation. For example, the average power consumption of the PAN coordinator node during normal operations (CPU) is around 0.089 mW. Moreover, it is noticed that the power consumption is the highest during Rx mode (around 1 mW), which is mainly due to the continuous listening nature of the PAN coordinator to manage and synchronize the communication of the nodes.

Furthermore, another MATLAB script is used to plot the power consumption data of the ten z1 motes (nodes 2:11) which communicate with the PAN coordinator of the network. Figure A.2 in Appendix A shows the average power consumption during different power states of the nodes after the ten-minute duration of the simulation.

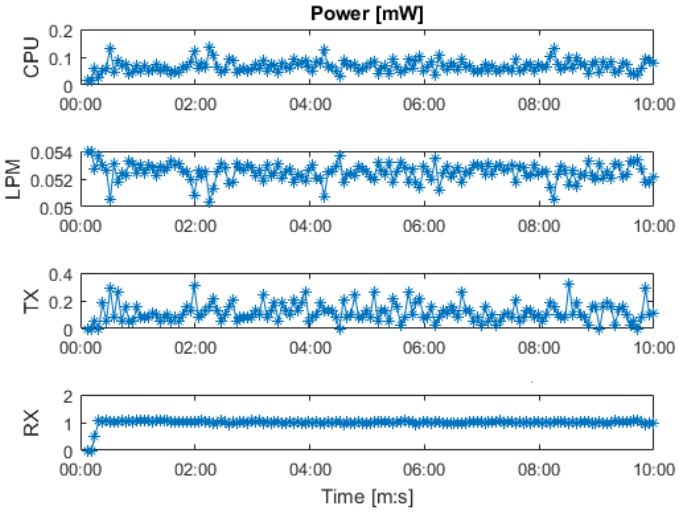

FIGURE 4.4: Power consumption analysis of the PAN coordinator in the TSCH example (X axis represents time in minutes:seconds, Y axis represents the power consumed in mW by the Z1 mote for CPU, LPM (low power mode/idle), Tx and Rx)

4.4.3 Power Consumption analysis of a ContikiMAC network

Similarly, the exact simulation setup is used to measure the power consumption of a network that uses ContikiMAC protocol instead of the TSCH mode. The same network topology is used with the main differences (shown in Table 4.2) in the RDC, MAC and framer sub-layers of the Contiki netstack. The main reasoning behind using the same higher network topology is to try to analyze the effect of changing the protocol under test only, and thus provide an accurate evaluation of these protocols.

Figure 4.5 shows the power consumption values of the server node during the same power states mentioned earlier for the same simulation period. Furthermore, it is noticed from the analysis that the average power consumption is around 0.79 mW during the server's normal CPU state which is slightly less than the TSCH network coordinator's power consumption. However, the average power consumption of the ContikiMAC server during reception (around 1.97 mW) is higher than the TSCH coordinator's power consumption during the same power state.

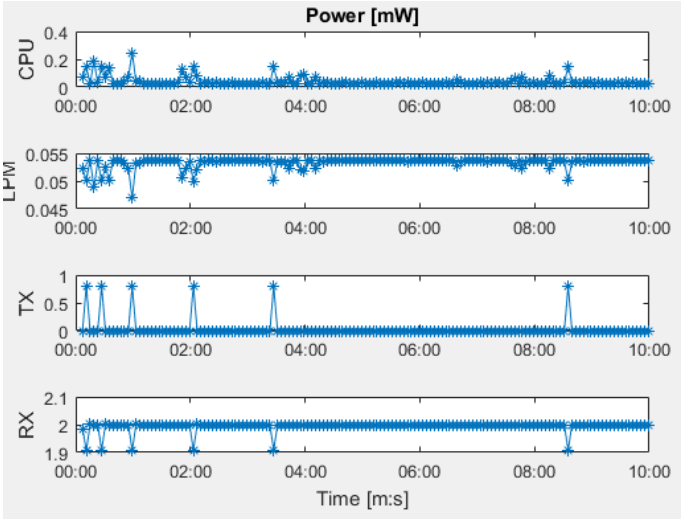

FIGURE 4.5: Power consumption analysis of the server in the ContikiMAC example (X axis represents time in minutes:seconds, Y axis represents the power consumed in mW by the Z1 mote for CPU, LPM (low power mode/idle), Tx and Rx)

4.4.4 Accuracy of the simulation results

Since the experiments done are only using simulations which take into account certain parameters and dependencies while using limited resources at the same time. The results of the simulations are not completely accurate and no final conclusions should be taken from these results. Moreover, since the Cooja simulation software does not take fading effects into account [17], the energy consumption values of the sensor nodes are not completely accurate. Therefore, real testbed experiments are required to fully evaluate the power consumption of the protocols and to be able to provide more concrete conclusions from this analysis.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The field of energy efficiency in wireless sensor networks is rapidly evolving. During the last few years, a number of hardware optimization and software algorithms have been introduced to improve the power consumption of the wireless sensors. Pure hardware optimizations can significantly decrease the power consumption of a sensor platform. Nevertheless, the software implementation will run eventually on different hardware platforms, and thus software optimizations can significantly enhance the overall system power consumption. Furthermore, the sensor's power consumption is mainly dependent on the activation period of its wireless radio transceiver. Additionally, the MAC layer of the network stack is the layer responsible for managing this activation mechanism. Therefore, the MAC protocols of wireless networks play a crucial role in preserving the power consumption of the hardware platform of a wireless sensor.

In this thesis, three different MAC protocols were evaluated in order to analyze their energy efficiency in a wireless sensor network. The wireless operating system Contiki was used to test and evaluate the software implementations. The three MAC behaviour mode protocols were the beacon-enabled mode, the TSCH mode and the ContikiMAC protocol. The beacon-enabled mode was only theoretically analyzed due to difficulties in porting the implementation in ContikiOS. However, the TSCH mode and ContikiMAC were evaluated through simulations using Cooja simulator.

Finally, we used the powertrace tool in order to retrieve the power consumption values of the emulated motes using different simulation configurations. The results show that the TSCH mode improves the power consumption of the sensor nodes in normal mode compared to ContikiMAC. However, ContikiMAC can still efficiently manage the power consumption of the nodes during transmission and reception when compared with the TSCH mode. The results imply that further real-life experimentation needs to be done in order to accurately evaluate the energy efficiency of the protocols under test.

5.2 Future Work

The significance of the research on this topic will increase as soon as more accurate practical results are provided. This thesis tries to pave the way for future work that can support the development of energy-efficient MAC protocols for wireless sensor networks. Therefore, future work includes validating the simulation results by testing the software implementations on a real platform. Furthermore, other aspects of the wireless network life cycle should be taken into account such as network discovery, synchronization schemes and scheduling mechanisms. Another aspect of the future work should analyze the network performance while the nodes are mobile using different mobility models.

Appendix A

Energy Measurements

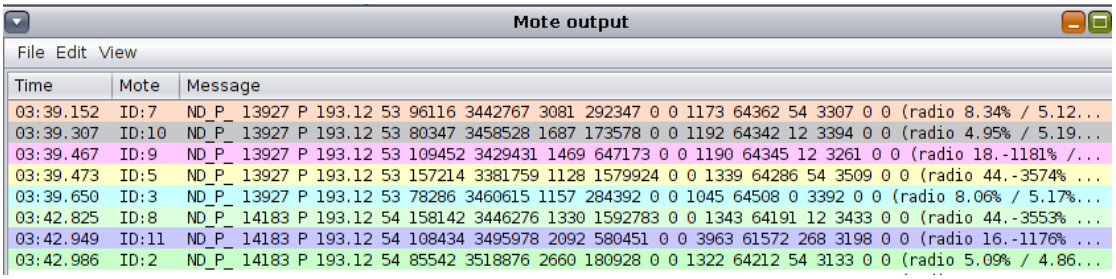

FIGURE A.1: A snippet of the power consumption values printed by powertrace tool

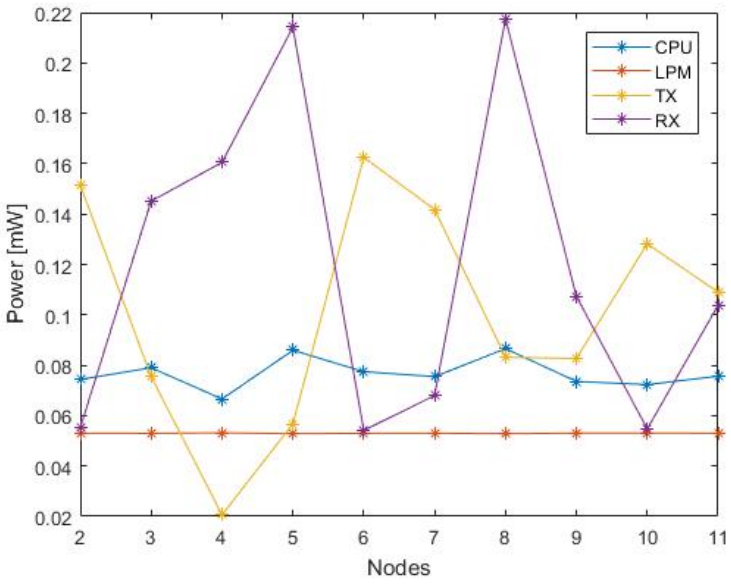

FIGURE A.2: Power consumption analysis of the nodes in the TSCH example during different power activities (X axis represents the nodes from 2 to 11, Y axis represents the power consumed in mW by the Z1 motes for CPU, LPM (Low Power Mode), Tx and Rx)

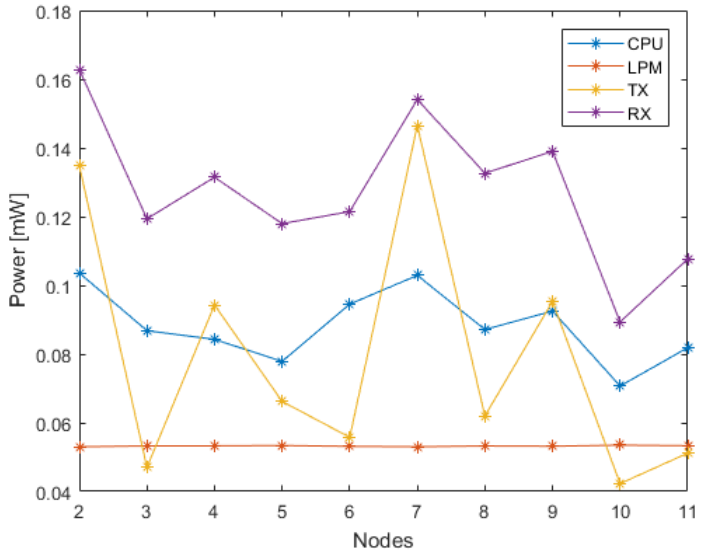

FIGURE A.3: Power consumption analysis of the clients in the ContikiMAC example during different power activities (X axis represents the nodes from 2 to 11, Y axis represents the power consumed in mW by the Z1 motes for CPU, LPM (Low Power Mode), Tx and Rx)

Bibliography

[1] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010.

[2] Matthew Gast. *802.11 wireless networks: the definitive guide*. " O'Reilly Media, Inc.", 2005.

[3] IEEE Computer Society. *"IEEE Standard for Local and Metropolitan Area Networks-Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specification for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. IEEE std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006). Sept. 2011, pp. 1–314.

[4] Kazem Sohraby, Daniel Minoli, and Taieb Znati. *Wireless sensor networks: technology, protocols, and applications*. John Wiley & Sons, 2007.

[5] Wendi Rabiner Heinzelman et al. "Energy-scalable algorithms and protocols for wireless microsensor networks". In: *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. Vol. 6. IEEE. 2000, pp. 3722–3725.

[6] Li-chun Ko, Yung-chih Liu, and Hua-wei Fang. "Design and implementation of IEEE 802.15. 4 beacon-enabled network devices". In: *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*. IEEE. 2006, 5–pp.

[7] Zach Shelby and Carsten Bormann. *6LoWPAN: The wireless embedded Internet*. Vol. 43. John Wiley & Sons, 2011.

[8] Jose A Gutierrez et al. "IEEE 802.15. 4: A developing standard for low-power low-cost wireless personal area networks". In: *IEEE network* 15.5 (2001), pp. 12–19.

[9] Jose A Gutierrez. "IEEE Std. 802.15. 4. Enabling Pervasive Wireless Sensor Networks". In: *Eaton Corp., Berkeley Uni* (2005).

[10] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. "Contiki-a lightweight and flexible operating system for tiny networked sensors". In: *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE. 2004, pp. 455–462.

[11] *Network Stack - Contiki. [Online]*. URL: http://anrg.usc.edu/contiki/index.php/Network_Stack (Accessed: 15 Sept. 2017).

[12] Michael Buettner et al. “X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks”. In: *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM. 2006, pp. 307–320.

[13] Kévin Roussel and Ye-Qiong Song. “A critical analysis of Contiki’s network stack for integrating new MAC protocols”. PhD thesis. INRIA Nancy, 2013.

[14] IEEE Computer Society. *IEEE Standard 802.15.4e, Part 15.4: Low-Rate Wireless Personal Area Networks (LRWPANs) Amendament 1: MAC sublayer*. IEEE std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4™-2011). Apr. 2012, pp. 1–225.

[15] Domenico De Guglielmo, Giuseppe Anastasi, and Alessio Seghetti. “From IEEE 802.15. 4 to IEEE 802.15. 4e: A step towards the internet of things”. In: *Advances onto the Internet of Things*. Springer, 2014, pp. 135–152.

[16] Adam Dunkels et al. “Protothreads: simplifying event-driven programming of memory-constrained embedded systems”. In: *Proceedings of the 4th international conference on Embedded networked sensor systems*. Acm. 2006, pp. 29–42.

[17] Fredrik Osterlind et al. “Cross-level sensor network simulation with cooja”. In: *Local computer networks, proceedings 2006 31st IEEE conference on*. IEEE. 2006, pp. 641–648.

[18] Adam Dunkels et al. “Powertrace: Network-level power profiling for low-power wireless networks”. In: Swedish Institute of Computer Science, 2011.

[19] Hadi Rasouli, Yousef S Kavian, and Habib F Rashvand. “ADCA: Adaptive duty cycle algorithm for energy efficient IEEE 802.15. 4 beacon-enabled wireless sensor networks”. In: *IEEE sensors journal* 14.11 (2014), pp. 3893–3902.

[20] Emanuele Toscano and Lucia Lo Bello. “Comparative assessments of IEEE 802.15. 4/ZigBee and 6LoWPAN for low-power industrial WSNs in realistic scenarios”. In: *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*. IEEE. 2012, pp. 115–124.

[21] Changsu Suh, Zeeshan Hameed Mir, and Young-Bae Ko. “Design and implementation of enhanced IEEE 802.15. 4 for supporting multimedia service in Wireless Sensor Networks”. In: *Computer Networks* 52.13 (2008), pp. 2568–2581.

[22] Jianliang Zheng and Myung J Lee. “A comprehensive performance study of IEEE 802.15. 4”. In: *Sensor network operations* (2006), pp. 218–237.

[23] Mikko Kohvakka et al. “Performance analysis of IEEE 802.15. 4 and ZigBee for large-scale wireless sensor network applications”. In: *Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*. ACM. 2006, pp. 48–57.

[24] Marina Petrova et al. “Performance study of IEEE 802.15. 4 using measurements and simulations”. In: *Wireless communications and networking conference, 2006. WCNC 2006. IEEE*. Vol. 1. IEEE. 2006, pp. 487–492.

[25] Gang Lu, Bhaskar Krishnamachari, and Cauligi S Raghavendra. "Performance evaluation of the IEEE 802.15. 4 MAC for low-rate low-power wireless networks". In: *Performance, Computing, and Communications, 2004 IEEE International Conference on*. IEEE. 2004, pp. 701–706.

[26] Chiara Buratti. "Performance analysis of IEEE 802.15. 4 beacon-enabled mode". In: *IEEE Transactions on Vehicular Technology* 59.4 (2010), pp. 2031–2045.

[27] Yu-Kai Huang, Ai-Chun Pang, and Hui-Nien Hung. "A comprehensive analysis of low-power operation for beacon-enabled IEEE 802.15. 4 wireless networks". In: *IEEE Transactions on Wireless Communications* 8.11 (2009).

[28] Sofie Pollin et al. "Performance analysis of slotted carrier sense IEEE 802.15. 4 medium access layer". In: *IEEE Transactions on wireless communications* 7.9 (2008).

[29] Tae Rim Park et al. "Throughput and energy consumption analysis of IEEE 802.15. 4 slotted CSMA/CA". In: *Electronics Letters* 41.18 (2005), pp. 1017–1019.

[30] Domenico De Guglielmo et al. "Analysis and experimental evaluation of IEEE 802.15. 4e TSCH CSMA-CA algorithm". In: *IEEE Transactions on Vehicular Technology* 66.2 (2017), pp. 1573–1588.

[31] Simon Duquennoy et al. "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation". In: *IEEE DCOSS* (2017).

[32] Shuguang Chen et al. "Performance Analysis of IEEE 802.15. 4e Time Slotted Channel Hopping for Low-Rate Wireless Networks." In: *KSII Transactions on Internet & Information Systems* 7.1 (2013).

[33] Iacob Juc et al. "Energy Consumption and Performance of IEEE 802.15. 4e TSCH and DSME". In: *Wireless Communications and Networking Conference (WCNC), 2016 IEEE*. IEEE. 2016, pp. 1–7.

[34] Mike Ojo et al. "An energy efficient centralized scheduling scheme in TSCH networks". In: *Communications Workshops (ICC Workshops), 2017 IEEE International Conference on*. IEEE. 2017, pp. 570–575.

[35] Adam Dunkels. "The contikimac radio duty cycling protocol". In: (2011).

[36] Texas Instruments. "CC2650 SimpleLink Multistandard Wireless MCU". In: *CC2650 SWRS15* 52 (2015).

[37] *Emul8: The open source embedded systems emulation framework*. URL: https://github.com/emul8/emul8 (Accessed: 13 Sept. 2017).

[38] André Cunha et al. "Open-ZB: an open-source implementation of the IEEE 802.15. 4/ZigBee protocol stack on TinyOS". In: *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE International Conference on*. IEEE. 2007, pp. 1–12.

[39] Philip Levis et al. "TinyOS: An operating system for sensor networks". In: *Ambient intelligence* 35 (2005), pp. 115–148.

[40] Jan-Hinrich Hauer. "TKN15.4: An IEEE 802.15. 4 MAC implementation for TinyOS". In: *Citeseer* (2009).

[41] MeshNetics. *OpenMAC Open-source project by MeshNetics (Archived)*. URL: http://web.archive.org/web/20110322050211/http://www.meshnetics.com:80/opensource/mac/ (Accessed: 20 Sept. 2017).

[42] Texas Instruments. *802.15. 4 MAC Application Programming Interface, Version: 1.5*. Tech. rep. 2009.

[43] NXP. *IEEE 802.15.4 NXP Stack User Guide v2.6*. Tech. rep. 2016.

[44] Thomas Basmer, Henry Schomann, and Steffen Peter. "Implementation analysis of the IEEE 802.15. 4 MAC for wireless sensor networks". In: *Mobile and Wireless Networking (iCOST), 2011 International Conference on Selected Topics in*. IEEE. 2011, pp. 7–12.

[45] LN Wickramasinghe. "Porting nesC Application to Contiki". In: (2013).

[46] Zolertia. *Z1 Datasheet*. URL: http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf (Accessed: 25 Sept. 2017).

[47] Atis Elsts Simon Duquennoy Beshr Al Nahas. *IEEE 802.15.4-2015 TSCH and IETF 6TiSCH Contiki Implementation*. URL: https://github.com/contiki-os/contiki/tree/master/core/net/mac/tsch (Accessed: 29 Sept. 2017).