FACULTY OF ENGINEERING
AND ARCHITECTURE

Sustainable and Interoperable MAC Protocol Design for Heterogeneous Internet of Things Systems

Jan Bauwens

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Information Engineering Technology

Supervisors

Prof. Eli De Poorter, PhD - Prof. Ingrid Moerman, PhD

Department of Information Technology
Faculty of Engineering and Architecture, Ghent University

September 2021

GHENT
UNIVERSITY

FACULTY OF ENGINEERING
AND ARCHITECTURE

Sustainable and Interoperable MAC Protocol Design for Heterogeneous Internet of Things Systems

Jan Bauwens

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Information Engineering Technology

Supervisors
Prof. Eli De Poorter, PhD - Prof. Ingrid Moerman, PhD

Department of Information Technology
Faculty of Engineering and Architecture, Ghent University

September 2021

GHENT
UNIVERSITY

ISBN 978-94-6355-520-3
NUR 986
Wettelijk depot: D/2021/10.500/68

Members of the Examination Board

Chair

Prof. Filip De Turck, PhD, Ghent University

Other members entitled to vote

Prof. Koen De Bosschere, PhD, Ghent University
Prof. Jeroen Famaey, PhD, Universiteit Antwerpen
Spilios Giannoulis, PhD, Ghent University
Wei Liu, PhD, Ghent University
Thomas Watteyne, PhD, Inria-Paris, France

Supervisors

Prof. Eli De Poorter, PhD, Ghent University
Prof. Ingrid Moerman, PhD, Ghent University

Dankwoord

Toen ik zes jaar geleden mijn doctoraatsonderzoek aanving, realiseerde ik mij nog niet helemaal goed welke uitdagingen mij te wachten stonden. Een doctoraat is immers een traject met veel vallen en opstaan, waar overwinningen en tegenslagen zich op een rap tempo opvolgen. Finaal kan ik enkel concluderen dat deze jaren als doctoraatsstudent tot de meeste interessantste en verrijkende horen in mijn leven.

Zonder de hulp en steun van een heel aantal mensen zou dit proefschrift niet tot stand gekomen zijn. Daarom wil ik graag mijn dank betuigen aan allen die rechtstreeks of onrechtstreeks hebben bijgedragen aan mijn doctoraat. Ik hoop hierbij dat ik niemand in het bijzonder vergeet te vermelden. Allereerst wil ik graag mijn promotoren professor Ingrid Moerman en professor Eli De Poorter bedanken om mij de kans te geven om te starten als onderzoeker bij IDLab. Ik heb het altijd als een eer beschouwd samen met hen te kunnen discussiëren over innoverende oplossingen in het draadloze domein. In het bijzonder wil ik professor Eli De Poorter bedanken voor de dagelijkse opvolging van mijn doctoraat, want voornamelijk dankzij zijn inbreng is dit doctoraat een geheel geworden van kwalitatieve papers. Verder wil ik ook Spilios Giannoulis bedanken om een groot deel van de opvolging van mijn doctoraat op zich te nemen, want hij kwam vaak met nieuwe (praktische) inzichten waar ik vaak nog niet aan gedacht had.

Het was een waar genoegen om in de wireless onderzoeksgroep van IDLab mijn onderzoek te kunnen doen. Deze onderzoeksgroep loopt boordevol met aangename en getalenteerde met wie het een eer was om samen te mogen werken. Ik ben ieder van hen dankbaar voor de tijd die ik met hen kon doorbrengen, maar in het bijzonder wil ik mijn bureaugenoten bedanken: Bart, Pieter, Vincent, Jono, Spilios, Peter, Irfan en Dries. Niet alleen was het heel leuk om met hen allen samen te werken, maar we hebben ook vele plezierige tijden beleefd samen op de bureau.

Ook wil ik graag mijn familie bedanken voor alle steun die ik doorheen mijn hele leven van hen gekregen hebben, die mij toe lieten om al mijn dromen na te streven. Zonder hen zou ik er niet in geslaagd zijn om te staan waar ik nu sta. In het bijzonder wil ik mijn grootvader Wilfried Coquyt bedanken, want ik ben er zeker van dat hij er enorm graag had bij geweest. Ik ben enorm trots dat ik dit avontuur tot een mooi einde heb kunnen brengen met veel steun van deze bijzondere man. Ook al mijn vrienden ben ik enorm dankbaar, want onze vele leuke momenten die we samen deelden zorgden voor de nodige ontspanning doorheen deze uitdagende jaren.

Ten slotte wil ik ook de belangrijkste persoon in mijn leven bedanken: mijn

fantastische vrouw Jolien. Bedankt voor je steun, je liefde, en je luisterende oor die er dagelijkse voor zorgen dat ik elke uitdaging zie zitten. Samen met jou kan ik alles aan.

Gent, September 2021
Jan Bauwens

Table of Contents

List of Figures

List of Tables

List of Acronyms

A

ACK	Acknowledgment
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
AGV	Autonomous Guided Vehicle
API	Application Programming Interface

B

BLE	Bluetooth Low Energy

C

CAGR	Compound Annual Growth Rate
CCA	Clear Channel Assessment
CPU	Central Processing Unit
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance

D

DAC	Digitally to Analog Converter
DCO	Digitally Controlled Oscillator
DTLS	Datagram Transport Layer Security

E

ECC Elliptic-Curve Cryptography

ETSI European Telecommunications Standards Institute

F

FCC Federal Communications Commission

FLL Phase Locked Loop

FPGA Field-Programmable Gate Array

G

GPRAM General Purpose Random-Access Memory

GPS Global Positioning System

H

HAL Hardware Abstraction Layer

HIL Hardware Interface Layer

HMAC Hash-based Message Authentication Code

HPL Hardware Presentation Layer

I

IFS Inter Frame Space

IEEE Institute of Electrical and Electronics Engineers

IoT Internet of Things

ISM Industrial, Scientific and Medical

ISR Interrupt Service Routine

K

kbps kilobit per second

L

LPL Low Power Listening

LPWAN Low Power Wide Area Networks

LPM Low Power Mode

M

M2M Machine to Machine

MAC Medium Access Control

MCU Micro Controller Unit

MCS Modulation and Coding Scheme

MF-TDMA Multi Frequency Time Divison Multiple Access

MTU Maximum Transmission Unit

N

NFC Near Field Communication

O

OFDM Orthogonal Frequency Division Multiplexing

OFDMA Orthogonal Frequency Division Multiple Access

OS Operating System

OSGi Open Services Gateway Initiative

OSI Open Systems Interconnection

P

PHY Physical Layer

PN Process Network

PoE Power over Ethernet

Q

QoS Quality of Service

R

RAM Random-Access Memory

RDC Radio Duty Cycling

RF Radio Frequency

ROM Read-Only Memory

RPL Routing Protocol for Low-Power and Lossy Networks

RSSI Received Signal Strength Indicator

RX Receive

S

SDN Software Defined Network

SDR Software Defined Radio

SHA Secure Hash Algorithm

SNMP Simple Network Management Protocol

SME Small and Medium-sized Enterprises

SoC System on Chip

SW Software

SSH Secure SHell

T

TAISC Time Annotated Instruction Set Computer

TCP/IP Transmission Control Protocol/Internet Protocol)

TDoA Time Difference of Arrival

TDMA Time Division Multiple Access

TOF Time Of Flight

TSCH Time Synchronized Channel Hopping

TX Transmit

TWR Two Way Ranging

U

UDP User Datagram Protocol

UPI Unified Programming Interface

UMTS Universal Mobile Telecommunications System

USRP Universal Software Radio Peripheral

UWB Ultra Wide Band

V

VLAN Virtual Local Area Network

VoIP Voice over IP

VPN Virtual Private Network

W

WMP Wireless MAC Processor

WSN Wireless Sensor Network

WuR Wake-Up Radio

X

XML eXtensible Markup Language

Samenvatting
– Summary in Dutch –

Sinds het Internet werd uitgevonden in 1969, is dit steeds een concept gebleven dat de mensheid geboeid houdt. Het werd oorspronkelijk uitgevonden door het Amerikaanse leger om computers met elkaar te verbinden over lange afstanden met behulp van telefoonlijnen. De uitwisseling van datapakketten over wat een pakketgeschakeld netwerk genoemd wordt, vormt de grondlaag waarop het Internet verder is gebouwd. Sindsdien is het Internet geëvolueerd van zijn initiële militaire toepassing, naar de universeel toegankelijke entiteit die we vandaag kennen die het toelaat om eenvoudig te communiceren en informatie uit te wisselen. Een van de belangrijkste concepten van het moderne Internet is zijn veelzijdigheid, want het laat toepassingen toe die veel verder gaan dan wat oorspronkelijk de bedoeling was. Een belangrijke uiting van deze veelzijdigheid was de uitvinding van het Internet der Dingen (Internet of Things, IoT), waarbij - naast mensen en computers - ook allerlei meer alledaagse "dingen" met het Internet verbonden worden. Het gebruik van het woord "ding" is met bedoeling heel vaag, zodat deze omschrijving geen limiterende factor is in het potentieel van IoT. Om toch wat meer duiding te geven, mag je zo een ding definiëren als elk object dat ofwel elementen uit de omgeving kan waarnemen als sensor, of acties op de omgeving kan uitvoeren als actuator. Door de sensor waarden draadloos uit te wisselen kunnen er intelligente beslissingen genomen worden op het vlak van interactie met de omgeving.

Momenteel is IoT zijn weg aan het vinden in verschillende aspecten van het moderne leven, zoals in steden, huizen, de landbouw, de gezondheidszorg, en in de industrie. Dit vergt een grote mate van veelzijdigheid in zowel het opzicht van hardware als software: de toestellen moeten immers klein, goedkoop en energie efficiënt zijn. Om deze voorwaarden te behalen wordt typisch gebruik gemaakt van hardware die minder krachtig is en beperkter is in zijn mogelijkheden. Naast de hardware componenten is er vanzelfsprekend ook software nodig om een degelijke vorm van draadloze communicatie tussen de verschillende toestellen te bereiken. Net zoals bij Wi-Fi netwerken - die typisch gebruik maken van de TCP/IP-software stack - gebruikt IoT een gelijkaardige gelaagde aanpak, maar met een andere set van netwerkprotocollen. Binnen dit doctoraatsonderzoek was er een focus op de medium toegangscontrole (Medium Access Control, MAC) laag, aangezien deze een prominente rol speelt in de performantie van het netwerk. Kort gezegd zal de MAC laag een protocol uitvoeren dat bepaalt hoe en wanneer het gedeelde draadloze medium mag gebruikt worden door de diverse toestellen. Het beoogde doel

hierbij is de kans op botsingen tussen data pakketten te verminderen, terwijl alsnog de performantie (doorvoersnelheid, latentie, levensduur van de batterij, enz.) gewaarborgd moet blijven. Vandaag zijn er al talloze MAC protocollen beschikbaar, die elk optimaal werken in een sub-domein binnen de wereld van IoT. Omdat deze wereld een constante evolutie kent, is er tot nu toe nog geen uitzicht op een allesomvattende MAC oplossing die naadloze connectiviteit levert in alle verschillende draadloze omgevingen en toepassingsdomeinen.

Om toch stappen te zetten naar deze naadloze oplossing, biedt dit proefschrift een aantal antwoorden op open vragen. Het beoogde doel hierbij is het bereiken van duurzaam en interoperabel MAC ontwerp voor heterogene IoT systemen. Meer specifiek werden de volgende uitdagingen nagestreefd:

- Huidige MAC protocol architecturen bieden slechts een beperkte vorm van porteerbaarheid naar verschillende hardware platformen.
- IoT toestellen kunnen niet efficiënt gebruik maken van meerdere radio's.
- Huidgie IoT netwerken hebben slechts een beperkte ondersteuning voor coëxistentie strategieën.
- Er is een gebrek aan afdoende strategieën om toestellen "in het veld" bij te werken.

Om deze uitdagingen aan te gaan, is er binnen dit proefschrift een drieledige aanpak gecreëerd. De volgende drie secties bieden informatie wat er bereikt is binnen elk onderdeel.

Het uniform ontwerp van MAC protocollen met betrekking tot heterogene netwerken

Omdat apparaten in IoT netwerken bestand moeten zijn tegen een groot aantal verschillende toepassingsdomeinen en draadloze omgevingen, bieden leveranciers een verscheidenheid aan apparaten met verschillende hardware aan. Om deze reden is het mogelijk dat IoT netwerken apparaten bevatten met een verschillend hardware design. Binnen deze "heterogene netwerken" moeten alle toestellen gebruik maken van een en hetzelfde MAC protocol. Indien dit niet het geval zou zijn, is het mogelijk dat sommige toestellen niet in staat zijn om met elkaar te communiceren. Om deze situatie te vermijden stellen wij het TAISC framework voor, dat de mogelijkheid aanbiedt om MAC protocollen te schrijven onafhankelijk van de onderliggende hardware. Ontwikkelaars van deze protocollen maken gebruik van de TAISC bouwblokken om op een eenvoudige manier nieuwe ontwerpen te ontwikkelen, met het grote voordeel dat de resulterende code kan gecompileerd worden voor eender welk platform.

Ondanks dat de mogelijkheid bestaat om eenzelfde MAC implementatie te gebruiken op verschillende platformen, betekent dit niet dat de performantie op elk platform hetzelfde is. Omwille van deze performantie verschillen is het mogelijk dat de IoT platformen onverwacht of asymmetrisch gedrag vertonen indien ze gecombineerd worden in een en hetzelfde netwerk. Onze software engine gebruikt

de uitvoeringstijden van de diverse instructies als invoer om de uitvoering van het MAC protocol in detail te plannen. Om de MAC performantie te egaliseren voor alle platformen aanwezig in het draadloze netwerk, stellen we de instructie timings op alle platformen gelijk. In de praktijk betekent dit dat de "snellere" platformen hun prestaties verminderen, om meer in lijn te zijn met de prestaties van de "langzamere" platformen.

De ondersteuning voor meerdere radio's in draadloze MAC protocollen

Vandaag de dag bestaat er een overvloed aan radio technologieën die het gebruik van een protocol set af te dwingen via een draadloze standaard. Door meerdere van deze radio's te combineren op één multimodaal apparaat, kunnen de voordelen van elke draadloze standaard worden gecombineerd. Om draadloze connectiviteit te verkrijgen moet elke radio bestuurd worden door een MAC protocol. Echter, wanneer meerdere MAC protocollen gecombineerd worden op eenzelfde IoT toestel, is het mogelijk dat de protocollen elkaar onderbreken, wat de interne timings kan beïnvloeden. In het slechtste geval kan dit leiden tot pakketverlies, vertragingen, daling van de batterijcapaciteit, enz. Door gebruik te maken van intelligente planningstechnieken, is het mogelijk om overlappende uitvoering van de verschillende protocollen te vermijden.

De inzichten die verworven werden met betrekking tot multimodaal MAC ontwerp, werden aangewend om een nieuw protocol te ontwikkelen in TAISC. Deze laat toe om toestellen te lokaliseren of om afstanden te bepalen tussen de verschillende toestellen. Waar de meeste huidige lokalisatie oplossingen uit gaan van batterijgevoede mobiele toestellen die communiceren met niet-batterijgevoede infrastructuur apparaten, worden in veel gevallen de infrastructuur apparaten ook door batterijen gevoed (bv. in omgevingen zonder bekabeling). Het is dus nodig om ook protocollen te ontwikkelen die energie efficiënt zijn voor de infrastructuur. Ons nieuwe UWB-MAC protocol combineert het gebruik van een subGHz communicatie radio en een UWB radio voor afstandsbepaling. Door alleen de infrastructuur apparaten te activeren op verzoek van een mobiele tag (via de subGHz link), kan veel energie worden bespaard. Er werd aangetoond dat, vooral bij lage updatefrequenties, de levensduur van de batterij tot vier maal kan worden verlengd in vergelijking met de huidige state of the art oplossingen.

Het bereiken van duurzame IoT oplossingen en coëxistentie

Om er zeker van te zijn dat de levensduur van IoT netwerken gewaarborgd blijft, is het belangrijk om er voor te zorgen dat ze kunnen omgaan met (veranderingen in) de omgeving. Ten eerste betekent dit dat ze in staat moeten zijn om het beperkte draadloze spectrum te delen met andere draadloze technologieën. Dit betekent dat data pakketten niet mogen worden verzonden, als een van deze andere technologieën het medium aan het gebruiken is. Dit kan immers leiden tot interferentie, en mogelijks zal de ontvanger niet meer in staat zijn om het verzonden pakket te decoderen. Binnen dezelfde technologie wordt dit typisch vermeden door het gebruik van een MAC protocol. Dit is echter niet mogelijk als het verschillende technologieën betreft, aangezien deze vaak niet in staat zijn om elkanders

draadloos signaal te decoderen. Wij beweren dat het nog steeds mogelijk is om het draadloos medium eerlijk te verdelen tussen Wi-Fi en IoT toestellen door middel van energie signalen. Door gebruik te maken van een TDMA protocol (een typische eerlijk MAC protocol dat gebruik maakt van tijdsloten), kan de tijd eerlijk verdeeld worden tussen Wi-Fi en IoT. De vereiste TDMA synchronisatie wordt bereikt met behulp van een technologie-agnostisch energie baken, dat door zowel IoT als Wi-Fi kan onderscheiden worden.

Ten tweede moet het mogelijk zijn om een toekomstbestendige oplossing aan te bieden, door toe te laten dat de toestellen in het veld worden bijgewerkt. Doorheen de levensduur van IoT toestellen (vaak in de orde van jaren) kan er een heleboel veranderen. Beveiligingsproblemen of bugs worden bijvoorbeeld vaak gedetecteerd pas nadat de toestellen al in gebruik werden genomen. Bovendien kunnen deze toestellen niet profiteren van nieuwe functies en/of optimalisaties, noch kunnen ze zichzelf aanpassen aan nieuwe vereisten van de netwerkbeheerder. Daarom claimen we dat draadloze software-updates een belangrijk onderdeel moeten zijn bij het ontwerp van IoT netwerken. Om deze updates uit te voeren, wordt een tweeledige aanpak ingevoerd. Ten eerste moet de compatibiliteit van de update vooraf worden beoordeeld. Pas bij succesvolle voltooiing mag de tweede fase - de effectieve verdeling van de software - aanvangen, rekening houdende met alle operationele aspecten zoals beveiliging, verspreiding en gelijktijdige activering van de software.

Summary

The concept of the Internet is one that has captivated humanity since its invention in 1969. Originally it was created by the American military to interconnect computing devices over large distances thereby making use of telephone lines. The "packet switching" concept used to exchange data packets, is the ground layer on which the Internet as we know it was built upon. Since then, it has been rapidly evolving from its initial military purpose, towards a universally accessible entity which makes it easier for people to get and share information, and to communicate with each other. One of the key concepts of the Internet today is its versatility, since it allows to perform actions far beyond its original scope. An important example of this was the invention of the Internet of Things (IoT), which promises to - in addition to people and computers - also connect a variety of "things" to the Internet. The terminology of "thing" is purposely vague, in order not to limit its potential. However, it can be defined as any object that is either able to sense the environment as an sensor, or to perform actions on the environment as actuator. Through the wireless exchange of sensing information, various intelligent decisions can be made regarding the interactions with the environment. The IoT concept is currently being integrated in several aspects of modern life, such as city environments, agriculture, healthcare, homes and industry, and this trend is not likely to stop any time soon.

The IoT paradigm is thus being introduced in several different application domains, thereby requiring a lot of versatility from a hardware and software standpoint. The devices should be small, cheap and energy efficient, which results in the use of less powerful or resource constrained hardware. To achieve connectivity between the various IoT platforms, software is required which is able to reliably transfer data packets between the different wireless devices. Instead of the typical TCP/IP software stack found on non-constrained devices, IoT uses a similar layered approach using a different set of networking protocols. Throughout this PhD research, there was a specific focus on the Medium Access Control (MAC) layer as it plays a prominent role on performance parameters such as throughput, latency, or battery lifetime. The MAC layer focuses on how and when the shared wireless medium should be accessed by the various devices, thereby minimizing the possibility of packet collisions. To this date, a large amount of inherently different MAC protocols have been created, each being optimal in a sub-domain in the world of IoT. Unfortunately, due to the continuous growth of the IoT ecosystem, there is currently no single vision on an all encompassing MAC approach which delivers seamless connectivity throughout all the different wireless environments

and application domains.

The work in this dissertation therefore presents solutions to open challenges, in order to achieve sustainable and interoperable MAC design in heterogeneous IoT systems. More specifically, following challenges were pursued:

- Current MAC protocol architectures only offer limited portability towards different hardware platforms;
- Constrained devices can not efficiently use multiple radios on a single platform;
- Current IoT networks have a limited support for coexistence strategies;
- There is a lack of coherent strategies to update devices after deployment.

To fulfill the previously mentioned challenges, a three-tier approach has been created in this PhD dissertation. The following three sections will give more detailed information on the achievements within each tier.

A unified design of MAC protocols across heterogeneous systems

Since devices in IoT networks should be able to cope with a large amount of different application requirements and wireless environments, hardware vendors offer a variety of devices with different hardware capabilities. As a result, current IoT networks could consist of a variety of hardware designs. Within these so called "heterogeneous networks", all devices should use the same MAC protocol. Otherwise, the devices might be un-interoperable, resulting in a degraded network performance. We therefore propose the Time Annotated Instruction Set Computer (TAISC) framework, which delivers a unified way to create hardware independent IoT MAC protocols. MAC developers use the TAISC building blocks to easily create novel designs, and are able to deploy the resulting code towards any device using a cross-platform compilation phase.

Although this enables multi-platform MAC protocols, the protocol performance can vary per device type due to the underlying hardware differences. Consequently, the same MAC protocol implementation acts differently per platform, resulting in unpredictable/asymmetrical behavior when multiple platforms are combined in the same network. To this end, the various instructions out of the TAISC API are measured per platform. Our novel execution engine uses these instruction timings as input to schedule the execution of the MAC protocol. To equalize the MAC level performance across all devices within heterogeneous networks, we enforce the use of the same instruction timings across all platforms. This means that the "faster" platforms drop their performance, to be more in line with the "slower" platforms.

Enabling multi-radio support in wireless MAC protocols

To this date, there exist a plethora of wireless radio technologies, which use a specific protocol set enforced through a wireless standard. By combining multiple

of these radio's on a single multi-modal device, the advantages of each wireless standard can be combined, thereby improving the overall wireless behavior. To achieve wireless connectivity, each radio should be controlled by a MAC protocol. However, when multiple MAC protocols are executed in parallel on constrained devices, protocols can interrupt each other, impacting their internal timings. In worst case, this can lead to increased packet loss, increased delays, loss of battery capacity, etc. Through the use of intelligent scheduling techniques, it is possible to minimize or negate the possibility of overlapping execution.

The insights on multi-modal MAC design approaches was used to create a novel MAC protocol in TAISC, used for localization/ranging purposes. Whereas most current localisation implementations assume battery powered mobile tag nodes to communicate with non-energy constrained infrastructure devices, in a lot of use cases infrastructure nodes are also battery powered and thus require extensive power saving (e.g. in environments without power cabling). Our new UWB-MAC protocol combines the use of a subGHz communication radio, and an UWB radio to perform ranging with infrastructure nodes. By only activating the infrastructure devices when requested by a tag node (via a subGHz link), energy consumption can significantly be reduced. It was shown that, especially on low update rates, the battery lifetime can be extended upwards to four times compared to current state-of-the-art solutions.

Enabling sustainable operation of MAC protocols and coexistence strategies

In order to ensure the longevity of IoT networks, it is important to ensure that they can cope with (changes to) the environment. Firstly, IoT solutions share the same spectrum resources as other wireless technologies. This means that the data transmissions of these other technologies are able to interfere the signal of the IoT devices, which possibly leads to packet loss. Between devices of the same technology, interference gets mitigated through the creation of a MAC protocol which dictates when the various devices may occupy the medium. For the previously mentioned multi-technology networks this is infeasible, since each technology has different incompatible Radio Frequency (RF) characteristics. However, we claim that it is still possible to achieve fair medium access between Wi-Fi and IoT devices through cross-technology signaling. A TDMA scheme is used in order to allow cross-technology medium access between the two technologies. The required TDMA synchronization is performed using a technology-agnostic energy beacon, which both IoT and Wi-Fi devices can distinguish.

Secondly, to deliver a future-proof solution, IoT devices should be able to be updated post-deployment. Most IoT devices have the requirement to achieve years of lifetime on a single battery. Since the world of IoT is a constantly evolving environment, a lot might change during this operational lifetime of the devices. For instance, security issues or bugs are often detected post deployment, thereby hindering operational IoT networks. Moreover, already deployed devices cannot take advantage of new features and/or optimizations, or even adapt to new application requirements. Therefore, we argue that over-the-air software updates should be a key part in the design of IoT networks. To perform these updates, a two step

approach should be followed. First, during the "software module management" phase, the compatibility of the update should be evaluated upfront. Only on successful completion should the second "secure software rollout" phase commence, taking into account all operational aspects such as security, dissemination and simultaneous activation of the software.

1 Introduction

"When wireless is perfectly applied the whole earth will be converted into a huge brain, which in fact it is, all things being particles of a real and rhythmic whole.

–Nikola Tesla (1856 - 1943)

This chapter situates the conducted research work, summarizes the main contributions and outlines the structure of this dissertation. It also provides an overview of the publications that were authored during this research period.

1.1 Enabling connectivity in resource constrained Internet of Things (IoT) networks

According to the English Cambridge Dictionary, communication can be identified as "*the activity of expressing or exchanging information, feeling, etc.*". This definition is somewhat vague, as it does not give information on who performs the information exchange, for whom the information is intended for, or what medium is used to perform the exchange. Throughout time, humanity created several innovative ways to fill in the vague definition of communication (see Figure 1.1). Firstly, direct inter-personal communication was achieved through sign language, facial expressions, or the spoken word. As people also wanted to perform communication over larger distances, they started writing the information down in the form of letters, which were transferred via courier. Due to the Industrial Revolutions, and the technological strides that accompanied these transformative years,

changes were made to how people perceived communication. During the Second Industrial Revolution, Samual Morse invented the telegraph system [1], which allowed people to transmit data over a wired communication line (thereby making use of an electrical signal). The invention of the telegraph was quickly followed by other wired communication technologies which were more practical to use for the common public, such as telephones [2], fax machines [3], etc. The end of the 20th century brought the start of the Third Industrial Revolution, which can be identified as a revolution in terms of telecommunication standards and computing. By (wirelessly) interconnecting digital devices, several - not to say most - industry processes are becoming digitized, thereby improving the efficiency of companies. Also, the influence on personal life can not be underestimated, as a majority of the human population is becoming connected to each other via the digital medium through the use of smartphones, laptops, tablets, etc. In 2012 the terminology "Industry 4.0" was created, which some identify as the "Fourth Industrial Revolution" [4]. Up until now, industry communication was primarily focused around inter-person communication. However, the Industry 4.0 promises to not only interconnect people, but also day-to-day objects or things. The internet like we know it now, thus transforms in an Internet of Things (IoT).

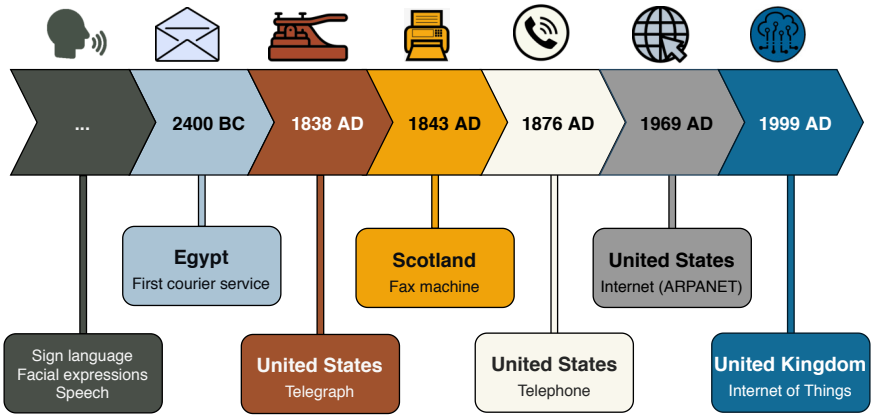


Figure 1.1: A brief history of communication techniques.

1.1.1 Internet of Things (IoT)

To explain the concept of IoT, we again take a look at the Cambridge dictionary for a detailed definition. The Internet of Things is a set of "*objects with computing devices in them that are able to connect each other and exchange data using the internet*". Again the definition is purposely vague, as it does not give any information on either how to achieve the concept, or what purpose these kinds of

devices hold. To gain more detailed insights into the core concepts of IoT, we first look at the toaster of John Romkey, which can be identified as the first real IoT device [5]. During the year 1990, the team of John Romkey was developing the Simple Network Management Protocol (SNMP) protocol [6], which allows for read/write access over remote agents. They wanted to show that next to controlling remote software deployments, the protocol could also be used to control physical systems. Therefore he chose to use a "Sunbeam Deluxe Automatic Radiant Control" toaster, and connected it to the internet. They achieved this by connecting a computer to a relay, which could cut off the toaster's power thereby raising the bread platform. Through configuration, the user could also choose the darkness of the toast. The first IoT device existed in 1990, however the term Internet of Things was only introduced in 1999 by Kevin Ashton [7]. He claimed that before the existence of IoT, "computers were brains without sensors - they only knew what we told them". However, through the invention of IoT, computers and in extension any "thing", gained the ability to sense (e.g. temperature, pressure, air quality, etc.) and perform actions onto the environment as actuators. By exchanging the sensing information, smart decisions can be made regarding the interactions with the environment. In conclusion, like predicted by Nikola Tesla (see introductory quote), the internet is becoming a brain which - similarly to a human body - can sense the environment and perform actions as a connected and rhythmic whole.

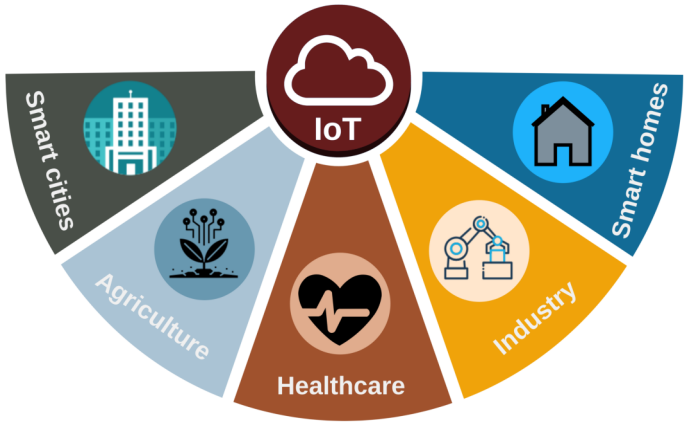


Figure 1.2: Current IoT systems are being deployed in several application domains

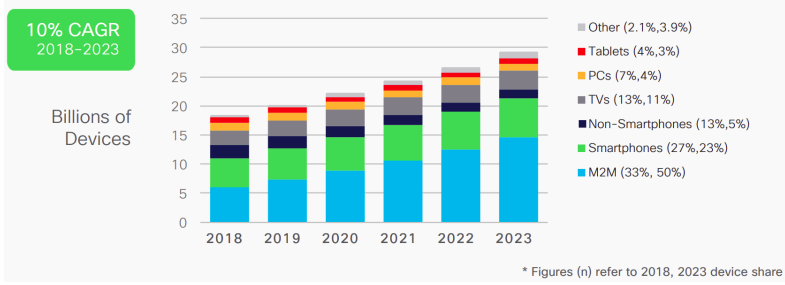
Since the initial introduction, IoT has been gradually digitizing different aspects of modern life. Figure 1.2 gives a brief overview of five different sectors which are adopting IoT technologies to achieve different use cases, including smart cities, agriculture, healthcare, smart homes, and industry. In smart cities, the IoT paradigm is adopted to make better use of the available public resources and ser-

vices, and to improve the quality of life within the city. Through the collection of data using a variety of sensors spread throughout the city, intelligent decisions can be made regarding several aspects of city life including traffic management, the usage of water or electricity, waste collection, etc. Secondly, through the use of IoT in agriculture, farmers can better monitor and improve the efficiency of their farm. For example, resources such as water and fertilizer can be more optimally used, reducing waste (and costs). In healthcare, doctors can make use of IoT to remotely monitor several aspects of their patients, which vastly improves the ability to detect abnormalities within the health of the patient. Additionally, IoT also makes it easier to monitor and maintain medical equipment in hospitals. Within domestic environments, IoT devices are used to perform use cases such as smart building metering, and building automation (smart homes). Lastly, the introduction of IoT in industrial environments is again causing an evolution on how factories are perceived. The digitization of factories is resulting in what some are calling the Fourth Industrial Revolution, which also gets denoted as Industry 4.0 [8]. The IoT concepts are used to shorten development periods, to perform on-demand personalization of products, to create modular and flexible factory environments, and to achieve efficiency throughout the factories grounds.

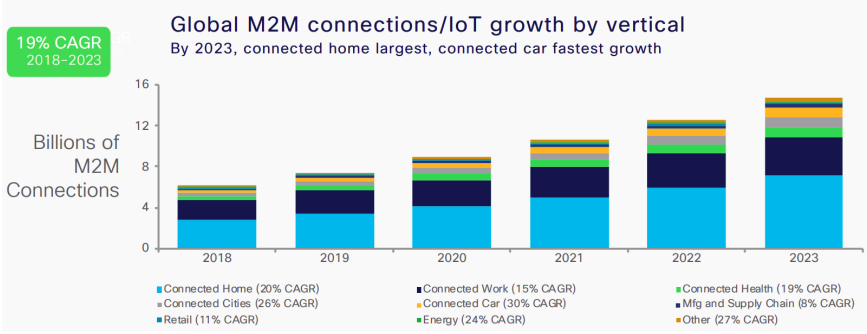
Thus, IoT is gaining traction in several key markets. This translates itself in an ever increasing number of wireless devices. Figure 1.3a shows the predicted growth of devices connected to the internet, most of which can be accounted to Machine to Machine (M2M) devices. The M2M terminology is used for device to device communication without any human intervention, which is the core concept which both IoT and Industry 4.0 rely on. By 2023, about half of the wireless devices will be IoT devices, thereby outgrowing current popular devices such as smartphones, laptops, etc. Figure 1.3b visualizes the expected growth of the M2M connections per vertical market, which again indicates how IoT is more and more becoming a part of day to day life. These predictions show the importance to make further advancements to realize better and more stable IoT networks, in any kind of environment.

1.1.2 Enabling connectivity

As stated before, the IoT paradigm is being introduced in several different application domains, thereby requiring a lot of versatility from a hardware and software standpoint. Typical requirements for IoT platforms are that they should be (i) small - since they need to be applied to any kind of device (big or small) without being visually apparent, (ii) battery powered - since they could be deployed in any area, some without the availability of other power sources, (iii) energy efficient - since they should be able to survive for years on a single battery charge, and (iv) cheap - since they are being deployed in very large numbers. These requirements severely



(a) Global device and connection growth



(b) Global M2M connection growth by industries

Figure 1.3: Predicted growth of wireless technologies from 2018 up to 2023 (both images courtesy of Cisco [9])

limit the possibility to use powerful hardware on the devices. Therefore embedded IoT devices are typically denoted as constrained devices, as they compromise their hardware design in order to meet the stringent requirements. Throughout this dissertation a number of widely used hardware platforms are mentioned, for which a brief summary of the hardware can be found in Table 1.1. It is apparent that, even though there is a positive evolution of the hardware designs, still the Micro Controller Unit (MCU) performance and storage capacity falls well behind those found in other wireless devices such as laptops, smartphones, etc. Additionally, in order to achieve the wireless data exchanges required to create IoT networks, the hardware platforms also become equipped with one or more communication radios. Similarly to MCU and memory, the radio is a constrained component, as most typical representatives are only able to transmit a few bits up to 1Mbit per second.

To achieve connectivity between the various IoT platforms, software is required which is able to reliably transfer data packets between the different wireless nodes. In non-constrained networks this is achieved through a network stack consisting of different network layers, where each layer is responsible to perform a

Table 1.1: Different IoT platforms used throughout this PhD research

Platform	Release date	CPU	Clock speed	RAM	ROM
TelosB	2005	16 bit	8 Mhz	10 kB	48 kB
Z1	2010	16 bit	16 MHz	8 kB	92 kB
RM090	2012	16 bit	16 MHz	16 kB	256 kB
Re-Mote	2015	32 bit	32 MHz	32 kB	512 kB
Lopos	2019	32 bit	64 MHz	64 kB	512 kB

specific sub-task. The rules and conventions which are followed to achieve the sub-task is denoted as a protocol. If two devices in a network use the same set of protocols, they should be able to communicate with each other. Currently TCP/IP is the most well-known example of such a network stack, which consists of four distinct layers. Firstly, the application layer is responsible to provide services towards the end user of the device. Secondly, the transport layer establishes a reliable end-to-end communication link between two applications, while deploying measures such as congestion control, error checking or re-transmissions. The internet layer accepts data from the transport layer, and transfers the data either directly to the destination, or via an intermediary device which is closer towards the destination. Lastly, the network layer is responsible to perform the transmission of the data either over a wired or wireless link. Given the constraints under which IoT works (e.g. memory, processing power, low bandwidth, etc.), it can not directly use the same TCP/IP-stack. Instead, IoT software utilizes the same four layers as TCP/IP, however with a protocol set specifically tailored towards resource constrained networks. Figure 1.4 gives an overview of common networking protocols per network layer, both for the TCP/IP-stack and the IoT stack.

Within the context of this PhD research, there was a specific focus towards the network layer, as it plays the most prominent role on performance parameters such as throughput, latency, or battery lifetime. The network layer can be further divided into a Medium Access Control (MAC) layer and a Physical Layer (PHY) layer. The PHY layer provides the means to transmit and receive data packets over a wireless medium, by encoding the contents using an Radio Frequency (RF) signal. This RF signal has technology specific characteristics such as center frequency, bandwidth, amplitude, etc. Data transmissions can not happen on any random piece of spectrum, as spectrum usage is regulated by governments or regulatory organizations such as the Federal Communications Commission (FCC) or European Telecommunications Standards Institute (ETSI). Typically, these regulations make a distinction between licensed spectrum which gives exclusive access rights over a specific frequency range, and unlicensed spectrum where the available spectrum can be freely used by any device [1]. An example of unlicensed spectrum

[1] Rules may apply (e.g. maximum transmission power or maximum duty cycle)

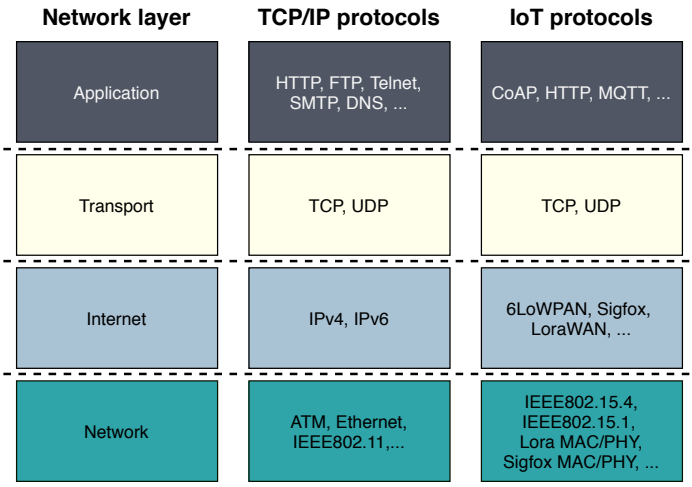


Figure 1.4: Example protocols in (i) the TCP/IP stack, and (ii) the IoT stack

is the Industrial, Scientific and Medical (ISM) band, which is used by wireless technologies such as Wi-Fi, Bluetooth and now also by several IoT technologies. Unfortunately, any interference has the ability to influence the transmission of the RF signal, thereby making it difficult or impossible for the recipient to decode the packet contents. Therefore an important characteristic of technologies residing in this ISM band should be their ability to cope with harmful interference originating from other devices transmitting in overlapping frequency bands.

The PHY layer only offers an interface to interact with the radio chip, while a MAC protocol takes control of the functionality provided by the PHY layer and dictates how and when the various (inter-)actions should occur. Therefore it has full control over the radio capabilities, and hence also the aforementioned performance metrics. A challenge, which MAC protocols are responsible to solve, originates from the wireless spectrum which is shared with other devices, either from the same or different wireless technologies. Through the use of intelligent MAC protocol techniques, it is possible to minimize the impact of lost packets (e.g. through re-transmissions), or to minimize the chances of packet collisions altogether. Common techniques are dividing time into different time slots and allocating the slots to individual devices, allocating different center frequencies to different (groups) of devices, or using Clear Channel Assessment (CCA) techniques to first check if the medium is clear before commencing a packet transmission.

1.2 Research challenges

Even though IoT is a concept which was introduced over two decades ago, still a lot of different research challenges remain. Most of these challenges relate to the future-proofness of IoT networks, as wireless networks in general are continuously evolving due to the ever increasing number of wireless devices and wireless connections. A change is required to the protocols running on these wireless platforms, in order to cope with the changing environmental conditions.

More specifically, this dissertation aims to give a solution for following challenges:

1. Current MAC protocol architectures only offer limited portability towards different hardware platforms;
2. Constrained devices can not efficiently use multiple radios on a single platform;
3. Current IoT networks have a limited support for coexistence strategies;
4. There is a lack of coherent strategies to update devices after deployment.

Next subsections give a more detailed description of each challenge.

1.2.1 Challenge 1: Current MAC protocol architectures only offer limited portability towards different hardware platforms

In Table 1.1 an overview was given on the hardware platforms used throughout this thesis. It clearly showed that - throughout the last decade - there was a continuous evolution in hardware performance. This is similar to the trend followed by processors found in more capable machines such as servers, laptops, or even smartphones. This trend was already predicted in 1965, when Gordon Moore proposed what most people know as “Moore’s law”. This law states that the transistor density on processors will double every two years [10], and as a result processors benefit from an increase in performance and a decrease in unit price. Today, we know that Moore’s prediction was mostly true, although the evolution is slowing down slightly as we are approaching the physical limitations of hardware designs [11]. Although IoT clearly benefits from a decrease in footprint size and an increase in processing power, these are not always the most important metrics. Power consumption is an equally important metric which should not be overlooked, as most IoT devices are battery powered. As these devices should be able to survive for years on a single battery charge, an IoT platform should be as battery efficient as possible. Unfortunately it is hard to achieve both a large amount of processing

power and battery efficiency, as they have a direct impact on each other. Therefore, most hardware vendors offer a wide variety of different designs, each offering a different level of hardware performance or hardware capabilities. The same is valid for the radio chips deployed on the hardware platforms, which are also continuously evolving to be more powerful and more energy efficient. As a result, current IoT networks could consist of a variety of hardware designs, either consisting of older devices, or by the previously mentioned variety of modern devices.

Although IoT networks can consist of different hardware platforms, they still should all "speak the same language" in order to be able to communicate with each other. From a software standpoint, this means that they all need to have the same protocol set. Especially on the MAC layer, special measures should be undertaken as this layer performs a lot of interactions directly with the hardware. MAC implementations are typically written in low level, hardware specific code, which makes it very hard to re-use existing MAC protocols for different radio chips and/or technologies. This is especially problematic for Small and Medium-sized Enterprises (SME) system integrators that target niche markets with specialized MAC protocols. Most of the time their custom designed MAC protocol can not directly be re-used on new radio chips, leading to a manpower and cost intensive redesign and testing phase that does not bring any added value to the end product.

Even if a multi-platform MAC implementation is available, the performance of the MAC protocol varies strongly depending on the actual underlying hardware. Hardware differences do still subtly alter the behavior of a MAC protocol compared to the one running on other radio platforms or legacy devices. Consequently, the same MAC protocol implementation acts differently per platform, resulting in unpredictable/asymmetrical behavior when multiple platforms are combined in the same network. In more extreme cases, the MAC protocol performance can degrade or even become incompatible between the different hardware platforms. A MAC protocol architecture should therefore be able to equalize the wireless performance across all radio platforms, independent of the underlying hardware.

1.2.2 Challenge 2: Constrained devices can not efficiently use multiple radios on a single platform

Throughout the history of Internet of Things (IoT), a large number of strides have been made to optimize the behavior of wireless devices in increasingly diverse application domains. This resulted in a plethora of wireless standards, which enforce the use of a set of protocols per network layer, each of which have a number of (dis)advantages compared to the other technologies making them suitable for a (sub)domain within the IoT eco-system. By combining multiple wireless standards and their dedicated radio on a single multi-modal device, the advantages of each wireless standard or protocol can be combined, thereby improving the over-

all wireless behavior. Making use of these multi-modal IoT platforms does not necessarily have a big impact on the unit price, given the ever decreasing costs of radio chips. However, it has an impact on other parameters as energy consumption, and comes at an additional processing cost. Additionally, due to the inherent limitations of IoT devices, multi-modal platforms require an extensive redesign of the wireless stack, especially to the MAC layer. Unfortunately, although a lot of research acknowledges the advantages related to multi-radio platforms, there is a lack of research on how to efficiently perform the required redesign of the MAC layer.

1.2.3 Challenge 3: Current IoT networks have a limited support for coexistence strategies

When different technologies use the same frequency bands in close proximity, the resulting interference typically results in performance degradation for both technologies. For example, when IoT nodes and Wi-Fi nodes are within the same collision domain, the resulting interference can lead to a throughput loss of up to 30% for the Wi-Fi nodes and 60% for the IoT nodes [12]. Coexistence methods exist, but these are often technology specific and require technology specific interference detection methods. To remove the root cause of the performance degradation, devices should be able to negotiate medium access even when using different technologies. However, agreements on medium usage between different technologies is difficult due to a lack of direct communication possibilities between devices. Several attempts were made at creating a more advanced architecture that allows cross-technology communication. These architectures were not scalable due to overhead, or were limited to only a couple of technologies [13, 14].

1.2.4 Challenge 4: There is a lack of coherent strategies to update devices after deployment

Previously, it was already mentioned that IoT devices can be deployed in a variety of different application domains, each of which have inherently different requirements in terms of throughput, battery lifetime, reliability, etc. Wireless network designers need to take into account the trade-offs between these performance metrics. As an example, when using wireless IoT devices to replace wired control loops in industry processes, the network protocols should focus on providing low latency and high reliability, whereas temperature monitoring applications often emphasize long network lifetime requirements. To this end, an important design decision is the choice of the MAC protocol, which manages how and when the wireless medium is accessed. Countless protocols have been designed with advantages and disadvantages regarding different performance metrics, making it challenging to make an informed decision about the optimal protocol. The initial

choice of MAC protocol might become less optimal during the lifetime of the device, as (i) it can not take advantage of novel features and/or optimizations, (ii) it can not adapt to new application requirements, and (iii) it can not cope with changing networking conditions. A very good example, ties in to Challenge 3. A device is being deployed with a set of co-existence strategies, which are able to cope with the current networking conditions. However, during the lifetime of the device it encounters different wireless technologies, which comes at a cost of wireless performance. It would be beneficial to perform an over-the-air update, integrating novel wireless paradigms into the device without impacting the operation of the device. Unfortunately, there is a lack of a coherent strategy to update IoT devices after deployment, which includes every aspect ranging from pre-deployment compatibility checks to update security and data dissemination.

1.3 Outline

This dissertation is composed of a number of publications that were realized within the scope of this PhD. The selected publications provide an integral and consistent overview of the work performed. The different research contributions are detailed in Section 1.4 and the complete list of publications that resulted from this work is presented in Section 1.6. Within this section we give an overview of the remainder of this dissertation and explain how the different chapters are linked together. Fig. 1.5 positions the different contributions that are presented in each chapter (Ch.). The dissertation is split between three different parts, each of which focus on different challenges which can be encountered within IoT. Table 1.2 shows the challenges that were highlighted in Section 1.2 and indicates which were targeted per part and chapter.

Throughout Part I, this dissertation aims to deliver an architecture and the relevant techniques required to achieve fully portable MAC protocols in heterogeneous IoT systems (see Challenge 1). First, Chapter 2 introduces the Time Annotated Instruction Set Computer (TAISC) framework. This work was the foundation on which most of the PhD research was built, as TAISC served as a key enabler for several solutions within the IoT ecosystem. In short, TAISC is a framework which offers a simplified way to perform hardware independent MAC protocol development and management, through the use of a platform independent language. A cross-platform compilation phase makes it possible to reuse the initial platform independent MAC protocol code across any supported IoT platform. Additionally, by annotating the platform independent instructions with platform specific timings, the execution of the MAC protocol is fully deterministic, thereby supporting time-critical behavior of the MAC protocol.

Chapter 3 is a continuation of the work on portable MAC protocols. Although TAISC was used as an enabler for this work, the concepts elaborated throughout

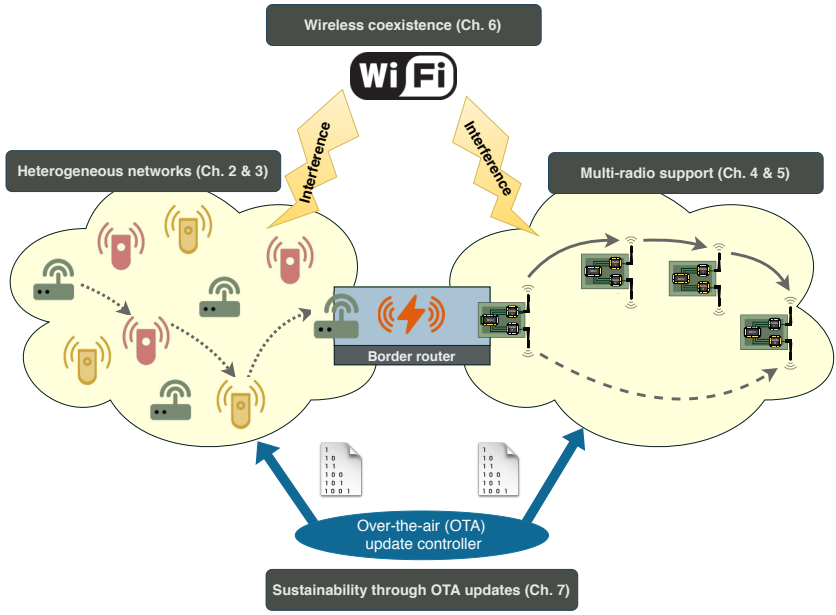


Figure 1.5: Schematic position of the different chapters in this dissertation, split into four categories: (i) MAC protocol design for heterogeneous networks, (ii) multi-radio support in IoT MAC protocols, (iii) wireless coexistence between IoT and Wi-Fi networks, and (iv) sustainability by enabling over the air software upgrades in constrained IoT networks.

the chapter are more generally applicable and can (in principle) be implemented in any MAC protocol framework. Even though the underlying software implementation could be the same, in real-life conditions the performance of MAC protocols vary strongly depending on the actual underlying hardware implementation (e.g. MCU and radio chip). As a result, the "same" MAC protocol implementation can act differently per platform, resulting in unpredictable or asymmetrical behavior when deployed into a single heterogeneous network. Therefore this chapter offers a three step methodology to avoid cross-platform compatibility issues to equalize the protocol performance across all devices in the network. Step 1 provides a hierarchical framework of interfaces which allows efficient code implementation while allowing fall-backs to alternative implementations for portability. Step 2 provides automatic benchmarking of MAC instruction timings, thereby gaining the possibility to optimize the protocol towards the specific radio chip timings. Lastly, through the addition of either margins to equalize the duration of all radio instructions, or through the alignment of critical instructions across multiple platforms, unpredictable multi-platform effects can be removed and thus fairness will be restored.

Part II adds another level of complexity, since it assumes that each platform

could have multiple radios or radio technologies at its disposal (see Challenge 2). Through the use of these so called "multi-modal" devices, Chapter 4 attempts to combine the advantages of each radio technology in order to achieve some advanced use cases (e.g. to switch between the technologies based on networking conditions, to create a multi-technology mesh network, etc.). However, since most IoT devices are constrained in terms of processing power, running multiple standards and more specifically MAC protocols on a single device, is not straight forward. The occurrence of interrupts originating from different protocols, which delay the time critical execution, can have a negative impact on the medium access and can lead to packet loss, loss of battery capacity, increased delays, etc. Within this chapter, multiple techniques are proposed to efficiently run multiple MAC protocols on a single device. An overview is given with the (dis)advantages per technique, so MAC designers can choose the optimal solution per requested use case.

Within Chapter 5, the multi-radio concepts elaborated throughout Chapter 4 are used to create a novel multi-radio MAC protocol, combining the use of a sub-GHz radio with an Ultra Wide Band (UWB) radio. The UWB technology has existed for a few decades, however only during the last years it is attracting more attention since it can also be utilized for indoor localization purposes. Whereas most current UWB research papers and commercial offerings assume battery-powered mobile tag nodes communicate with non-energy constrained infrastructure devices, in a lot of use cases infrastructure nodes are also battery powered (e.g. in environments without power cabling). Thus, in these situations the power efficiency of infrastructure nodes becomes equally important, thereby requiring novel techniques minimizing the energy consumption for the infrastructure. This chapter introduces a multi-radio protocol which uses a sub-GHz radio for medium contention between the tags, and wake up signaling of a selection of infrastructure nodes. The secondary UWB radio is subsequently activated to determine the range between the mobile tag and anchors. A first implementation was created using the TAISC framework, introduced in Chapter 2.

Part III tackles two issues, ensuring the longevity of the wireless IoT networks. Firstly, Chapter 6 offers a coexistence technique in order for IEEE 802.15.4 networks (which is a standard within IoT) and IEEE 802.11 networks (which is the Wi-Fi standard) to co-exist with each other, sharing the same wireless resources available within the 2.4GHz domain. For this purpose, this chapter proposes a time slotted approach, where individual time slots are allocated to the two available wireless technologies. In order to achieve cross-technology synchronization, which is required for the slotted solution, an energy pattern beacon is transmitted from a IEEE 802.11 device. For the IoT devices, a proof of concept implementation was developed using the TAISC framework (Chapter 2).

Lastly, Chapter 7 offers an approach to achieve sustainability for the future, by

being able to update the constrained IoT devices post-deployment. Multiple over-the-air related techniques are already available, each covering a specific aspect of the update process, such as (partial) code updates, data dissemination and security. However, up until now a comprehensive overview describing the different update steps and quantifying the impact of each step is missing in scientific literature, making it hard to assess the overall feasibility of an over-the-air update. To this end, this chapter proposes a step-by-step approach to integrate software updates in IoT solutions, and quantifies the energy cost of each of the involved steps.

In conclusion, this PhD thesis makes is possible to efficiently create novel MAC protocols, which are able to be deployed on any supported wireless platform. Wireless networks consisting of a heterogeneous set of IoT hardware platforms can now ensure fairness between all devices, since performance is equalized between all the platforms. Additionally, the constraint of only have a single radio/MAC protocol per device is removed, as several techniques are proposed to achieve multi-radio support in MAC protocols. A technique is proposed which allows coexistence between IoT networks and Wi-Fi networks. Through the over-the-air update process, it is possible to integrate new MAC protocols into existing networks with minimal effort, or to integrate novel coexistence strategies. The resulting architecture is thus a fully sustainable solution, offering solutions for current IoT networks, and offering the means to keep deployed devices up to date with the latest software versions.

Table 1.2: An overview of the contributions per chapter in this dissertation.

	Part I		Part II		Part III	
	Ch.2	Ch.3	Ch.4	Ch.5	Ch.6	Ch.7
Challenge 1	•	•				
Challenge 2			•	•		
Challenge 3					•	
Challenge 4						•

1.4 Research contributions

In Section 1.2, the problems and challenges related to achieving wireless connectivity in heterogeneous networks are formulated. They are tackled in the remainder of this PhD dissertation for which the outline is given in Section 1.3. To conclude, we present an elaborated list of the research contributions within this dissertation:

- The design of a solution to enable efficient MAC protocol design in heterogeneous IoT systems (Chapter 2).

  - The design of the TAISC framework for device-agnostic MAC protocol development and management.
  - The implementation of a cross-platform compilation step, resulting in a dedicated binary code optimized in terms of execution and code size.
  - The addition of platform dependent time annotation of the platform independent instructions, enabling time-critical execution of MAC protocols on a per platform basis.
  - The use of flow verification, which performs a pre-execution check on the binary code (e.g. to check if the radio is in the correct state to perform certain actions).
  - Quantitative analysis of the TAISC framework, showing only a minimal instruction overhead and a reduction in code size through the use of a bytecode format.
  - The ability to provide upgrade-ability of MAC protocols post-deployment (used as a showcase for Chapter 7).

- The design of a methodology enabling portability and reuse of MAC protocols across different IoT platforms (Chapter 3).
  - The identification of three challenges related to heterogeneous IoT networks:
    * There is a lack of feature-rich Application Programming Interface (API)'s which still support portability;
    * MAC protocol instruction timings have a strong hardware dependency;
    * Heterogeneous networks exhibit unpredictable behavior.
  - An experimental quantification of the performance impact per challenge.
  - The proposal of a solution per challenge:
    * The creation of a hierarchical framework of interfaces which allows efficient code implementation while allowing fall-backs to alternative implementations for portability;
    * Automatic benchmarking of MAC instruction durations, thereby gaining the possibility to optimize the MAC protocol towards the specific radio chip timings;
    * The addition of instruction margins to equalize the duration of all radio instructions, or the alignment of critical instructions across multiple platforms, thereby removing unpredictable multi-platform effects.

  - The creation of a methodology to enable proper portability and reuse of MAC protocols across different IoT platforms.
  - An experimental validation of the methodology using state-of-the-art MAC design approaches.

- Supporting multi-radio platforms in constrained IoT networks (Chapter 4).
  - Giving an overview of current issues related to multi-modal devices, and more specifically to the MAC-level implementation.
  - The creation of several MAC development techniques to efficiently use the hardware resources of multi-modal platforms:
    * To switch between multiple pre-installed MAC protocols;
    * To create a single MAC protocol controlling all available interfaces;
    * To divide the MAC protocols in un-interruptable code blocks;
    * To execute instructions in a slot-based approach, allocating slots to individual protocols;
    * To use multiple processors on a single platform, each controlling a single radio;
  - A simulation based evaluation is given, comparing each approach in terms of achievable throughput and execution latency.
  - An overview is given with the (dis)advantages per technique, so MAC designers can choose the optimal solution per requested use case.

- The design (and implementation) of UWB-MAC, which is a MAC protocol for UWB localization using ultra-low power anchor nodes (Chapter 5).
  - The UWB-MAC protocol is proposed, which is an innovative contention based multi-tag MAC protocol where anchor nodes are only activated in the presence of mobile tags.
  - A low complexity anchor node selection and wake-up algorithm is described to scale the solutions towards large deployments where multiple areas need to be covered.
  - The constraints under which the protocol works are expressed using several formulas.
  - An initial implementation was created using the TAISC framework, previously described in Chapter 2.
  - The performance of the protocol is compared to state of the art solutions, showing that our work significantly decreases the power consumption of the UWB infrastructure nodes.

- The design (and implementation) of a coexistence strategy between IEEE 802.15.4 and IEEE 802.11 (Chapter 6).
  - The use of a cross-technology synchronization phase based on energy detection.
  - The use of cross-technology communication which enables channel access control in different technologies.
  - A multi-platform set-up and evaluation of the solution in a large scale testbed environment.
- The design of a strategy to perform over-the-air software updates in IoT (Chapter 7).
  - An analysis concerning the distribution of software development effort in different parts of widely used IoT operating systems.
  - A comprehensive overview of the key steps in an over-the-air update process.
  - A quantification of the energy overhead per deployment phase, showing the relative energy impact.
  - A discussion on the impact of updates on operational processes, such as the versioning approach used for software modules.
  - A list of future research directions that could enhance the potential of over-the-air updates.

1.5 Software

All the software, tools, experiments, and data sets created in the context of this PhD thesis are available for public use under a dual licensing agreement. By using these resources, you agree to:

- Only use them for academic purposes.
- If you publish an article that contains results you obtained by using the framework, you will cite the following publication ”Bart Jooris, Jan Bauwens, Peter Ruckebusch, Peter De Valck, Christophe Van Praet, Ingrid Moerman, Eli De Poorter, TAISC: A cross-platform MAC protocol compiler and execution engine, Computer Networks, ISSN 1389-1286”
- You will not engage in a funded project using the frameworks without consulting the owners.

To request the resources, please contact the IDLab research group.

1.6 Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences. The following list provides an overview of the publications during my PhD research.

1.6.1 Publications in international journals (listed in the Science Citation Index [2])

1. Bart Jooris, **Jan Bauwens**, Peter Ruckebusch, Peter De Valck, Christophe Van Praet, Ingrid Moerman, Eli De Poorter *TAISC: a cross-platform MAC protocol compiler and execution engine* Published in Elsevier Computer Networks, Volume 107, Part 2, 9 Octrober 2016, Pages 315-326.

2. Michael Tetemke Mehari, Adnan Shahid, Tom Van Steenkiste, **Jan Bauwens**, Ivo Couckuyt, Violet R. Syrotiuk, Dirk Deschrijver, Tom Dhaene, Ingrid Moerman, Eli De Poorter *Metamodel Based WSN MAC Optimization in Dynamic Environments using Cloud Repositories.* Submitted to IEEE/ACM Transactions on Networking, 2018.

3. **Jan Bauwens**, Bart Jooris, Spilios Giannoulis, Irfan Jabandžić, Ingrid Moerman, Eli De Poorter *Portability, compatibility and reuse of MAC protocols across different IoT radio platforms* Published in Elsevier Ad Hoc Networks, Volume 86, 1 April 2019, Pages 144-153.

4. Nicola Macoir, **Jan Bauwens**, Bart Jooris, Ben Van Herbruggen, Jen Rossey, Jeroen Hoebeke, Eli De Poorter *Uwb localization with battery-powered wireless backbone for drone-based inventory management* Published in MDPI Sensors, Volume 19, 2019(3), Pages 144-153.

5. **Jan Bauwens**, Peter Ruckebusch, Spilios Giannoulis, Ingrid Moerman, Eli De Poorter *Over-the-Air Software Updates in the Internet of Things: An Overview of Key Principles* Published in IEEE Communications Magazine, Volume 58, 2020(2), Pages 35-41.

6. Dries Van Leemput, **Jan Bauwens**, Robbe Elsas, Jeroen Hoebeke, Wout Joseph, Eli De Poorter *Adaptive Multi-PHY IEEE802.15.4 TSCH in Sub-GHz Industrial Wireless Networks* Submitted in Elsevier Ad Hoc Networks, 2020.

[2] The publications listed are recognized as 'A1 publications', according to the following definition used by Ghent University: A1 publications are articles listed in the Science Citation Index Expanded, the Social Science Citation Index or the Arts and Humanities Citation Index of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper.

7. **Jan Bauwens**, Dries Van Leemput, Spilios Giannoulis, Ingrid Moerman, and Eli De Poorter. *Multi-radio support in energy constrained Internet of Things (IoT) networks: Overview of current and future approaches* Submitted in Elsevier Pervasive and Mobile Computing, 2020.

8. **Jan Bauwens**, Nicola Macoir, Spilios Giannoulis, Ingrid Moerman, and Eli De Poorter. *UWB-MAC: MAC protocol for UWB localization using ultra-low power anchor nodes* To be published in Elsevier Ad Hoc Networks, 2021, Volume 123.

1.6.2 Publications in book chapters

1. Nicholas Kaminski, Spilios Giannoulis, Ilenia Tinnirello, Peter Ruckebusch, Piotr Gawlowicz, Domenico Garlisi, **Jan Bauwens**, Anatolij Zubow, Robin Leblon, Pierluigi Gallo, Ivan Seskar, LA DaSilva, Sunghyun Choi, Jose De Rezende, Ingrid Moerman. *Wireless Software and Hardware Platforms for Flexible and Unified Radio and Network Control (WiSHFUL)* Chapter in the book Building the future internet through FIRE, ISBN: 978-87-93519-12-1, 2016.

1.6.3 Publications in international conferences (listed in the Science Citation Index [3])

1. Peter Ruckebusch, **Jan Bauwens**, Bart Jooris, Spilios Giannoulis, Eli De Poorter, Ingrid Moerman, Domenico Garlisi, Pierluigi Gallo, Ilenia Tinnirello *Cross-technology wireless experimentation: Improving 802.11 and 802.15. 4e coexistence* Published in proceedings of the 17th IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016, pages 529-534, Atlanta, United States.

2. **Jan Bauwens**, Bart Jooris, Peter Ruckebusch, Domenico Garlisi, Josesph Szurley, Marc Moonen, Spilios Giannoulis, Ingrid Moerman, Eli De Poorter *Cross-technology TDMA synchronization using energy pattern beacons: demo* Presented at INFOCOM, the IEEE Conference on Computer Communications Workshops, pages 529-534, Atlanta, United States, Jun. 2017.

3. **Jan Bauwens**, Bart Jooris, Peter Ruckebusch, Domenico Garlisi, Josesph Szurley, Marc Moonen, Spilios Giannoulis, Ingrid Moerman, Eli De Poorter *Coexistence between IEEE802. 15.4 and IEEE802. 11 through cross-technology*

[3]The publications listed are recognized as 'P1 publications', according to the following definition used by Ghent University: P1 publications are proceedings listed in the Conference Proceedings Citation Index - Science or Conference Proceedings Citation Index - Social Science and Humanities of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper, except for publications that are classified as A1.

signaling Published in proceedings of the 2017 CNERT workshop at the IEEE Conference on Computer Communications Workshops, pages 529-534, Atlanta, United States.

4. Jetmir Haxhibeqiri, Adnan Shahid, Martijn Saelens, **Jan Bauwens**, Bart Jooris, Eli De Poorter, Jeroen Hoebeke *Sub-gigahertz inter-technology interference. How harmful is it for LoRa?* Published in proceedings of the 2018 IEEE international smart cities conference (ISC2), pages 1-7, Missouri, United States.

5. Nicola Macoir, Matteo Ridolfi, **Jan Bauwens**, Bart Jooris, Ben Van Herbruggen, Jen Rossey, Jeroen Hoebeke, Eli De Poorter. *Low power, portable and infrastructure light indoor UWB ranging solution* Published in proceedings of the 18th International Conference on Information Processing in Sensor Networks, pages 337-339, Montreal, Canada, 2019.

6. Adnan Sabovic, Carmen Delgado, **Jan Bauwens**, Eli De Poorter, Jeroen Famaey *Accurate Online Energy Consumption Estimation of IoT Devices Using Energest* Published in proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications, pages 363-373, Antwerp, Belgium, 2019.

1.6.4 Publications in other international conferences

1. **Jan Bauwens**, Bart Jooris, Eli De Poorter, Peter Ruckebusch, Ingrid Moerman *Towards a MAC protocol app store: demo* Presented at EWSN2016, the International Conference on Embedded Wireless Systems and Networks, pages 249-250, Graz, Austria, Feb. 2016.

References

[1] S. F. Morse. *Telegraph Signs*. US Patent, 1647, 1840.

[2] A. G. Bell. *Improvement in telegraphy*, March 7 1876. US Patent 174,465.

[3] L. M. Mag. *Mr. Bain's Electric Printing Telegraph*. Journal of the Franklin Institute, of the State of Pennsylvania, for the Promotion of the Mechanic Arts; Devoted to Mechanical and Physical Science, Civil Engineering, the Arts and Manufactures, and the Recording of American and Other Patent Inventions (1828-1851), 8(1):61, 1844.

[4] W. Wahlster. *From industry 1.0 to industry 4.0: Towards the 4th industrial revolution*. In Forum Business meets Research, 2012.

[5] J. Romkey. *Toast of the IoT: the 1990 interop internet toaster*. IEEE Consumer Electronics Magazine, 6(1):116–119, 2016.

[6] J. Case, M. Fedor, M. L. Schoffstall, and J. Davin. *RFC1157: Simple network management protocol (snmp)*, 1990.

[7] K. Ashton. *An Introduction to the Internet of Things (IoT)*. RFID Journal, 1999.

[8] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann. *Industry 4.0*. Business & information systems engineering, 6(4):239–242, 2014.

[9] Cisco. *Cisco Annual Internet Report - Cisco*. https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html. (Accessed on 07/13/2020).

[10] G. Moore. *Moore's law*. Electronics Magazine, 38(8):114, 1965.

[11] T. N. Theis and H.-S. P. Wong. *The end of moore's law: A new beginning for information technology*. Computing in Science & Engineering, 19(2):41–50, 2017.

[12] R. A. Saeed, S. Khatun, B. M. Al, and M. Khazani. *Performance analysis of ultra-wideband system in presence of IEEE802. 11a and UMTS/WCDMA frequency bands*. In Information and Communication Technology, 2007. ICICT'07. International Conference on, pages 117–122. IEEE, 2007.

[13] R.-C. Wang, R.-S. Chang, and H.-C. Chao. *Internetworking between ZigBee/802.15. 4 and IPv6/802.3 network*. SIGCOMM Data Communication Festival, 2007.

[14] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler. *Trio: enabling sustainable and scalable outdoor wireless sensor network deployments*. In Proceedings of the 5th international conference on Information processing in sensor networks, pages 407–415. ACM, 2006.

Part I

Unifying the design and implementation of MAC protocols across heterogeneous platforms

2

TAISC: a cross-platform MAC protocol compiler and execution engine

This chapter introduces the Time Annotated Instruction Set Computer (TAISC) framework, which forms the foundation on which most of the PhD research was built on. In short, TAISC is a framework which offers a simplified way to perform hardware independent Medium Access Control (MAC) protocol development and management, through the use of a platform independent language.

⋆ ⋆ ⋆

B. Jooris, J. Bauwens, P. Ruckebusch, P. De Valck, C. Van Praet, I. Moerman, and E. De Poorter.

Accepted in Elsevier Computer Networks, March 2016.

Abstract MAC protocols significantly impact wireless performance metrics such as throughput, energy consumption and reliability. Although the choice of the optimal MAC protocol depends on time-varying criteria such as the current application requirements and the current environmental conditions, MAC protocols can not be upgraded after deployment since their implementations are typically written in low level, hardware specific code which is hard to reuse on other hardware platforms. To remedy this shortcoming, this paper introduces TAISC, a framework for hardware independent MAC protocol development and management. The solution

presented in this paper allows describing MAC protocols in a platform independent language, followed by a straightforward compilation step, yielding dedicated binary code, optimized for specific radio chips. The compiled code is as efficient in terms of memory footprint as custom-written protocols for specific devices. To enable time-critical operation, the TAISC compiler adds exact time annotations to every instruction of the optimized binary code. As a result, the TAISC approach can be used for energy-efficient cross-platform MAC protocol design, while achieving up to 97% of the theoretical throughput at an overhead of only 20μs per instruction.

2.1 Introduction

The presence of wireless communication technologies is ever increasing and is expected to increase further due to the rise of Internet of Things (IoT) paradigm. Over the last two decades the speed at which these standards are being developed has grown significantly, resulting in a proliferation of wireless communication standards, each suitable for different application domains. Due to the shared nature of the limited Industrial, Scientific and Medical (ISM) radio frequencies used in many wireless communication technologies, the use of an efficient MAC protocol is crucial to ensure efficient wireless communications without collisions [1]. Depending on the targeted application domain, wireless MAC protocols are typically optimized for low-power operation (e.g. ZigBee, Bluetooth low energy), low latency (e.g. wireless HART), high reliability (e.g. TSCH), high throughput (e.g. Wi-Fi) or a combination of the aforementioned requirements.

Due to increasingly diverse deployments and application loads, a sole one-size-fits-all MAC protocol can no longer meet the demands of many future wireless applications. As a result, a large number of MAC protocols have been proposed and evaluated in the scientific literature [2]. However, in practice it is often not possible to select the most optimal MAC protocol for a specific application domain due to the following reasons: (i) wireless radio chips typically do not provide any support for updating, changing or replacing the MAC after deployment and (ii) since MAC implementations are typically written in low level, hardware specific code, it is very hard to reuse existing MAC protocols for different radio chips and/or technologies. To remedy this situation, this chapter proposes the Time Annotated Instruction Set Computer (TAISC) framework for radio chips that solves both of these problems by (i) *providing upgrade-ability of the MAC protocol* and (ii) *supporting reuse of MAC implementations through a cross-platform compilation phase*.

Several good reasons exist to support *upgrade-ability and maintainability of MAC protocols*. (i) MAC protocols from vendors often exhibit unwanted behavior or contain bugs that limit the performance of the wireless network. By

supporting over-the-air upgrade capabilities of the radio firmware, these limitations can be corrected also after deployment. (ii) Wireless technologies are often replaced by *new standards* after just a few years, often building on top of the same physical layer (e.g.: ZigBee, Wireless HART, Time Synchronized Channel Hopping (TSCH) all use the IEEE 802.15.4 PHY layer). By supporting MAC upgrade-ability, users would be able to switch towards newer standards without having to replace all wireless radio boards. (iii) Wireless upgrade-ability of MAC protocols would allow users to cope with changing environmental conditions, for example by switching to a more interference robust MAC protocol. (iv) Similarly, over time, application requirements might change, for example when adding new devices to the network, it may be necessary to switch from Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to Time Division Multiple Access (TDMA) based MAC protocols to cope with the increasing density of connected wireless devices.

Since MAC protocols are typically written in low level, hardware specific code, *current MAC protocols can not be reused for different radio chips and/or technologies*. This is especially problematic for Small and Medium-sized Enterprises (SME) system integrators that target niche markets with specialized MAC protocols. Due to hardware obsolescence (i.e. the fact that radio chips from chip vendors typically have a life cycle of just a few years, far less than the life cycle of the end products they are used in), system integrators are frequently forced to integrate newer radio chip versions in their product. Most of the time their custom-designed MAC protocol can not directly be reused on new radio chips, leading to a manpower and cost intensive redesign and testing phase that does not bring any added value to the end product.

To remedy the above problems, TAISC introduces a cross-compilation process allowing MAC developers to design MAC protocols once, and then compile them for reuse on different radio chips. Although the MAC protocols are device-independent, TAISC allows the design of very efficient sleep schemes for energy constrained devices through the use of time-annotated instructions, thereby supporting time-critical behavior of the MAC protocol.

The remainder of this chapter is structured as follows. Section 2.2 gives an overview of related work. Next, Section 2.3 gives a high level overview of the work-flow used to create and upload MAC protocols using the TAISC approach. Afterwards, Section 2.4 describes in detail the TAISC architecture, followed by a detailed performance evaluation in Section 2.5. Finally, Section 2.6 concludes the chapter.

2.2 Related work

In the field of wireless communications we see an evolution away from the rigid and layered communication stack towards more flexible cross-layer approaches. In terms of reconfigurability at the MAC level, we distinguish two types of platforms: Software Defined Radio (SDR) and off-the-shelf.

2.2.1 Reconfigurable MAC protocols for SDR platforms

SDRs offer previously unseen flexibility due to the presence of reconfigurable physical layer parameters [3]. However, although MAC solutions for SDRs exist [4], flexibility in terms of over-the-air run-time upgrade-ability and portability is still lacking. Two notable architectures for the design of reconfigurable MAC on SDRs are the *Iris* and *GNU radio* platforms.

The Iris architecture [5] implements a MAC engine to handle higher layer functionality on SDR platforms, allowing SDR developers to build reconfigurable devices. To this end, the architecture includes a Process Network (PN) engine, allowing reconfiguration of the physical layer blocks (such as filters, encoders, modulation schemes, etc.), as well as a Stack engine that can be used to implement a MAC protocol. The MAC protocol designer can choose to expose a number of parameters for dynamically adjusting the operation of the components.

GNU radio was originally designed for physical layer processing on SDR platforms. Since GNU radio runs on a computing device that is connected to the SDR, implementing a time critical MAC over a slow connection bus can be problematic [1]. To this end, [6] implements a MAC architecture by making a distinction between MAC functions and core MAC functions. TheMAC functionality that needs to be executed without delays are called core MAC functions and are executed on the SDR platform, whereas other functions can be executed on the computing device. Since GNU radio by itself does not support the concept of packets or frames, MAC protocol design is possible through the addition of Click [7], a framework for higher layer protocol development, or a similar framework that supports higher layer networking.

2.2.2 Reconfigurable MAC protocols for off-the-shelf radios

In addition to reconfigurable MAC solutions for SDR platforms, several solutions among which Wireless MAC Processor (WMP) [8] and snapMAC [9] exist for reconfiguring off-the-shelf devices.

The WMP [8] is designed for reconfiguring Wi-Fi MAC protocols on commodity Broadcom B43 Wi-Fi radio chips. MAC protocols are implemented in the form of software-defined state machines using a set of MAC “commands”. The state machines are executed by an execution engine that runs directly on the radio

hardware. The MAC programmer can schedule operations with a granularity that depends on the B43 chipset clock (88MHz). The MAC programmer can take into account latencies introduced by actions with fixed durations and manually compensate them by scheduling desired actions in advance. WMP also supports real-time code switching as well as MAC multi-threading with negligible delay [10].

SnapMAC [9] provides re-programmability of off-the-shelf IEEE 802.15.4 embedded radio platforms. Due to hardware interrupts, interrupt routines and asynchronous tasks, time-critical behavior is difficult to support on embedded devices. In contrast to the previous mentioned solutions, snapMAC overcomes the timing problem by writing MAC protocols as a chain of time-annotated commands. Unfortunately, timing information has to be added manually to each chain by the developer, which is a time-consuming process and strongly limits the portability and re-usability of the protocols.

2.2.3 Comparison

Table 2.1: Comparison of flexible radio and MAC layer architectures

	Target platform	PHY layer reconfigurability	MAC layer reconfigurability	Accurate time control
Iris [5]	SDR	Yes	At run-time	No
GNU radio & Click [6]	SDR	Yes	Offline	Yes
Wireless MAC Processor (WMP) [8, 10]	Off the shelf IEEE802.11	No	At run-time	Partly (manual)
snapMAC [9]	Off-the-shelf IEEE802.15.4	No	At run-time	Yes
TAISC	Cross-platform	No	At run-time	Yes

Table 2.1 summarizes the major differences between the discussed architectures and indicates how TAISC differs from these. TAISC reuses several innovations from previous work, such as the fact that MAC protocols can be written as state machines (i.e. similar to WMP [8]) and the support for accurate time control (similar to snapMAC [9]), but extends these to include a cross-platform compilation phase.

Whereas snapMAC required MAC developers to manually annotate each instruction, in TAISC the timing information of each instruction is exported to an XML-datasheet that can be used by external tools to create and annotate the bytecode for TAISC. We have built a reference compiler that uses this data-sheet to automatically transform a MAC described in a C-like language to time-annotated bytecode for multiple radio chips. The use of time annotated software has previously been proposed for industrial applications such as the automotive control software [11], but to the best of our knowledge this work is the first one to automate the process of generating time annotated instructions for MAC implementations.

2.3 Work-flow for creating cross-platform MAC protocols

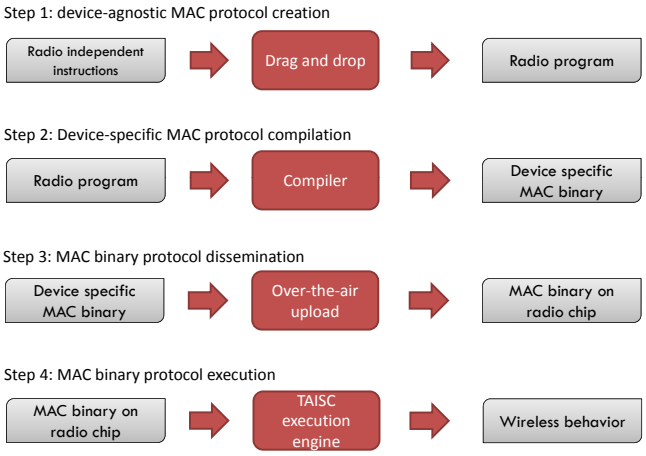


Figure 2.1: Work-flow describing the four steps for creating a TAISC MAC protocol.

The overall TAISC work-flow to develop and execute a MAC protocol consists is illustrated in Figure 2.1.

1. **Step 1: Device-agnostic MAC protocol creation.** First, the MAC protocol designer creates a high-level, device independent radio program to describe the MAC logic using predefined commands in a C-like language, either using high-level C syntax or using a more intuitive drag-and-drop interface. This human readable code consists of a sequence of commands that describe the generic behavior of the MAC protocol and is largely independent of hardware specifics of the final hardware platform.

2. **Step 2: Device specific compilation.** Next, this human-readable sequence is compiled by the TAISC compiler into efficient, device-specific binary bytecode that can be executed by the TAISC execution engine running on the radio platform.

3. **Step 3: Protocol dissemination.** Afterwards, the bytecode is wirelessly transmitted to the target hardware platform and added to the MAC application repository on the local TAISC execution engine.

4. **Step 4: MAC protocol execution.** Finally, the TAISC kernel executes the bytecode.

These 4 steps are discussed in more detail in the remainder of this section.

2.3.1 Step 1: Device-agnostic MAC protocol creation

TAISC is designed to ease the process of developing and reprogramming MAC protocols. During the first step, the programmer can focus on writing human-readable radio programs that are easy and intuitive to create. To this end, MAC protocol design in TAISC requires no knowledge about specific radio chip behavior such as the transition timings (for example the time to switch to a new channel) or radio dependencies (for example the prerequisite conditions before a packet can be transmitted).

The high-level logic of a MAC protocol is described in a **radio program** that consists of a sequence of instructions for the radio platform. Radio programs are created using a simple C-like language to describe the desired radio behavior.

The syntax of such a description is C-compatible but uses the following TAISC specific components:

- **Instructions**. An instruction describes a single action of the TAISC engine. Instructions are implemented in TAISC as function calls and are declared in the *taisc.h* header.

- **Registers.** Developers can define their own variables, but a number of often used variables are offered by default to TAISC protocol developers. A list of these registers is shown in Table 2.2.

- **Events.** A MAC protocol can react to external impulses by making use of events. Events describe impulses that can be triggered by both hardware (such as the radio) qnd software (such as the network stack). A radio program can wait for one of these triggers before continuing execution or skip several instructions by using if statements. An example of the available events on the RM090 platform are shown in Table 2.3.

Radio programs can include variables that will be allocated in the TAISC memory space and conditional execution is supported by using if-statements as well as event triggers. As an example, Listing 2.1 shows a short code fragment for the back-off calculation from a CSMA/CA MAC protocol.

Table 2.2: Overview of TAISC registers. Similar to the registers one can find in micro controllers these registers are volatile and are updated by the TAISC modules.

Register	Description
currentRxFrame	Pointer to the current received frame
currentTxFrame	Pointer to the current frame to transmit
startOfFrameTimestamp	Start of frame timestamp
endOfFrameTimestamp	End of frame timestamp
mediumChangedTimestamp	Time when medium changed
uptimeTimestamp	Uptime

Table 2.3: Overview of TAISC events. These events can be used for the implementation of conditional radio programs.

Event	Source	Description
txFrameAvailable	dataplane	Frame ready to transmit
rxFrameAvailable	dataplane	Frame received
startOfRxFrame	radio	Start of a receiving frame has been detected
endOfRxFrame	radio	End of a receiving frame has been detected
startOfTxFrame	radio	Start of a transmitting frame has been detected
endOfTxFrame	radio	End of a transmitting frame has been detected
mediumIsIdle	radio	Medium is idle
mediumIsBusy	radio	Medium is busy
mediumStateChanges2Idle	radio	Medium changed to idle
mediumStateChanges2Busy	radio	Medium changed to busy

```
1  //backOff = random(2^BE - 1) * slotTime
2  calcBackOff(backOff, BE, slotTime);
3  if(waitForTrigger(backOff, mediumIsBusy | mediumStateChanges2Busy
       | rxFrameAvailable)){
4   if(waitForTrigger(worstCaseWait4RXFrame, rxFrameAvailable)){
5    rxTrigger(1);
6   }
7   // Current number of backoffs performed (NB) is increased by one
8   add(1, NB, NB, 1, _noMASK1);
9   // Current backoff exponent (BE) is increased by one unless it
        has already reached the maximum BE
10  add(1, BE, BE, 1, macMaxBE);
11  if (check(1, NB, maxRetries, _noMASK1, CHECK_EQUAL)) {
12   txTrigger(0); //Transmit failed Release currentTxFrame
13  }
14  stop(1); //asap radio program restart
15 } //medium clear, we can transmit
16 txFast((*(TAISC_RAM.currentTxFrame)),10,0);
```

Listing 2.1: MAC CSMA/CA code example: use of the back-off instruction

2.3.2 Step 2: Device-specific MAC protocol compilation

Once the protocol logic has been described in high-level C language, the TAISC compiler translates the MAC logic into optimized, time-annotated device-specific binary code. Since the exact sequence of low level instructions (and their timing) that is necessary to realize the MAC time constraints depends on the hardware platform, the compiler translates the same radio program into different optimized binaries depending on the target radio hardware platform.

The compilation process consists of the following phases, which are described in more detail in the subsequent sections.

- **Translation**. In this first step lexical correctness of the MAC description is checked. To this end, a Clang [12] script parses the high-level C radio programs and *performs syntax checking*.

- **Flow verification**. Next, high-level instructions are translated into the necessary subinstructions for the target radio platform, thereby creating a low-level model of the MAC protocol logic.

- **Time annotation**. Once the model is verified the compiler will annotate each instruction with the exact timing information required by the TAISC core.

- **Byte code generation**. Based on the model and annotations the compiler generates bytecode that can be run by the TAISC execution engine.

2.3.2.1 Translation

To facilitate the compiler process, a global repository is used that contains additional information about each target platform, including information about the supported radio instructions. An example is shown in Figure 2.2 for the "set channel" instruction. Each instruction includes a description, information regarding the configurable parameters, timing information, platform specific prerequisites and implementation code. This information is used by the TAISC compiler when compiling a MAC protocol for a specific target. To coordinate all tools involved in this process, a global eXtensible Markup Language (XML) file is used as a source for all configuration parameters. As such, whenever a new platform needs to be supported, the XML file needs to be updated with the specifics of the new platform. The XML file also provides a single location for the automatic generation of documentation.

To automatically process the radio programs, a Clang [12] front-end is created that processes the MAC description and generates a structural representation. This representation is used by the compiler to create an abstract model of the radio

	Explanation	Example
Description	A high level description describing the command	Change the operating channel of the transceiver
Parameters	Some commands require additional information to perform their task	Channel number
Timing	Time required to execute the command and for changes to take effect	123 µs
Prerequisites	Prerequisites that must be fulfilled if the command is to be executed	Transceiver must be on and idle
Implementation	The actual code that must be executed to achieve the results mentioned in the high level description	spi_transmit(WRITE_REG) spi_transmit(REG_CHANNEL) spi_transmit(channo)

Figure 2.2: Example instruction: set channel

programs. This model is further verified by the compiler to ensure that there are no static inconsistencies in the MAC description such as invalid state transitions.

2.3.2.2 Flow verification

During the next phase of the compilation process, the generic MAC model is translated into a sequence of platform specific instructions. Depending on the target platform, a high level instruction can require multiple radio configuration instructions. For example, turning the radio on from off for a CC2520 radio requires the following subinstructions: (i) check the voltage regulator, (ii) start the oscillator, (iii) check the oscillator, (iv) initialize low power mode. However, other hardware platform might require a different sequence of low level instructions to turn the radio on. The compiler utilizes the prerequisites information from the XML repository to derive a platform specific sequence of low-level instructions for each of the high-level radio program instructions.

2.3.2.3 Time annotation

Crucial to the operation of TAISC is the possibility to enact radio instructions at very exact times. To optimally use the spectrum and to minimize the radio energy consumption, the compiler has provisions for scheduling specific radio instructions using exact timestamps. For example, for TDMA protocols the radio should be set to active on a specific radio frequency exactly at the beginning of each active time slot. To this end, the high-level C language for MAC protocol creation (see Section 2.3.1) allows the definition of 'reference times', indicating that an instruction should be executed at a specific time in the future (indicated by the 'sync' command on Figure 2.3).

Since the sequence of prerequisite low-level instructions was already calculated during the previous compilation phase, the timing information from the XML

is used to calculate how much in advance the preceding instructions should be scheduled to meet the timing constraints, resulting in a sequence of 'time annotated instructions'. Figure 2.3 illustrates this process. In this example, the loadframe() command and the setChannel() commands require respectively 178μs and 10μs, meaning that the radio program needs to start 188μs before the reference time.

```
setChannel(channel);        → takes 10 µs
loadFrame(0, 0);            → takes 178 µs
sync();                     → reference
tx();                       → takes 192 µs

→Execution of setChannel will start 188 µs before the reference time
```

Figure 2.3: The sync() instruction is used to indicate that an instruction needs to be executed at an exact time in the future. Both instruction timing and sequence information are used to annotate each radio program.

To calculate the instruction duration, four different duration influences are taken into account in the XML instruction repository.

- **Static execution**. Commands such as switching the radio off always take a fixed amount of time to execute.
- **Dynamic execution**. The duration of dynamic commands is a function of the used command variable. For example, the loadFrame command will have different timings depending on the size of packet frame.
- **Static transition**. Some commands require a transition before execution. For example, starting up the oscillator of the radio consists of a static amount of execution time and a static transition time that the oscillator needs to settle.
- **Dynamic transition**. Similarly, some commands require a transition time that depends on the current configuration of the radio. For example, the time needed to transmit a frame will depend, amongst others, on the PHY bit rate and preamble size variables.

The overall instruction duration can consist of multiple of these aspects. For example, the time to transmit a frame that is already loaded consists of a static execution time to switch the radio into Transmit (TX) mode, a static transition time to transmit the preamble, and a variable transition time to transmit the bytes of the frame. To cope with dynamic durations, the XML repository time stores the time needed to handle one unit (byte), which is then multiplied by the overall frame size.

Due to their hardware dependent nature, the timing information of all instructions is stored per platform in the TAISC library. Updating the timing information of the XML file to support a new hardware type can be done using information from the datasheet, or by performing automated benchmarks that measure the duration of instructions under different conditions, such as different parameter settings.

2.3.2.4 Bytecode generation

Finally, the time annotated model is transformed into compact bytecode using a python script. This script generates bytecode consisting of three data blocks.

- RAM: this block contains the dynamic variables of the user-defined MAC protocol. Upon loading the bytecode this data is copied into the TAISC RAM where it resides together with any registers defined by the TAISC execution engine itself.
- Instruction code: the second block contains the code corresponding with all the instructions in the radio program.
- Radio program configuration: the final data block has an entry for each radio program in the system and contains the actual location of the instruction code and start time of each radio program.

```
0x0e,0x56,0x00,0x00,0x06,0x00,0x00,0x05,0x00,0x04,0x13,0x00,
```

Listing 2.2: Example bytecode representation of a function for calculating the current back-off window.

An example of native bytecode for a single instruction of a compiled CSMA/CA protocol for a CC2520[1] radio chip is shown in Listing 2.2. The line is the result of a calcBackOff instruction. The first byte represents the instruction id 0x0e. Next, a 2 byte time annotation is stored. The next byte 0 represents direct RAM addressing, whilst the next two bytes (0x0006 in little endian representation) point to the address in Random-Access Memory (RAM) where the variable backOff is stored. Due to the compactness of the bytecode, the resulting bytecode for the whole CSMA/CA bytecode is only 459 bytes in Read-Only Memory (ROM) and 10 bytes in RAM.

2.3.3 Step 3: TAISC protocol dissemination

By loading this bytecode on a TAISC enabled radio chip the target platform will now be able to support the desired MAC protocol. Disseminating the bytecode is

[1] https://www.ti.com/product/cc2520

done using the GITAR architecture [13] which allows automatically managing, updating and upgrading the software on embedded devices such as micro-controllers. Once the bytecode is disseminated through GITAR, it is handed over to the TAISC execution engine running on the micro-controller of the radio chip.

2.3.4 Step 4: TAISC protocol execution

After receiving the new bytecode, the TAISC execution engine switches out the previous bytecode and starts executing the new MAC protocol bytecode consisting of one or more radio programs. To ensure timely execution of the time-annotated commands, the TAISC execution engine operates in highest priority context. To this end, the engine resides as close as possible to the radio itself (see Figure 2.4) either on a dedicated controller (e.g. for Wi-Fi plug-in cards) or on the micro-controller of an embedded device (e.g. for sensor nodes). Since the bytecode requires no further interpretation, the responsibilities of the kernel are limited to ensuring the scheduling of the pre-calculated sequences of radio instructions, thereby limiting the complexity of the TAISC execution engine.

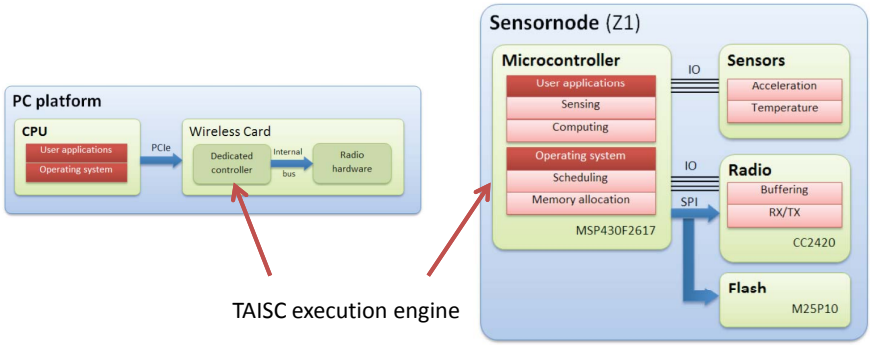


Figure 2.4: Situation of the TAISC execution engine close to the radio hardware for a wireless plug-in card (left) and for an embedded sensor device (right).

2.4 TAISC architecture

This section will describe the TAISC architecture.

2.4.1 TAISC execution engine

Figure 2.5 illustrates the TAISC execution engine that resides on the radio chip and executes the bytecode. The *TAISC RAM unit* (Figure 2.5, left side) contains dynamic information, including (i) radio program management (keeps all the radio programs consistent regarding their ROM and RAM boundaries and stores

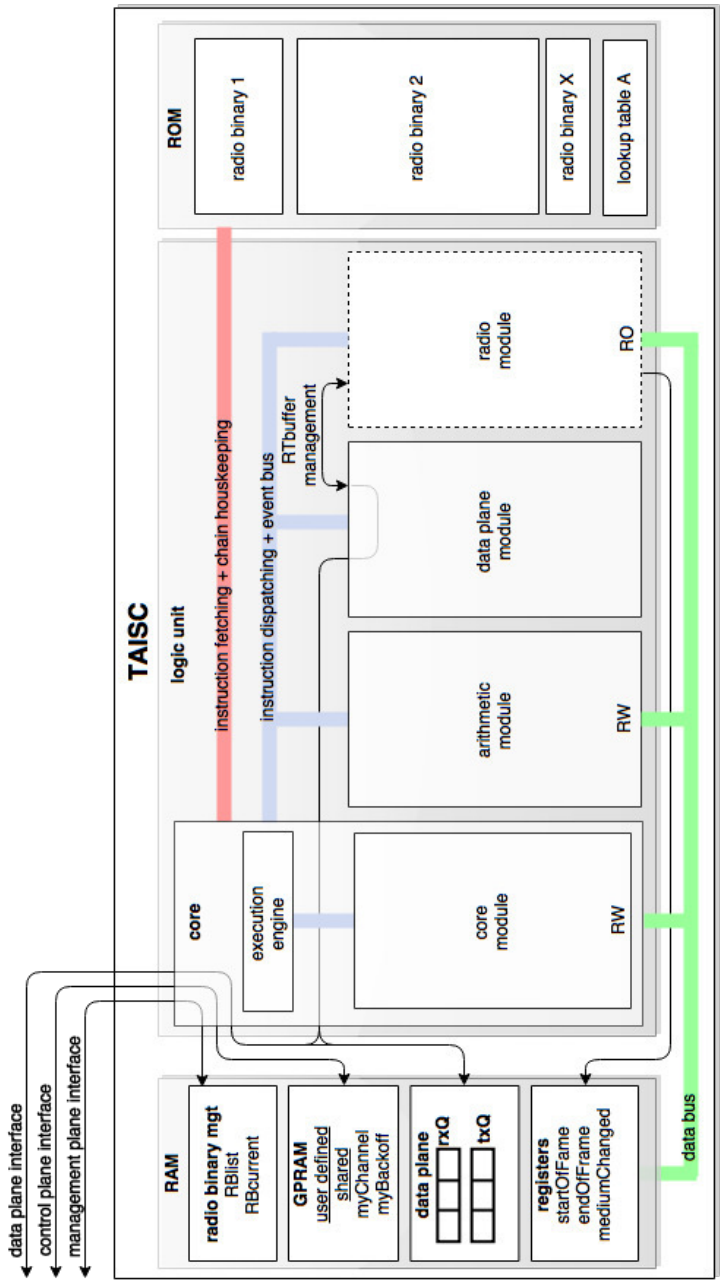


Figure 2.5: Overview of the TAISC architecture. Left: RAM memory blocks. Middle: logical processing unit. Right: ROM memory.

which radio program is currently active), (ii) General Purpose Random-Access Memory (GPRAM) for storing dynamic radio program variables, (iii) access to the receive / transmit transmission queue, and (iv) TAISC registers that contain radio specific status information, such as the current radio frequency (see Section 2.3.1). The *TAISC ROM unit* (Figure 2.5, right side) stores static information, including one or more compiled radio programs as well as their static information such as lookup tables (for example to retrieve the radio frequency to use in each slot for a hopping based MAC protocol). Finally, the *TAISC logic unit* (Figure 2.5, middle) forms the core of the architecture. Its most important component is the execution engine that implements a scheduler which manages the execution times of the instructions, dispatches the instruction to the correct module, processes incoming events and schedules which instruction or radio program to fetch next. Instructions are executed by the core module (that a.o. manipulates the program counter for reacting to triggers or conditional events), the arithmetic module (supporting operations such as copy, add, subtract, etc.), the data plane module (that a.o. manages access to and from the data plane and manages conflicts between different modules) and the radio module (implementing chip specific radio implementations). Finally, TAISC provides three *upper layer interfaces* (Figure 2.5, upper left): (i) the data plane interface interacts with incoming / received packets from higher layers, (ii) the control interface provides higher layers access to the radio program specific variables from the GPRAM and (iii) the management plane interface provides functionality to upload and/or activate new compiled MAC protocols.

2.4.2 TAISC scheduler

The scheduler is responsible for timely execution of time-annotated instructions.

As discussed in Section 2.3.2 and shown in Figure 2.3, the sync instruction is used to indicate to TAISC that an instruction should be executed at a specific time in the future. Since any type of instruction could precede the sync instruction we need strict time information of all the instructions, otherwise we limit the design options of MAC developers. Instructions that are executed after the sync instruction are still annotated, but the scheduler allows more leeway in the actual execution time since the time critical execution moment has already been executed.

For many synchronized MAC protocols, it is important that execution time of specific instructions is met exactly, e.g. for the generation of synchronization beacons. In practice, variability of the executed commands can happen because of several reasons: (i) the presence of interrupts, (ii) the variability in execution time for each instruction and (iii) the clock frequency and drift. TAISC reduces this variability through several scheduler optimizations.

1. **Higher priority context.** The TAISC scheduler runs in a higher priority

context than the applications. As such, other software programs can not interrupt the scheduler, whereas the scheduler can interrupt applications or higher-layer protocols.

2. **Well-known execution times.** A TAISC radio program exists out of small instructions which are implemented as short atomic blocks. These blocks are executed inside a timer Interrupt Service Routine (ISR) that cannot be interrupted, and hence have fixed execution times. In these atomic blocks, each instruction has a well-known associated execution time. For instructions with variable execution times, the exact latency is calculated depending on the configuration parameters (e.g. see Section 2.3.2.3). For example, for a CC2520 radio, when a packet with known length is transmitted, the scheduler takes into account a dynamic execution time of 32μs per byte. For instructions from which the configuration parameters are not known in advance, a worst case scenario is assumed. This way, the radio program is loaded sufficiently in advance to ensure that the sync instruction can be executed exactly at the requested time.

3. **Interfering interrupts.** In case TAISC shares the processor with other peripherals (serial ports, ADCs,...), interrupts could still interfere with the controlled execution environment of TAISC. To cope with these interrupts, and thus provide timing guarantees, several options are available to the MAC developer. (i) One possibility is to configure TAISC to allow serial communication only when the radio is in sleep modus. In other words, the non TAISC ISRs become secondary users on the shared Central Processing Unit (CPU). The performance evaluation in Section 2.5.3 utilizes this method, whereby the scheduler allows serial communication only during TDMA slots where the radio is in sleep modus. (ii) On a shared CPU non-TAISC related ISRs can result in jitter on the start time of the TAISC instructions. By reducing the execution time of all the ISRs this jitter can be reduced. (iii) Furthermore we can prioritize the TAISC ISR to ensure it can interrupt ISRs with lower priority. (iv) Finally, by use of a System on Chip (SoC) processor, TAISC can run on a dedicated core and is hence not influenced by interrupts.

4. **Clock drift.** Finally, the clock also exhibits clock drift. Since this variation is hardware dependent, this variation can only be reduced by utilizing more accurate timer crystals. MAC developers need to take into account the clock drift when designing MAC protocol, e.g. by repeatedly sending synchronization beacons.

2.5 Performance evaluation

This section evaluates the performance of the TAISC execution engine and the performance of several implemented MAC protocols.

2.5.1 TAISC execution engine: memory overhead

To execute time annotated MAC binaries, the TAISC architecture needs to be installed on the processor controlling the radio chip. The architecture has been implemented on top of a MSP430F5437 micro-controller and the CC2520 IEEE802.15.4 radio integrated on the RM090 embedded platform [14].

Table 2.4: Overview of resource usage of the TAISC architecture.

object	ROM (Bytes)	RAM (Bytes)
core module	3271	86
radio program housekeeping (4)	0	36
registers (4)	0	16
arithmetic module	1111	8
dataplane module	960	0
dataplane buffers (rxQ and txQ)	0	276
cc2520	5006	26
total	**10348**	**470**

The architecture is available as a generic, platform independent C-library optimized for 16 bit micro-controllers. Since most embedded sensor nodes are more limited in RAM memory than in ROM memory [15], the implementation has been optimized for minimal RAM usage. Overall, the implementation of the TAISC architecture requires 10kB ROM and 500B RAM (Table 2.4), which is well within the limits of typical embedded devices. Of the TAISC architecture depicted in Figure 2.5, only the radio module is radio chip specific. As a result, porting the architecture to a new platform is limited to implementing the radio module block. Overall, 50 % of the C-library is radio chip specific, allowing fast portability of the TAISC execution engine towards new radio platforms.

2.5.2 TAISC execution engine: scheduling overhead

The minimum time to schedule an instruction depends on several factors: the complexity of the scheduling algorithm, the clock speed and the presence of interrupts. We kept the complexity of the algorithm as simple as possible and disable interrupts during the time critical part of the scheduling. As a result, scheduling a new instruction can be done in at less than 10μs. Although we can further reduce this scheduling overhead, the current implementation is very conservative and assumes a fixed 20μs interval dedicated for scheduling between each instruction. For duty

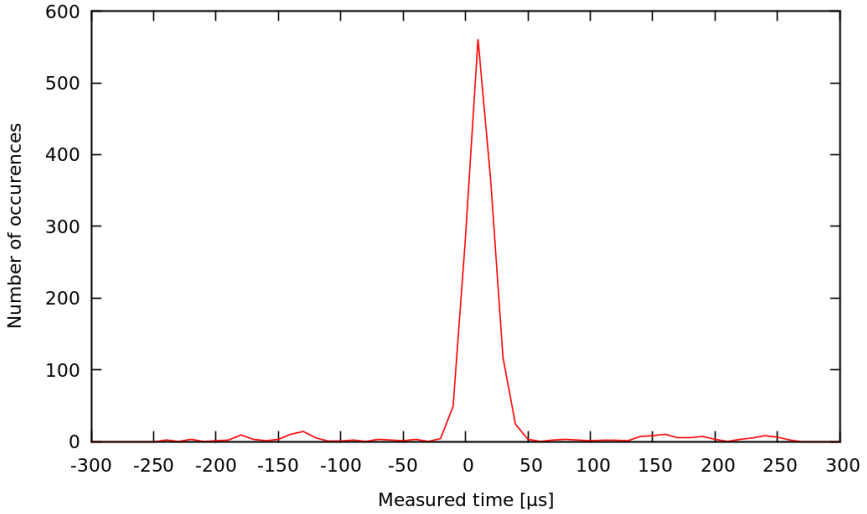


Figure 2.6: Variability of the scheduled operation of TAISC instructions. A beacon frame is transmitted every 60 milliseconds. After 60 milliseconds, the execution of the instruction has an average deviation of 9μs with a standard deviation of 52μs. Note that the platform in use (RM090) had an unstable clock, due to issues with the external crystal, hence the large spread in the graph.

cycled devices, further energy savings could be realized by reducing this interval towards lower values.

2.5.3 TAISC execution engine: scheduling variability

To show the variability in the fixed execution of instructions a simple experiment using the TDMA protocol has been performed. Sensor nodes with an msp430f5437 processor were used with a 16MHz clock (FLL functionality with 32000Hz crystal reference clock and DCO divider 500). One sensor node sends a beacon every 60000μs. Another sensor node listens for these beacons and logs the time difference between each two consecutive beacons using the timing information available from TAISC. The resulting graph shows that the average interval that has been registered is 60009μs (i.e 9 μs deviation every 60 seconds) with a standard deviation of 52μs. However, it is important to note that this variability is not inherent part of the TAISC architecture but depends the stability of the clock: a more accurate clock will produce a more stable result. It must be noted that the clock used for the experiment was a very unpredictable one - due to issues with the external crystal - hence there is a lot of variability in the execution timings. However, thanks to the time annotated nature of the TAISC, the execution engine can cope with this variability by taking into account the worst case execution timings.

2.5.4 MAC protocol evaluation

To evaluate the effectiveness of the compilation process, several MAC protocols have been implemented and compiled for TAISC: (i) a maximum throughput protocol, (ii) Low Power Listening (LPL), a non-synchronous MAC protocol for energy constrained device [16], (iii) a CSMA/CA protocol [17], and (iv) a latency-bound TDMA protocol for time-critical and reliable operation ('robotTDMA'). The latter protocol was developed in house for the management of (up to 28) mobile robots that can move simultaneously in a wireless test facility w-iLab.t[2]. Three access points are used to ensure full coverage of the test-lab area of 22.5m by 60m, whereby each access point operates on a different radio frequency to avoid packet loss due to interference. To control the robots simultaneously with sufficiently high accuracy, access points need to reliably transmit up to 90 movement commands per second, thus imposing strict synchronization requirements. To this end, a TDMA protocol for IEEE 802.15.4 radios has been designed with a super-frame structure of 100 ms consisting of 32 dedicated slots of 3.125 ms each, and a number of empty slots that can be used for free, contention based access. At the beginning of each super-frame, a synchronization beacon is transmitted by each access point. The synchronization beacon has a dual function: (1) support of mobile hand-overs between the three different access points, and (2) time synchronization of the mobile robots within 4 μs.

The use of TAISC protocols imposes a very limited impact on the overall *throughput*. The theoretical maximum throughput for an IEEE 802.15.4 MAC protocol is 225 kilobit per second (kbps). Due to scheduling overhead of the TAISC implementation, practical throughputs will be lower. However, due to code optimization and the use of non-blocking implementations, the processing overhead per instruction by the TAISC execution engine is limited to 20 μs on a 16MHz microcontroller, allowing fine-grained execution of radio instructions. When transmitting packets with 125 bytes payload (the maximum IEEE 802.15.4 MAC payload), throughputs up to 218 kbps can be achieved, which corresponds to 97% of the theoretical upper bound on single hop throughput. The resulting bytecode is very efficient and takes a mere 114 bytes of ROM and a single byte of RAM, making it possible to distribute the MAC bytecode in just 1 IEEE 802.15.4 frame.

An important aspect for reconfigurable MAC protocols is to replace standardized, hardware implemented behavior by more flexible software solutions. A typical example is the *generation of Acknowledgment (ACK) messages*, which are typically generated by the hardware chip. By creating software ACKs, protocol designers can introduce optimizations such as piggybacking status information (for example link estimations) on ACKs, or even modify the acknowledgment behavior to acknowledge a series of packets rather than individual packets. Thanks to

[2]https://www.wilab2.ilabt.iminds.be/

the low overhead of the TAISC execution engine, software acknowledgments can be generated in 612 μs, which is within the time limits of 864 μs imposed by the IEEE 802.15.4 Physical Layer (PHY) standard.

In addition to low processing overheads, compiled MAC protocols should have a low *memory overhead*. This is important for several reasons. (i) Compiled binary protocols need to be transmitted wirelessly to the target platform. Reducing their overall size reduces the impact of this update process on the overall wireless performance, especially in multi-hop networks of embedded devices. (ii) Secondly, typical microprocessors are limited in terms of available memory. The memory of lower end networked microcontrollers is typically in the order of 48-256 kB ROM and 4-16 kB RAM. Table 2.5 gives an overview of the RAM and ROM requirements of several MAC protocols. The overhead is low enough to allow several MAC protocols to be installed in the ROM memory of a typical sensor node, allowing dynamic switching between MAC protocols depending on the current application requirements. Even for the low latency TDMA protocol, which is far more complex than the other MAC protocols, the binary size is limited to 1568 bytes of ROM and 83 bytes of RAM.

Finally, the *stability* of the TAISC framework has been evaluated through longterm testing of the TDMA protocol for robot control. The protocol has been running in the testlab for more than one year without any communication failures.

Table 2.5: Memory overhead for different MAC protocols.

	ROM (bytes)	RAM (bytes)
Max throughput	114	1
LPL [16]	557	9
CSMA/CA [17]	460	10
Low latency TDMA (robotTDMA)	1568	83

2.6 Conclusions

Although a variety of MAC protocols optimized for different conditions or network requirements exists, nowadays MAC protocols can not be reused on other hardware platforms. However, for many IoT applications, (embedded) devices need to remain operational for 10+ years after deployment and need to operate efficiently during this time. These devices typically require low-level software updates for bug fixing, upgrading to newer standards, to adapt to changing network conditions or to adapt to changing application requirements. However, current MAC protocols are designed for one specific target platform, limiting reuse of MAC protocols and hindering system integrators from freely switching between

different radio manufacturers. As such, there is a need for cross-platform MAC protocols, allowing network managers to choose amongst a wide range of existing MAC protocols the ones that are best suited for current conditions.

In this chapter we have demonstrated that TAISC, a novel framework hardware independent MAC protocol development and management, is capable to simplify the development of new MAC protocols by providing a hardware independent language consisting of high level radio commands without losing their real time aspect. The main advantages of TAISC are (i) its intuitive MAC protocol design, (ii) its capabilities for cross-platform compilations, thereby promoting MAC protocol portability, (iii) its capability of over-the-air MAC upgrades and (iv) accurate radio control due to its inherent time-awareness. By using timing information for all instructions TAISC MAC protocols can automatically be fine-tuned for optimal energy consumption and spectrum efficiency. Unlike existing solutions such as Iris and WMP, TAISC is designed to run on very constrained hardware and introduces time-awareness for the execution of protocols. This means TAISC is able to guarantee the exact time of transmission of packets, while minimizing the radio-on time. Typical MAC protocols for TAISC can be compiled to less than 2 kB. Although small in size, they can achieve performances matching the ones from custom-designed protocols, reaching throughputs up to 97% of the theoretical maximum and achieving latencies low enough to generate software ACKs within the IEEE 802.15.4 standard time bounds.

To conclude, similar to the success of smartphone and tablet app-stores, in which generic hardware is used as an enabler for a wide range of purposes, we envision a future in which also radio chips can be updated and can switch between multiple MAC protocol implementations depending on their current needs. TAISC can be seen as a first step in this direction by allowing the generation of radio-independent MAC protocol code that can reside in a radio-app store, and which can efficiently be compiled towards a wide range of target radio platforms.

Acknowledgments

This material is based in part upon work supported by the IWT project SAMURAI (Software Architecture and Modules for Unified RAdIo control) and by the European Commission Horizon 2020 Programme under grant agreement n 688116 (eWINE) and grant agreement n 645274 (WISHFUL).

References

[1] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste. *Enabling MAC Protocol Implementations on Software-defined Radios*. In Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09, 2009. Available from: http://dl.acm.org/citation.cfm?id=1558977.1558984.

[2] E. Chukwuka and K. Arshad. *Energy Efficient MAC Protocols for Wireless Sensor Network: A Survey*. Computer Research Repository, 2013. Available from: http://arxiv.org/abs/1309.2690.

[3] B. Wang and K. Liu. *Advances in cognitive radio networks: A survey*. Selected Topics in Signal Processing, IEEE Journal of, 5(1):5–23, Feb 2011. doi:10.1109/JSTSP.2010.2093210.

[4] M. Chowdhury, Asaduzzaman, and M. F. Kader. *Cognitive Radio MAC Protocols: A Survey, Some Research Issues and Challenges*. Smart Computing Review, 2015. Available from: http://dx.doi.org/10.6029/smartcr.2015.01.003, doi:10.6029/smartcr.2015.01.003.

[5] P. Sutton, J. Lotze, H. Lahlou, S. Fahmy, K. Nolan, B. Ozgul, T. Rondeau, J. Noguera, and L. Doyle. *Iris: an architecture for cognitive radio networking testbeds*. Communications Magazine, IEEE, 2010. doi:10.1109/MCOM.2010.5560595.

[6] R. Dhar, G. George, A. Malani, and P. Steenkiste. *Supporting Integrated MAC and PHY Software Development for the USRP SDR*. In Networking Technologies for Software Defined Radio Networks (SDR 06), 2006. doi:10.1109/SDR.2006.4286328.

[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. *The Click Modular Router*. ACM Trans. Comput. Syst., 2000. Available from: http://doi.acm.org/10.1145/354871.354874, doi:10.1145/354871.354874.

[8] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli. *Wireless MAC processors: Programming MAC protocols on commodity Hardware*. In INFOCOM, 2012 Proceedings IEEE, pages 1269–1277, March 2012. doi:10.1109/INFCOM.2012.6195488.

[9] P. D. Mil, B. Jooris, L. Tytgat, J. Hoebeke, I. Moerman, and P. Demeester. *snapMac: A generic MAC/PHY architecture enabling flexible MAC design*. Ad Hoc Networks, 2014. Available from: http://www.sciencedirect.com/science/article/pii/S1570870514000158, doi:http://dx.doi.org/10.1016/j.adhoc.2014.01.004.

[10] G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, F. Gringoli, and I. Tinnirello. *MAClets: Active MAC Protocols over Hard-coded Devices*. pages 229–240, 2012. Available from: http://doi.acm.org/10.1145/2413176.2413203, doi:10.1145/2413176.2413203.

[11] P. Derler. *Efficient Execution and Simulation of Time-Annotated Embedded Software*. Phd dissertation, September 2010.

[12] *Clang: a C language family frontend for LLVM*.

[13] P. Ruckebusch, E. D. Poorter, C. Fortuna, and I. Moerman. *GITAR: Generic extension for Internet-of-Things ARchitectures enabling dynamic updates of network and application modules*. Ad Hoc Networks, 2015. Available from: http://www.sciencedirect.com/science/article/pii/S1570870515001225, doi:http://dx.doi.org/10.1016/j.adhoc.2015.05.017.

[14] RM090. *http://www.rmoni.com/en/products/hardware/rm090*.

[15] EU-FP6 Embedded WiSeNts Project. *Report 2.1: Critical evaluation of research platforms for wireless sensor networks*. 2007.

[16] C. Merlin and W. Heinzelman. *Duty Cycle Control for Low-Power-Listening MAC Protocols*. Mobile Computing, IEEE Transactions on, 2010. doi:10.1109/TMC.2010.116.

[17] *Application Note: JN-AN-1035. Calculating 802.15.4 Data Rates.*

3

Portability, compatibility and reuse of Medium Access Control (MAC) protocols across different Internet of Things (IoT) radio platforms

In the previous chapter we focused on portability of MAC protocols across a variety of hardware devices. However due to hardware and timing differences the same protocol can behave differently per platform resulting in unpredictable or asymmetrical behavior when deployed into a single heterogeneous network. This chapter extends on the previously achieved portability in Time Annotated Instruction Set Computer (TAISC) by offering a three step workflow to avoid any cross-platform compatibility issues, and thereby ensures fairness between all devices within the network.

⋆⋆⋆

J. Bauwens, B. Jooris, S. Giannoulis, I. Jabandžić, I. Moerman, and E. De Poorter.

Published in Elsevier Ad Hoc Networks, Nov. 2018.

Abstract To cope with the diversity of IoT requirements, a large number of MAC protocols have been proposed in scientific literature, many of which are designed

for specific application domains. However, for most of these MAC protocols, no multi-platform software implementation is available. In fact, the path from conceptual MAC protocol proposed in theoretical papers, towards an actual working implementation is rife with pitfalls. (i) A first problem are code bugs related to execution timings, frequently encountered in MAC implementations. (ii) Furthermore, once implemented, many MAC protocols are strongly optimized for specific hardware, thereby limiting the potential of software reuse or modifications. (iii) Finally, in real-life conditions, the performance of the MAC protocol varies strongly depending on the actual underlying radio chip. As a result, the same MAC protocol implementation acts differently per platform, resulting in unpredictable/asymmetrical behavior when multiple platforms are combined in the same network. This chapter describes in detail the challenges related to multi-platform MAC development, and experimentally quantifies how the above issues impact the MAC protocol performance when running MAC protocols on multiple radio chips. Finally, an overall methodology is proposed to avoid the previously mentioned cross-platform compatibility issues.

3.1 Introduction

In the fast growing world of IoT devices, wireless sensor networks are deployed in increasingly diverse application domains, ranging from smart homes to factories of the future [1]. Each of these domains have inherently different application requirements such as throughput, battery lifetime, reliability, etc. Wireless network designers need to take into account the trade-offs between these performance metrics. As an example, when using wireless IoT devices to replace wired control loops in industry processes, the network protocols should focus on providing low latency and high reliability, whereas temperature monitoring applications often emphasize long network lifetime requirements.

To this end, an important design decision is the choice of the MAC protocol, which manages how and when the wireless medium is accessed. Countless protocols have been designed with advantages and disadvantages regarding different performance metrics, making it challenging to make an informed decision about the optimal protocol [2]. Some architectures even load several protocols on the device, in order to select the optimal at runtime [3]. For example the Time Synchronized Channel Hopping (TSCH) MAC protocol is designed for reliability and low battery usage but has high jitter (deviation of the inter-arrival time between packets) [4], whereas using Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) results in low jitter and high throughput but a higher battery consumption [5]. Many open-source implementations of these MAC protocols contain performance limitations. (i) They are often designed for one specific hardware platform (for example the CC2420-radio) and thus can not be reused for other

radio chips. (ii) Conversely, many IoT operating systems only support a single MAC implementation (e.g. RIOT OS only supports CSMA/CA [6]). (iii) Finally, in case multi-platform support is available, the MAC performance typically degrades due to the use of a feature-poor hardware abstraction layer (e.g. Contiki [7] and openWSN [8]). This limited pool of implemented MAC protocols per radio/-operating system combination hinders MAC innovations since researchers can not directly use the desired MAC protocol on their development systems. The alternative, implementing the MAC logic from scratch or porting an existing MAC implementation to a new platform, requires extensive knowledge of the target Operating System (OS) and radio chip. Even then it takes a significant amount of time to port a protocol to new platforms, since MAC protocols interact with low level hardware components and as a result have a large and difficult to understand code base. An improvement would consist of implementing MAC protocols once in a hardware independent language, and reuse the code on all desired platforms. These multi-platform protocol implementations could be made available on a MAC protocol cloud storage [9].

Unfortunately, even if devices are able to run the same MAC protocol code, hardware differences might still subtly alter the behavior of a MAC protocol compared to the one running on other radio platforms or legacy devices. For example, the stability and speed of the clock might differ resulting in different timings. If the timing differences become too large, this can result in asymmetrical link behavior. In more extreme cases, the MAC protocol performance can degrade or even become incompatible between the different hardware platforms [10].

The existence of asymmetric links can have an effect on the performance of higher layers, as is shown in [11, 12]. Furthermore, [13] demonstrates that the presence of an asymmetrical link performance severely hurts the performance of the Routing Protocol for Low-Power and Lossy Networks (RPL) routing protocol.

In contrast to most scientific papers which focus on the design of novel MAC protocol algorithms, this work proposes a methodology which allows efficient implementation of MAC algorithms on real hardware while simultaneously supporting portability and interoperability between multiple platforms. To this end, this chapter discusses three challenges that influence the performance of MAC implementations running on different hardware platforms.

Challenge 1: The lack of feature-rich Application Programming Interface (API)'s which still support portability. Existing MAC API's or MAC design frameworks do not offer the means to create a protocol which is multi-platform, while still allowing full access to the platform capabilities. In this chapter, a hierarchical layered implementation approach is proposed which makes it possible to trade in portability for performance. Our approach gives the developer an informed choice of either building a protocol which is fully portable, or a protocol which uses low-level network/radio capabilities.

Challenge 2: MAC protocol instruction timings have a strong hardware dependency. MAC protocols contain numerous timers that determine exactly when radio features such as sleep, transmit, Clear Channel Assessment (CCA), etc. should be executed. However, transitions between radio states are often not documented. Many implementations cope with this by either using timings which are too strict due to a lack of understanding of the low-level timing dependencies of the radio chip instructions, thereby introducing unpredictable MAC behavior. Others choose timings which are too large in order to ensure cross-platform compatibility, thereby reducing throughput and reducing energy efficiency. We demonstrate that the burden of tweaking timers should not fall upon the MAC implementer, but rather the MAC compiler should automatically adjust timings based on the platform in use.

Challenge 3: Multi-platform networks exhibit unpredictable behavior. Even when multiple platforms with different radio chips run the same MAC software, issues remain when deploying these different devices in the same network. Unfairness between platforms can occur due to inherent hardware and timing differences. We demonstrate that it is possible to equalize the behavior of a MAC protocol across different platforms to avoid these unpredictable unfairness effects.

The rest of the chapter is structured as follows. First, Section 3.2 gives an overview of related work describing previous approaches aiming to support MAC protocols running on multiple embedded devices and/or operating systems. Next, Sections 3.3, 3.4 and 3.5 experimentally quantify[1] the performance impact for each of the three above challenges and subsequently propose and evaluate a solution using state-of-the-art MAC design approaches. Finally, Section 3.6 concludes the chapter by giving an overview of all previously mentioned solutions.

3.2 Related work

This section discusses state-of-the art approaches for dynamic MAC frameworks which show promise for multi-platform design.

3.2.1 Multi-platform MAC architectures

Although a multitude of IoT platforms currently exist, several problems related to running a MAC protocol on different platforms remain unsolved. The authors of [15] highlight several performance issues with (radio duty cycling) MAC protocols, most of which are caused by incorrect timing implementations. It is worth noting that the correct implementation of timers is identified as problematic, even

[1] All experimental evaluations used the w-iLab.t wireless testbed facilities, which is a state of the art IoT testbed in Europe [14]. The testbed contains large-scale deployments of sensor nodes of different hardware types, which have been used to conduct cross-platform experiments. All conducted experiments were performed in a single-hop star topology.

when implementing a MAC protocol for a single platform. This chapter puts forward recommendations which could prevent most of the identified issues from happening.

In terms of MAC protocol reuse, a number of existing research papers provide a cross-platform MAC implementation by creating an abstraction layer between the MAC and network layer per operating system [16, 17]. They allow the use of a platform specific MAC protocol on multiple operating systems. One can argue that, although minimizing the effort of a MAC protocol running on multiple operating systems, this is not really cross-platform MAC design but rather cross-operating system MAC design.

Several operating systems, e.g. Contiki [7] and Mantis OS [18], provide an interface for physical layer calls, a so called Hardware Abstraction Layer (HAL). Due to the need for simplicity, these API interfaces omit hardware specific information such as data rates, timings, power modes. As a result MAC protocol developers can not utilize more advanced hardware capabilities. Even worse, MAC protocol implementers lack information regarding the exact implementation and timing constraints of the HAL interfaces. For example, the off-function for the TMote Sky platform in Contiki does not inform the user about the required time to execute this command. Moreover, the actual implementation puts the radio in a Low Power Mode (LPM) rather than completely off, resulting in energy loss. An interface needs to be clear and informative, as the developer should not be limited in his possibilities while creating the MAC protocol code.

A more rich interface is provided by the Wireless MAC Processor (WMP), which is a programmable MAC architecture devised to run a MAC protocol defined in terms of a state machine [19]. It offers flexibility to easily create and adapt novel protocol ideas. However, WMP is only available for the AirForce54G chipset from Broadcom and thus does not support cross-platform MAC portability and reusability.

Another MAC design architecture is the TAISC framework, which has a number of advantages compared to traditional MAC protocol development architectures [20]. The main innovation of MAC was the introduction of time-annotated radio instructions. The compiler adds exact timing information which can be used by an execution engine for instruction scheduling. Although the authors hint that this approach could help to support multi-platform compilation, this claim is not further explored or experimentally validated.

In conclusion, several multi-platform MAC protocol architectures already exist. Unfortunately, most of them only offer a HAL, and do not solve other possible multi-platform issues. In this chapter, the TAISC concept of radio instruction time annotation is utilized and extended to solve the multi-platform problems stated by this chapter.

3.2.2 State-of-the-art low power MAC protocols

To quantify the impact of multi-platform design on the MAC performance, this chaper will use two different types of low-energy MAC protocols: the contention-based ContikiMAC and the Time Division Multiple Access (TDMA)-based TSCH MAC protocol.

ContikiMAC is a frequently used Carrier Sense Multiple Access (CSMA)-based protocol. From [21]: *"ContikiMAC is a radio duty cycling protocol that uses periodical wake-ups to listen for packet transmissions from neighbors. If a packet transmission is detected during a wake-up, the receiver is kept on to be able to receive the packet. To transmit a packet, a sender repeatedly sends its packet until it receives a link layer acknowledgment from the receiver."* Due to its complex time dependent nature and the fact that it is available for multiple platforms, ContikiMAC represents an ideal protocol for evaluation.

In addition, we also evaluate an implementation of IEEE 802.15.4e TSCH, a standardized Multi Frequency Time Divison Multiple Access (MF-TDMA) variant that is dominant in TDMA MAC protocols for wireless sensor networks. From [4]: *"Through time synchronization and channel hopping, TSCH enables high reliability while maintaining very low duty cycles."*

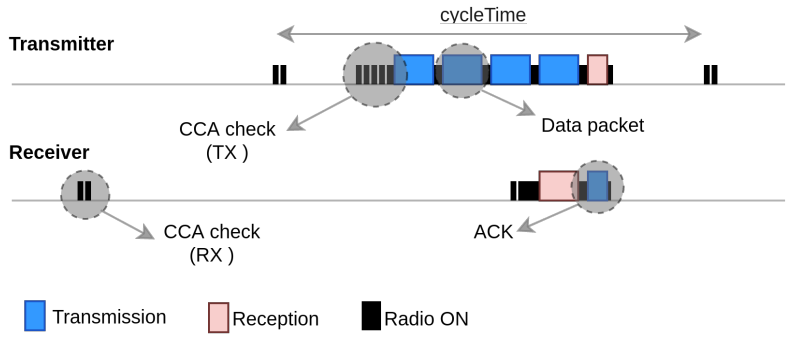
These two widely used MAC protocols serve as a showcase of typical MAC development challenges. The workings of these MAC protocols are visualized in Figure 5.1.

3.3 Challenge 1: The lack of feature-rich API's that support portability

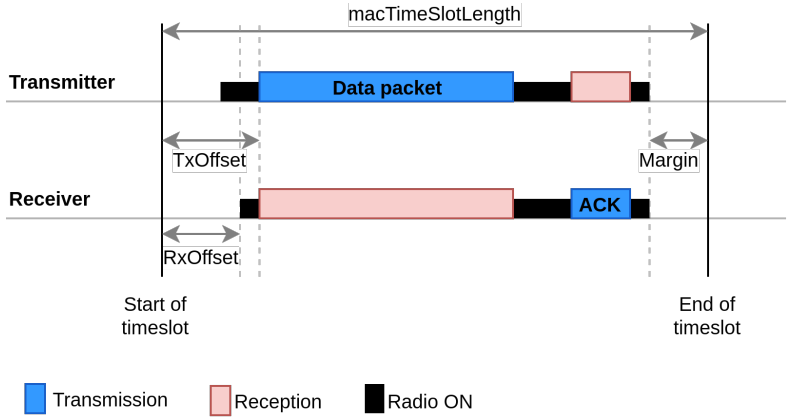
Context. When setting up a wireless sensor network, the MAC protocol is one of the most important decisions as it impacts how efficiently the medium is accessed. If the desired MAC protocol is not available on a platform of choice it needs to be implemented or ported, which is a time consuming process due to the code complexity and hardware dependencies.

Problem. In most operating systems and/or MAC development architectures (e.g. Contiki and openWSN) MAC implementations use a HAL or radio driver which provides an interface towards the instruction set of the radio chip. A number of limitations are caused due to the inherent design of common hardware abstraction layers.

- A first common pitfall is the creation of a generic instruction set which limits the possibilities of the designer. Most radio chips have advanced features which should be used to their fullest extent in order to create a MAC protocol which is as efficient as possible. However, these are often not included in



(a) Conceptual figure of ContikiMAC. The cycletime is the time between consecutive CCA check cycles. CCA Receive (RX) is a CCA check performed to listen for incoming frames. CCA Transmit (TX) is a CCA check performed to listen if medium is available to transmit a data packet.



(b) Conceptual figure of IEEE802.15.4e TSCH. The most important metric for this chapter is macTimeSlotLength, which is the time duration necessary to safely transmit a data packet and to receive the consecutive Acknowledgment (ACK).

Figure 3.1: Illustration of the protocol logic from two widely used MAC protocols: (a) ContikiMAC and (b) Institute of Electrical and Electronics Engineers (IEEE) 802.15.4e TSCH.

the hardware abstraction layer due to many interdependencies between the radio system states. For example, putting a device to sleep mode requires a wide range of checks on multiple subsystems of the hardware platform before you can safely decide which LPM mode is possible without loss of data or control of your device. To avoid this complexity, e.g. in Contiki and openWSN the on/off function call is instead implemented as an idle function on some platforms which will not save energy and is different in terms of timing. A developer who only uses the abstraction layer and is not aware of

the underlying hardware implementation, will not realize why his protocol is suboptimal.

- Alternatively, some MAC architectures provide a more elaborate interface which is more expressive and intelligent in terms of behavior dependencies. For example RIOT OS and TAISC allow full access to the radio state. Unfortunately, the extended abstraction layer has a drawback. Since not all radio chips support these advanced capabilities they might not be able to run the protocol code, again hindering portability of the MAC implementation.

- TinyOS builds on the concept of abstraction layers. At the lowest layer, the Hardware Presentation Layer (HPL) provides chip specific hardware features (memory access, timers, ADC/DAC, etc.) without providing explicit interfaces for MAC design. Specifically for MAC design, two abstraction levels are offered: (i) the HAL provides platform dependent radio level instructions (e.g. chip specific), and (ii) the Hardware Interface Layer (HIL) provides generic, platform independent MAC level instructions.

As such, current MAC protocol architectures either support multiple platforms by offering only a generic radio/platform-independent instruction set, or they offer full radio control but allow no portability options.

Solution. In contrast to existing hierarchies, the major differences between our approaches are as follows. (i) TinyOS and RIOT MAC interfaces are inherently time-unaware, leaving the burden of calculating when to execute a command to the developer, thereby reducing protocol efficiency and hindering portability. (ii) In contrast to the two API layers of TinyOS which are either chip specific or very generic, we introduce three time-annotated layers. As such, the developer can choose to write a MAC protocol which is not portable (similar to the TinyOS HAL API, but more efficient due to time annotations), portable between devices of the same technology (not supported by TinyOS), or fully portable between all devices (similar to the TinyOS HIL API, but more efficient due to time annotations). (iii) Finally, we have extended the tier-architecture with fallback compilation functions. These functions allow MAC designers to implement a protocol for optimal performance while still allowing compilation towards higher platform independent layers, which is to the best of our knowledge a novel addition.

More specifically, three levels of compatibility were identified.

1. **Radio functionality level.** Using this abstraction layer, the MAC protocol implementation will only be compatible with platforms using the exact same radio chip. This allows to use all radio features.

2. **Technology level.** Using this abstraction layer, the MAC protocol implementation will be compatible with platforms using the same technology type

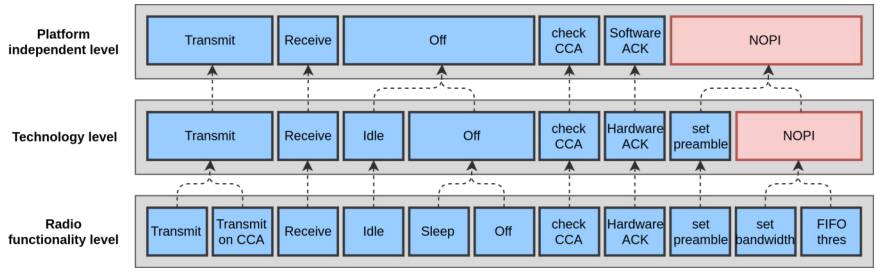


Figure 3.2: An example hierarchical MAC API is defined allowing explicit portability trade-offs. The radio functionality level provides all features of the radio and only offers compatibility between platforms using the same radio chip. The technology level offers all features of a specific network technology, and are thus compatible with radios supporting the same network type. The last level offers a fully platform independent instruction set. If a function does not exist on a hardware platform, a fall back function or dummy implementation is automatically selected in the compilation phase.

(e.g. BLE, Lora, IEEE 802.15.4, etc.). For example the set_bandwidth function is available for all LoRa radio chips (but not for e.g. IEEE 802.15.4 radios).

3. **Full platform independence.** Using the abstraction layer, the MAC protocol implementation will be compatible with most wireless radio chips. This functionality layer offers basic radio functions (on/off, send, etc.), which are available on most wireless hardware platforms.

In Figure 3.2 the three levels are visually represented. A developer can choose which interfaces to use, and thus by extension with which platforms he wants to be compatible with. The higher hierarchies contain more generic, often less efficient, implementations of the radio chip specific function calls.

For each function, fall-back functions are explicitly provided. This way, it is possible to implement MAC protocols using a rich-feature HAL, but to still be able to compile MAC protocols to other, less-feature rich platforms[2]. For example, in case fine-grained low power modes are unavailable, the MAC compiler replaces these function calls with the more general "off"-instruction. Some unusable configuration functions (e.g. set_preamble for 2.4GHz networks) can be replaced by a placeholder instruction ("NOPI") while keeping the MAC protocol logic intact[3]. Since all fallback functions are time-annotated, the timing impact of using these functions can automatically be compensated by the compiler (see Section 3.4).

Evaluation. The hierarchical MAC API functionality has been implemented in the MAC framework and its benefits have been evaluated using an example sce-

[2]These fall-back mechanisms provide timing information which can also be used to allow interoperability between different radio chips (see Section 3.5)

[3]In case no fallback is available on the requested compatibility level a compiler warning is shown.

nario. We consider a simple MAC protocol that has been designed for a radio that supports the creation of hardware ACK's (lowest API level). When the same MAC protocol is later compiled for a radio without hardware acknowledgement support, the compiler provides a warning and automatically replaces this functionality with a fallback function (in this case: a software based acknowledgement generation). To validate the correctness, the impact of using this specific fall-back function was experimentally quantified. For the experiment, a Zolertia Remote device aims to achieve maximum unidirectional throughput to a second device using the ALOHA MAC protocol. Using hardware acknowledgements, the maximum throughput was measured to be 153.8kilobit per second (kbps). Next, we artificially disabled hardware support for acknowledgement generation. Rather than breaking the MAC protocol, the compiler automatically used the software acknowledgement fall-back option. The resulting throughput was 142.5kbps (a decrease of only 7%), showing that fallback functions are a viable approach when porting MAC protocols to hardware with different capabilities, allowing the MAC protocol to run on less-feature rich platforms, albeit at slightly decreased performance.

3.4 Challenge 2: Hardware dependent instruction execution times

Context. MAC protocol implementations mainly consist of time critical code, e.g. when to start listening to the wireless medium or when to start transmitting. These timers impact the efficiency of the protocol (e.g. reducing the time spent awake or impact the number of transmissions per unit of time) and thus by extension the overall energy consumption of the system.

Problem. Unfortunately, due to hardware differences, there exist large timing differences between platforms. To illustrate the impact of these timing differences between platforms, Figure 3.3 shows the time required to turn a radio on, switch to a specific radio channel, load and transmit a short packet of 32 bytes and turn the radio off again for different IEEE 802.15.4-compliant radios[4]. This sequence represents for example the minimum duration of a TDMA slot. As shown in Figure 3.3 the performance differs dramatically between devices even for a basic sequence of radio instructions. These differences are most likely caused by hardware differences, but also differences in the implementation of the underlying API. Failure to take into account these specific platform timings into the MAC implementation results in suboptimal timings, which can cause the protocol logic to break. Because the MAC implementation is typically created for one specific radio chip, instances running on other platforms often contain hard to identify bugs [15].

[4]Respectively the CC2420 radio on the TMote Sky and the Zolertia Z1, the CC2520 radio on the RM090 and the CC2538 radio on the Zolertia Re-mote

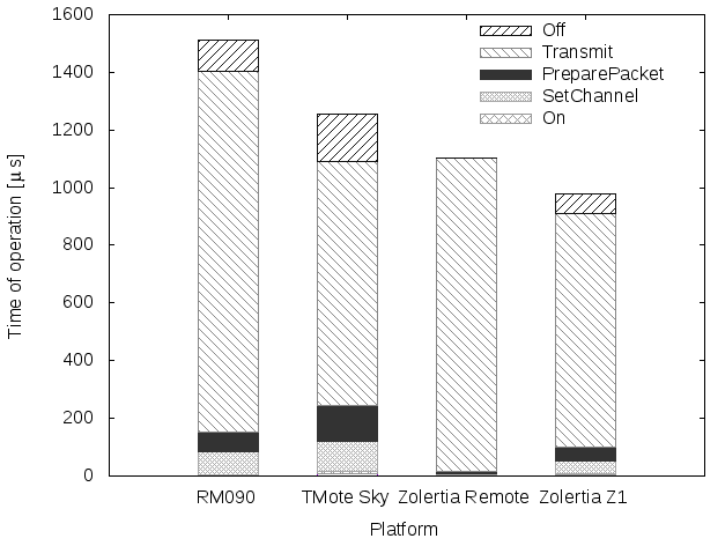
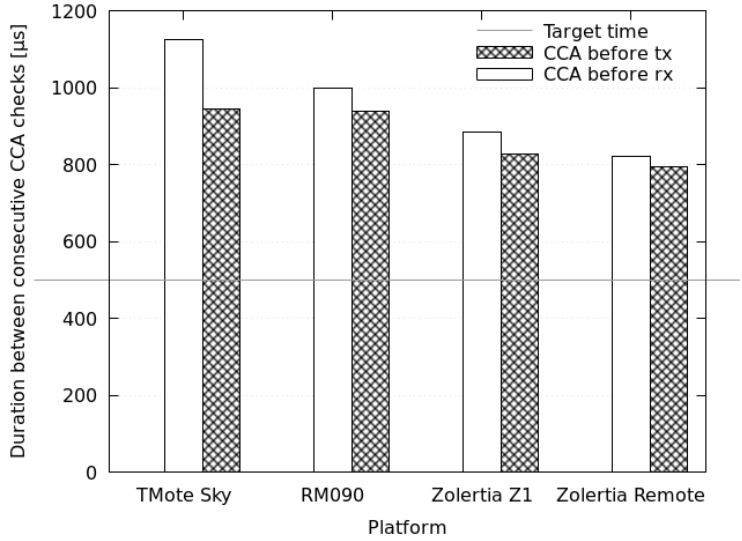


Figure 3.3: Comparison of the duration of basic radio instructions. The instruction timings vary significantly between different platforms.

Ideally, each separate radio function call should happen as soon as possible. However, slower platforms might not be able to perform all needed functionality within the defined limited amount of time. This poses a dilemma for the MAC designer: should he include additional safeguard time between each function call, thereby making the protocol less efficient but more portable?

- Consider for example the ContikiMAC time between two consecutive CCA checks. According to the official documentation, this time is assumed to be 500 µs. Figure 3.4 represents the average CCA timings directly measured on several hardware platforms. None of them come even close to reaching the target time, making it likely that a short packet cannot be detected by the CCA checks. To make matters worse there is a time difference between CCA checks in transmit- and receive-mode, due to the logic in between the checks being different.

- Conversely, developers can include margins in their timings, to counter clock drift or to ensure multi-platform compatibility. An example can be found in the CSMA/CA implementation in Contiki, which waits to receive an acknowledgment. Even if the acknowledgment has already been received the radio remains in the receive state until the AFTER_ACK_DETECTED-_WAIT_TIME has passed. Not only is this wait period too long, the MAC protocol also does not react to events happening at the physical layer. A



$0.025\,\mu s$.

Figure 3.4: Time between consecutive ContikiMAC CCA checks for different platform, demonstrating that the timing varies significantly depending on the platform. The figure also illustrates that the default ContikiMAC duration (horizontal line) is too strict for the considered platforms. Fifty measurements were performed per plaform/CCA type combination. The variance of the measurements was negligible, since for the slowest devices it was measured to be only

radio should go to idle or LPM when the acknowledgment is received. Although this choice might ensure the protocol runs on multiple radio platforms, it results in performance degradations.

Solution. The above problems can be alleviated by having more knowledge regarding the timing of different instructions for different hardware platforms. These timings are hard to calculate theoretically since they depend on many external factors (clock speed, transition type, etc.). Instead, we propose to benchmark the execution time of different instruction timings (e.g. CCA timings, duty cycle, packet transmission timings, etc.) automatically before compile time, so that these values can be included in the MAC code.

Injecting real radio instruction execution durations measured on the device ensures the protocol logic will not break due to wrongly chosen timings, and removes the need to include large margins to reach multi-platform compatibility. From a practical point of view, measuring the instruction timings (rather than code block timings) is more useful since the data can then be reused for future MAC implementations. Measuring the instruction timings could be done in two ways. (i)

Either the instructions are executed, and the timings are measured by a second device. In this case, the authors advice to use designated devices (e.g. a logic analyzer) which are more accurate and precise. (ii) Alternatively, a pre-compiler can automatically execute each used individual instructions sufficient times to derive statistically relevant information about its execution times. These instruction timings can be stored locally, or could be shared through an automated repository with external database. Using these values, the duration of logical blocks can be calculated out of the individual instruction timings. The burden of selecting optimal timing information is now shifted from the MAC developer to the pre-compiler, allowing the MAC developer to focus on implementing the protocol logic without manually choosing the platform-specific timings.

To estimate the duration of each instruction, the benchmarking framework needs to be able to predict the duration of instructions with variable execution times, caused by several factors.

- The first factor of variability is caused by configurable function parameters, for example the number of bytes to be transmitted. This can be solved by measuring how long it takes to execute a single unit (e.g. the transmission duration of a byte is $32\,\mu s$), which can be used to calculate the duration of larger transmissions.

- The second factor is the occurrence of interrupts during function execution. If an interrupt occurs, the instruction timing is dependent on the duration of the Interrupt Service Routine (ISR). This can partly be solved by running the MAC protocol in the highest possible interrupt context, minimizing the possibility that the execution of the MAC protocol can be interrupted or delayed.

- Lastly, the instruction execution timing can be influenced by clock drift. To handle this, a number of measurements should be performed to capture the distribution of instruction durations (e.g. min, max, average duration). The evaluation in this chapter assumes a worst case scenario, to make sure that the instruction is always completed in the defined amount of time. Alternatively, the performance could be increased by assuming an instruction duration which is valid in e.g. 95% of the time (95 percentile).

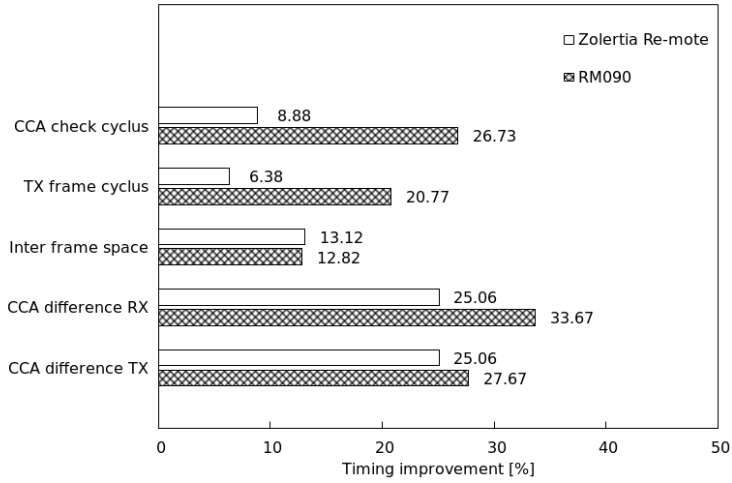
Evaluation. To evaluate these concepts, we extended TAISC with an automated benchmarking framework for time duration. For every evaluated platform an eXtensible Markup Language (XML) is created with the instruction times for all instructions of all three hierarchical HAL layers of Section 3.3. This multi-platform XML document is then used to support cross-platform compilation of MAC code. This information is used for scheduling the next instruction: at what point in the future should a next instruction be executed to make sure that execution

does not start before the current one has finished, e.g. the off-instruction should not be executed as a TX-instruction is still active. The next instruction should be scheduled as close as possible to the end of the current instruction[5]. This allows to approach the limits of the system in a controlled manner: the result is a MAC implementation that can be compiled to multiple radio platforms with limited, or even non-existing margins. Next the performance gains will be evaluated for the MAC protocols: ContikiMAC and IEEE 805.15.4g TSCH.

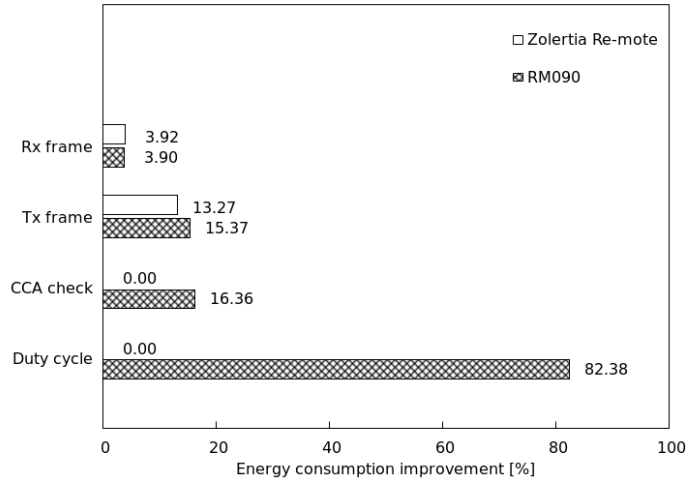
Figure 3.5a shows the timing improvements of the automatically calculated timings, compared to the default Contiki implementation, for basic ContikiMAC operations. For every operation, the automated timing calculation outperforms the default implementation. The performance benefits from the shorter acquired timings. the CCA times are a lot closer towards the physical limit of the device, up to 33.67% better compared to the default implementation. Also the Inter Frame Space (IFS) is 13.32% better compared to the default implementation. Thus, by including automated time annotations, the performance significantly improves and the protocol logic can efficiently be reused on multiple radio platforms. The better timings achieved through time-annotation also have an impact on radio energy consumption as the radio is more likely to be in the correct state at the right moment, e.g. the radio is less in the energy-consuming 'on'-state. Figure 3.5b shows the improvement in energy consumption of several basic ContikiMAC operations. The RM090 benefits with a major improvement in energy consumption as the radio goes into lower power mode if no operations need to be executed, while in the default implementation it only goes to idle-mode. This difference becomes apparent in the duty cycle and CCA check operations in Figure 3.5b as the energy consumption for the RM090 is considerably lower. In Figure 3.5b it can be seen that the same improvement does not happen on the Zolertia Re-mote, since the radio can only put into a lower power mode if the Micro Controller Unit (MCU) is also put in low power mode making it impossible to control from the MAC layer.

In contrast to ContikiMAC, TSCH is a synchronized TSCH MAC protocol. The default slot sizes described in the IEEE 802.15.4 and IEEE 802.15.4e standards include large margins for two main reasons: (i) to compensate clock drift on the devices, and (ii) to use multiple device types within the same network. Current TSCH implementations are compatible because the slot duration is chosen so slower radio platforms can adhere to these timings, regardless of the actual presence of such platforms in the network. The default fixed slot duration (TsSlotDuration) of 15ms is not optimal for the performance: a reduced slot duration would improve the network performance (energy consumption and throughput). Hence, the current approach towards portability does not result in optimal performance. The XML containing instructions was used to automatically calculate the mini-

[5]It should be noted that the provided MAC API instructions have a very predictable performance (variance is less than 1 µs).



(a) Several timings of ContikiMAC were measured on both the time annotated instruction version and the default version. This graph represents the improvement (in %) which the time annotated instructions achieve compared to the default ContikiMAC.



(b) For several operations, the energy consumption was measured on both the the time annotated instruction version and the default version. This graph represents the improvement (in %) which the time annotated instructions offer compared to the default ContikiMAC.

Figure 3.5: Performance evaluation (timings and energy consumption) from including instruction timing information in the ContikiMAC protocol: the MAC protocol is automatically optimized for the capabilities of each individual radio platform.

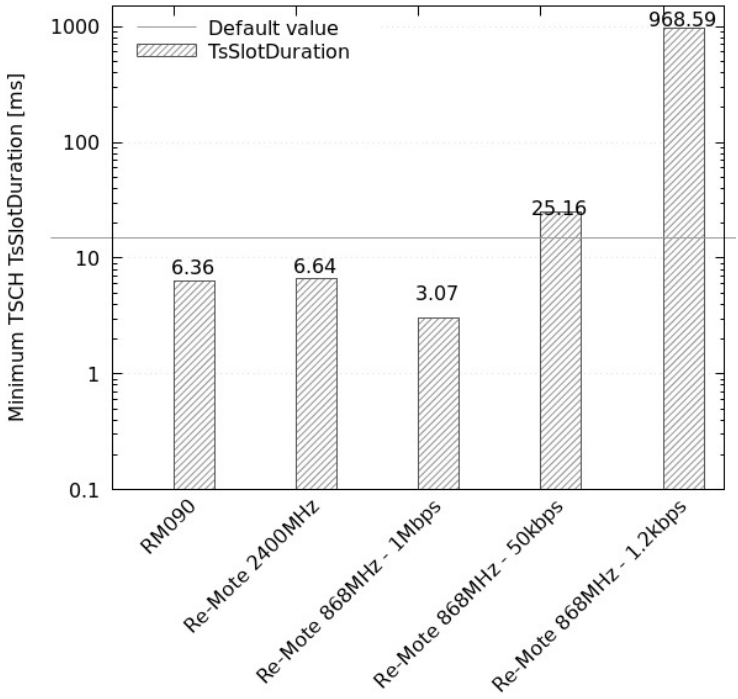


Figure 3.6: Automatically calculated minimum TSCH slot duration per platform. The default TSCH slot duration (15ms) is indicated by a horizontal line.

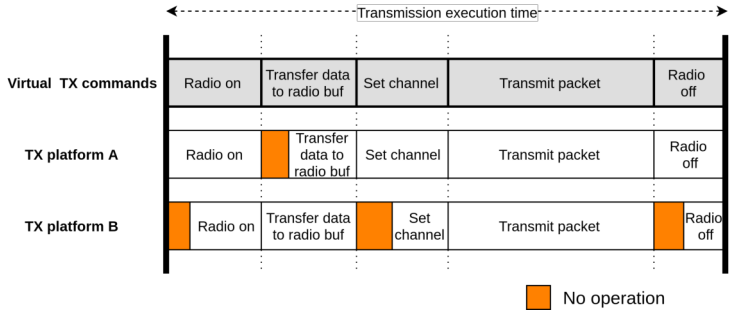
mal TSCH slot duration for a number of platforms (not including clock drift), for which the result can be observed in Figure 3.6. The timings are either significantly shorter, or might require more then 15ms depending on the selected modulation. This demonstrates the limitations of imposing manual slot durations, which can be overcome by automatically calculating the actually needed instruction timings.

3.5 Challenge 3: Multi-platform networks exhibit unpredictable behavior

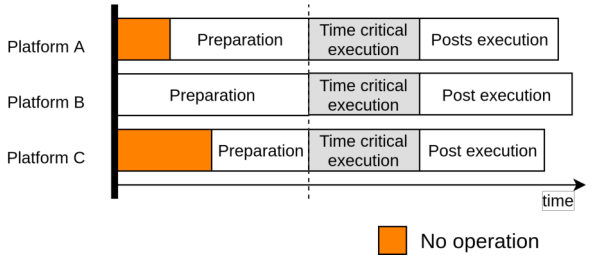
Context. Previous sections focused on problems related to running the same MAC protocol code on different devices. This section describes problems that arise from having multiple devices of a different hardware type in the same network. Even when these devices run the same MAC protocol code, the performance might still differ due to time differences introduced by a different physical layer implementation and a different Central Processing Unit (CPU) clock speed.

Problem. To demonstrate the performance impact of different platforms running the same code, an experiment was conducted where different hardware platforms send data to a central server. All nodes form a star topology, whereby the leaves were 5 RM090 and 5 Zolertia Remote devices, all using the default implementation of the ContikiMAC protocol (with a cycle time of 125ms) and the same configuration (channel, transmission power, clear channel assessment threshold, etc). All nodes were configured to transmit a single packet per second to a central node. The experiment was repeated twice, each time with a different type of device as central node. Figure 3.8a shows the percentage of successfully sent packets over the total per platform. Although the same software code was used, the percentage of successful packet transmissions of the Zolertia Remote is significantly higher than the ones of the RM090, since the Zolertia Remote can access the medium more quickly due to faster instruction timings (see Figure 3.3). As such, there is a need to ensure fairness in the presence of different hardware platforms, since this is not inherently offered by the use of the same MAC protocol.

Solution. To counter cross-platform incompatibility the execution duration of the different instructions should be equalized across the different platforms.



(a) An additional margin is added after each instruction to ensure all instructions have the same duration across all platforms.



(b) A single margin is added before the full instruction sequence to ensure that the execution of the critical time components (for example TX or CCA) happens simultaneously across all platforms.

Figure 3.7: Approaches to equalize the MAC performance across multiple device types within the same network, thereby ensuring fairness and cross-platform compatibility.

- One solution is to add wait periods to ensure that instructions require the exact same amount of time on all platforms in the network (Figure 3.7a). For each instruction, the corresponding duration of the worst platform in the network is used as the target duration. Figure 3.7a shows how this approach ensures that all platforms now finish the packet transmission in the same amount of time. By using the time-annotated commands to calculate the additional wait periods, the performance of all platforms is equalized, at the cost of extending the duration of some states on faster platforms[6].

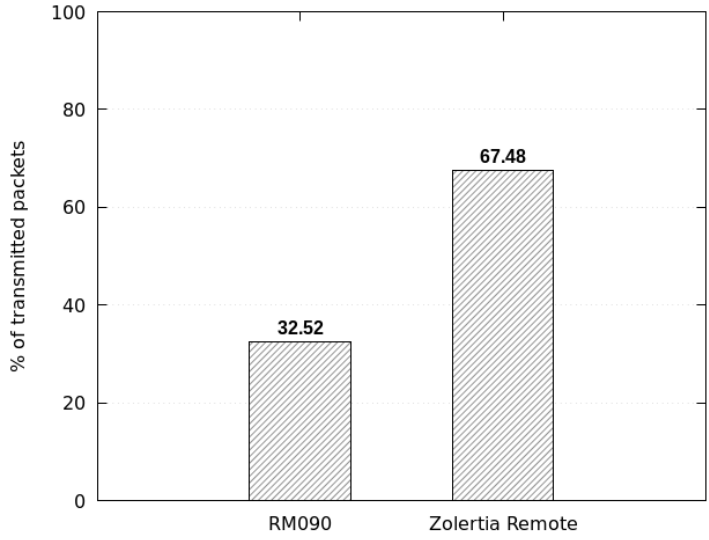
- Alternatively, it is possible to use the timing information to reschedule essential instructions (such as TX instructions) so that they start at the same time across different platforms. Most MAC logic blocks consist of three phases: (i) a preparation phase, (ii) a critical execution, and (iii) parsing of the execution results. By adding wait periods or margins at the beginning of the MAC block, the instructions can be scheduled so that the time critical execution occurs simultaneously (Figure 3.7b). This approach ensures fairness and does not negatively impact energy efficiency of the individual platforms, but it does require knowledge about which parts of the execution should be aligned.

Evaluation. The cross-platform compatibility solutions have been evaluated by extending the TAISC framework with a ”sync”-command that can be used for multi-platform designs. Every MAC protocol instruction sequence can be annotated with one such sync command indicating the critical execution part. Using these extensions time-critical parts of the MAC protocol (e.g. a frame transmission) are scheduled immediately after the sync, and the preparation for the time-critical execution (e.g. copying a packet into the radio buffer, putting the radio in the correct state) are scheduled before. It becomes possible to align critical parts of instruction blocks, e.g. put the radio in receive mode just before another node transmits a packet, without the need to calculate the timings/margins by the MAC implementer. The impact of these changes is shown in Figure 3.8b. Compared to the same experiment in default Contiki from Figure 3.8a, the platforms now share the spectrum fairly due to the alignment of the time critical execution parts.

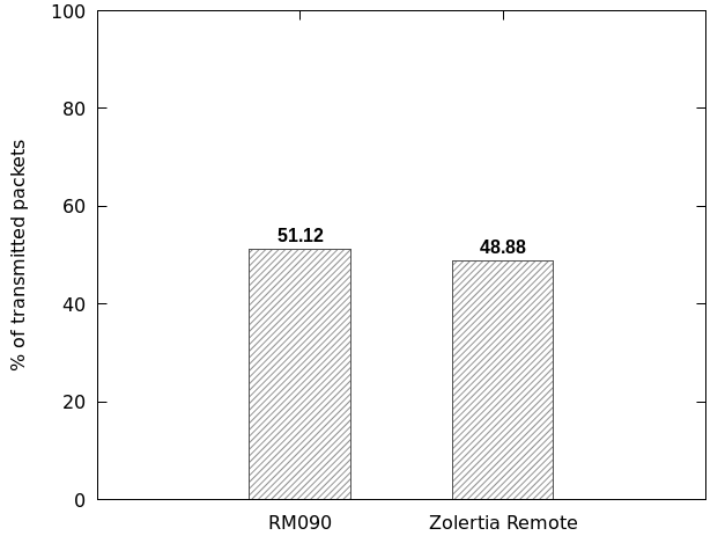
3.6 Conclusions

Scientific literature regarding MAC protocols currently focuses on the design and evaluation of new MAC protocol prototypes, most often without considering reuse of code towards future platforms. As a consequence, the availability of MAC

[6]For example in 3.7a, platform B goes to the radio on state faster compared to platform A. By doing so, the radio is already consuming energy when it is not yet strictly necessary.



(a) Percentage of transmitted packets over the total for two platforms running the same MAC code in a single network. The performance variation between different platforms is significant.



(b) Percentage of transmitted packets over the total for two platforms running the same MAC code in a single network. By using time annotations (Section 3.4) the overall performance increases. By also aligning the critical execution parts (Section 3.5), the fairness increases.

Figure 3.8: Impact of different platforms in a single network running the same MAC code on different hardware.

protocols designed for a specific device is often very limited. Even though multi-platform MAC protocol implementations exist, performance degradation and offered functionality differing from defined/designed behavior caused by hardware differences and implementation mistakes are common. In addition, multi-platform network deployments suffer from unfairness and unpredictable performance since, as it was proven, the same MAC protocol logic on multiple devices behaves differently in terms of timing due to hardware differences. Consequently, different devices running the same MAC protocol logic can become incompatible with one another.

Table 3.1: Summary of how the three proposed solutions in this chapter contribute to the portability, compatibility and reuse of multi-platform MAC protocols.

Solution	**Portability**	**Compatibility**	**Reuse**
1. Hierarchical framework	x		x
2. Automatic benchmarking	x		
3. Instruction aligning		x	

This chapter described in detail the challenges related to multi-platform MAC development, and experimentally quantified how the above issues impact the MAC protocol performance when deploying and executing MAC protocols on multiple radio chips. (i) Firstly, it was shown that current interfaces are either too radio chip specific, thereby hindering portability of MAC protocols, or too general, thereby forcing developers to generate inefficient code. By providing a hierarchical framework of interfaces, our proposed methodology allows efficient code implementation while allowing fall-backs to alternative implementations for portability. (ii) Secondly, it was shown that MAC timings depend strongly on the underlying hardware. By automatically benchmarking the duration of MAC instructions, our approach is able to optimize the MAC towards the specific radio chip timings, thereby exhibiting better performance in terms of radio energy consumption (up to 82% improvement), timings (up to 33% improvement) and throughput compared to the existing multi-platform implementations. (iii) Thirdly, we proved that the same MAC protocol implementation acts differently per platform, resulting in unfair/unpredictable behavior when multiple platforms are combined in the same network. By adding margins to equalize the duration of all radio instructions, or by aligning the critical instructions across multiple platforms, these unpredictable effects are removed and fairness is restored. Table 3.1 summarizes which purpose each the proposed solutions serves in the overall goal of achieving portable, multi-platform MAC protocols.

It is worth noting that all three solutions can be applied individually, or the can be combined with each other towards an overall methodology for the design of portable MAC protocols. The different steps of the methodology are visualized in Figure 3.9. By following the described steps it becomes possible to achieve a

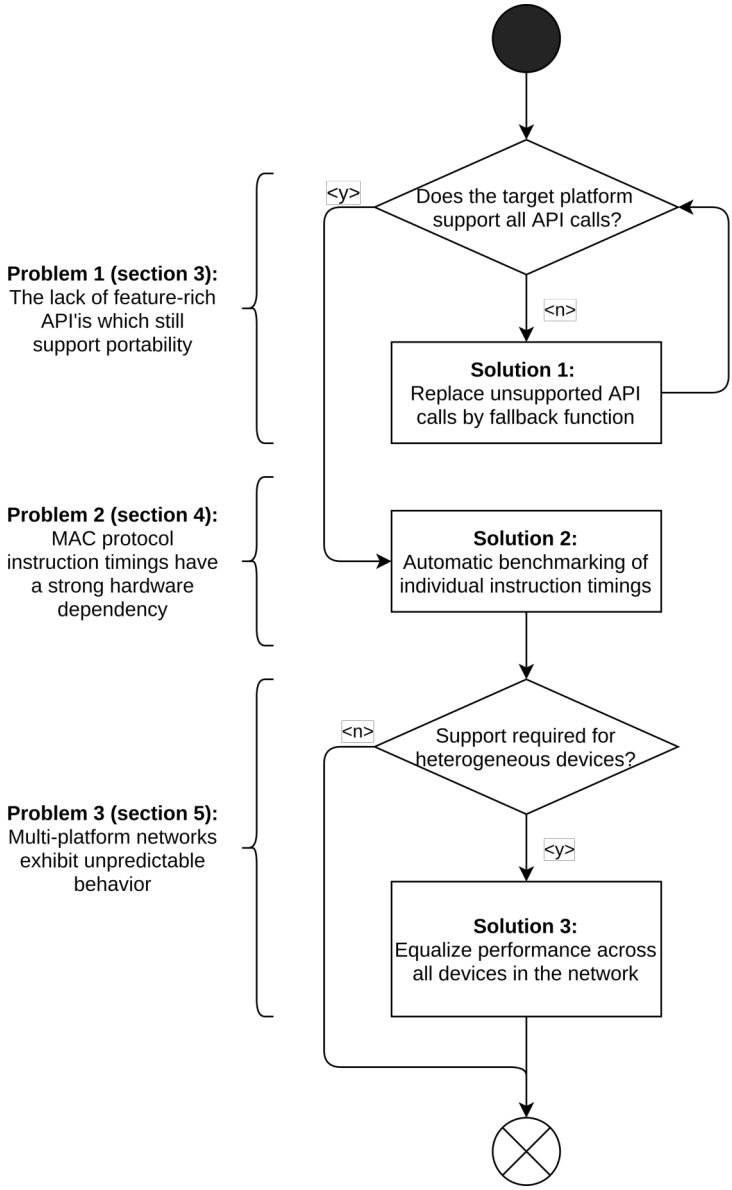


Figure 3.9: Flow diagram summarizing the three solutions to multi-platform issues in MAC protocols.

MAC design which (i) is more developer friendly, by moving the burden of time annotation towards the pre-compiler, (ii) is more efficient compared to traditional approaches, significantly improving MAC performance and energy efficiency and (iii) allows efficient reuse and combination of MAC protocols over a wide range of IoT platforms without additional development efforts.

Acknowledgment

This work was partially supported by project SAMURAI: Software Architecture and Modules for Unified RAdIo control, and European Commission Horizon 2020 Programme under grant agreement no. 645274 (WiSHFUL).

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann. *Industry 4.0*. Business & Information Systems Engineering, 6(4):239–242, 2014. doi:10.1007/s12599-014-0334-4.

[2] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi. *The evolution of MAC protocols in wireless sensor networks: A survey*. IEEE communications surveys & tutorials, 15(1):101–120, 2013. doi:10.1109/SURV.2012.040412.00105.

[3] J. R. Cordeiro, D. F. Macedo, and L. F. Vieira. *FS-MAC: A flexible MAC platform for wireless networks*. In Wireless Communications and Networking Conference (WCNC), 2018 IEEE, pages 1–6. IEEE, 2018. doi:10.1109/WCNC.2018.8376981.

[4] P. Thubert. *An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4*. Internet-Draft draft-ietf-6tisch-architecture-12, Internet Engineering Task Force, August 2017. Work in Progress. Available from: https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-architecture-12.

[5] J. A. Gutierrez, E. H. Callaway, and R. Barrett. *IEEE 802.15.4 Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensor Networks*. IEEE Standards Office, New York, NY, USA, 2003. doi:10.1109/WCNC.2003.1200605.

[6] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. C. Schmidt. *RIOT OS: Towards an OS for the Internet of Things*. In Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on, pages 79–80. IEEE, 2013. doi:10.1109/INFCOMW.2013.6970748.

[7] A. Dunkels, B. Gronvall, and T. Voigt. *Contiki-a lightweight and flexible operating system for tiny networked sensors*. In Local Computer Networks, 2004. 29th Annual IEEE International Conference on, pages 455–462. IEEE, 2004. doi:10.1109/LCN.2004.38.

[8] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister. *OpenWSN: a standards-based low-power wireless development environment*. Transactions on Emerging Telecommunications Technologies, 23(5):480–493, 2012. doi:10.1002/ett.2558.

[9] J. Bauwens, B. Jooris, E. De Poorter, P. Ruckebusch, and I. Moerman. *towards a MAC protocol app store*. In EWSN 2016, the International Conference on Embedded Wireless Systems and Networks, pages 1–2, 2016. doi:1854/LU-7172061.

[10] J. G. Ko, N. Tsiftes, A. Dunkels, and A. Terzis. *Pragmatic low-power interoperability: ContikiMAC vs TinyOS LPL.* In Sensor, mesh and ad hoc communications and networks (SECON), 2012 9th annual IEEE communications society conference on, pages 94–96. IEEE, 2012.

[11] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, J.-P. Vasseur, M. Durvy, A. Terzis, A. Dunkels, and D. Culler. *Industry: beyond interoperability: pushing the performance of sensor network IP stacks*. In Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, pages 1–11. ACM, 2011.

[12] J. Vanhie-Van Gerwen, E. De Poorter, B. Latré, I. Moerman, and P. Demeester. *Real-life performance of protocol combinations for wireless sensor networks*. In 2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, pages 189–196. IEEE, 2010.

[13] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, A. Terzis, A. Dunkels, and D. Culler. *Contikirpl and tinyrpl: Happy together*. In Workshop on Extending the Internet to Low Power and Lossy Networks (IP+ SN), 2011.

[14] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester. *The w-iLab. t Testbed.* In TridentCom, volume 46, pages 145–154, 2010. doi:10.1007/978-3-642-17851-1_11.

[15] M. Uwase, M. Bezunartea, J. Tiberghien, J. Dricot, and K. Steenhaut. *Experimental Comparison of Radio Duty Cycling Protocols for Wireless Sensor Networks*. IEEE Sensors Journal, 2017. doi:10.1109/JSEN.2017.2738700.

[16] M. Brzozowski and P. Langendoerfer. *Is cross-platform protocol stack suitable for sensor networks? Empirical evaluation*. In Wireless and Mobile Networking Conference (WMNC), 2013 6th Joint IFIP, pages 1–8. IEEE, 2013. doi:10.1109/WMNC.2013.6548983.

[17] M. Brzozowski, H. Salomon, K. Piotrowski, and P. Langendoerfer. *Cross-platform protocol development for sensor networks: lessons learned*. In Proceedings of the 2nd Workshop on Software Engineering for Sensor Network Applications, pages 7–12. ACM, 2011. doi:10.1145/1988051.1988054.

[18] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. *MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms*. Mobile Networks and Applications, 10(4):563–579, 2005. doi:10.1007/s11036-005-1567-8.

[19] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli. *Wireless MAC processors: Programming MAC protocols on commodity*

hardware. In INFOCOM, 2012 Proceedings IEEE, pages 1269–1277. IEEE, 2012.

[20] B. Jooris, J. Bauwens, P. Ruckebusch, P. De Valck, C. Van Praet, I. Moerman, and E. De Poorter. *TAISC: A cross-platform MAC protocol compiler and execution engine*. Computer Networks, 107:315–326, 2016. doi:10.1016/j.comnet.2016.03.027.

[21] A. Dunkels. *The contikimac radio duty cycling protocol*. Swedish Institute of Computer Science, 2011.

Part II

Enabling multi-radio support in wireless MAC protocols

4

Multi-radio support in energy constrained Internet of Things (IoT) networks: Overview of current and future approaches

Nowadays IoT platforms make use of multiple radios in their hardware design. Through the combination of these different radio technologies (and their accompanying network standard) the advantages of each technology can be combined in an attempt to improve the network-level Quality of Service (QoS). However, current IoT operating systems do not have the capability to optimally use multiple radio technologies on a single platform. This chapter therefore proposes a number of techniques to use multiple Medium Access Control (MAC) protocols on a constrained device.

⋆⋆⋆

J. Bauwens, D. Van Leemput, S. Giannoulis, I. Moerman, and E. De Poorter.

Submitted in Elsevier Pervasive and Mobile Computing

Abstract
Throughout the history of the Internet of Things (IoT), a large number of strides

have been made to optimize the behavior of wireless devices in increasingly diverse application domains. A plethora of wireless standards exist which enforce the use of a set of protocols per network layer, each of which have a number of (dis)advantages compared to the other technologies making them suitable for a (sub)domain within the IoT ecosystem. By combining multiple wireless standards and their dedicated radio on a single multi-modal device, the advantages of each wireless standard or protocol can be combined, thereby improving the overall wireless behavior. For example, depending on the application requirements or current environmental conditions, a different MAC protocol can be preferred for packet transmission. However, since most IoT devices are constrained in terms of processing power, running multiple MAC protocols on a single device is not straight forward. The execution of one protocol can interfere with the execution (and associated timings) of other protocols, thereby influencing the wireless performance. This chapter gives an overview of possible MAC techniques which can be used to make better use of the available radios, by (i) switching between pre-installed protocols, (ii) using a single MAC protocol controlling multiple interfaces, (iii) executing the protocols in parallel by dividing execution time between the protocols, and (iv) utilizing a dedicated Micro Controller Unit (MCU) per available MAC protocol.

4.1 Introduction

The Internet of Things (IoT) is a terminology which refers to the concept of wirelessly interconnecting a variety of ‘things‘ to form a network. By giving these devices the possibility to sense the environment and perform (inter-)actions, they are transformed into intelligent objects. Through their ability to ‘talk‘ to other things, they can be applied to realize several advanced use cases. For example, IoT solutions are already being deployed to serve a wide range of applications, ranging from day-to-day use cases in health care, surveillance, agriculture, or personal fitness, all the way to home and industry automation. This widespread adoption is leading to a rapid increase in (i) the number of wireless devices per person and (ii) the number of wireless devices per square meter. An important aspect of IoT devices are their inherent limitations: they are typically small, cheap and have a limited battery capacity. Additionally most of these devices are constrained in terms of processing power, memory and data storage to keep the unit price low.
To achieve wireless connectivity, IoT devices typically make use of a network stack containing several layers: (i) the application layer, (ii) the network layer, (iii) the MAC layer, and (iv) the physical layer. A plethora of wireless standards exist which enforce the use of a set of protocols per layer, each of which have a number of (dis)advantages compared to the other technologies making them suitable for a (sub)domain within the IoT ecosystem (see Figure 4.1 and Table

Table 4.1: An overview of a number of commonly used radio technologies, showing their inherent differences.

Wireless standard	**Radio frequency [MHz]**	**Modulation**	**Throughput [kbps]**	**Typical distance [m]**	**MAC protocols**	**Usage**
IEEE802.15.4	868, 915 or 2400	BPSK or O-QPSK	20, 40 or 250	10-100	CSMA/CA or TSCH	communication for low-power personal area networks
Bluetooth Low Energy	2400	GFSK	BLE4.0: 1000 BLE5.0: 2000	BLE4.0: 100 BLE5.0: 400	TDMA with AFH	Low power variant of Bluetooth
Z-Wave	868	FSK/GFSK	40	30	CSMA/CA	Residential automation
IEEE 802.11ah	868	BPSK, QPSK, 16-QAM, 64-QAM or 256-QAM	150 up to 346.670	100 m up to 1 km	IEEE802.11 with RAW, TIM and TWT	Low power, long range variant of Wi-Fi
IEEE 802.15.4g	868	MR-FSK, MR-OFDM, MR-O-QPSK	50 up to 800	2000-3000	CSMA/CA or TSCH	Smart utility network (SUN) communications
SIGFOX	868	BPSK	0.1 to 1.0	50.000	Send data over 3 frequencies at different times	Low Power Wide Area Networks
LoRa	433, 868, 915	CSS	0.3 to 50	15.000	LoRAWAN	Low Power Wide Area Networks

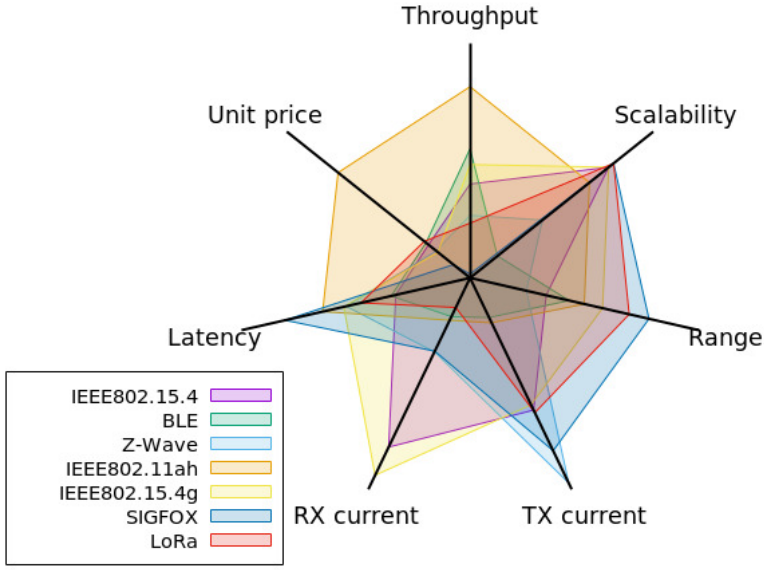


Figure 4.1: A number of widely used wireless standards are compared by using several key characteristics. It is shown that there is no single standard which predominantly outscores the others, making the choice of standard a trade-off highly depending on the use case requirements.

4.1). At first, these wireless standards focused on providing reliable and energy efficient communication over a limited distance, typically denoted as a Wireless Sensor Network (WSN) [1]. Two typical variants are the IEEE 802.15.1 (used by Bluetooth and Bluetooth Low Energy) and IEEE 802.15.4 (used by Zigbee, 6LowPan, Thread, etc.) standards [2, 3]. They mostly rely on multi-hop topologies to achieve a larger coverage. Lately there is a push towards Low Power Wide Area Networks (LPWAN) providing connectivity over larger distances, but those are limited in throughput [4]. Examples of LPWAN wireless standards include IEEE 802.11ah, Sigfox, or Lora. Based on the requirements set forward by the network operator (e.g. penetration characteristics, transmission distance, achievable throughput, etc.) a selection should be made regarding the optimal wireless standard. However, as Figure 4.1 showed, there is no single technology which outperforms all other technologies, making the eventual choice more difficult.
To simplify the choice, for several use cases it could be beneficial to combine several wireless standards in a single device, in order to create a more optimal solution. This can be achieved through the use of multi-modal wireless platforms,

which combine multiple radio standards on an individual hardware platform. Making use of these multi-modal IoT platforms does not necessarily have a big impact on the unit price, given the ever decreasing costs of radio chips. However, it has an impact on other parameters as energy consumption, and comes at an additional processing cost.
The need for these multi-modal platforms has already been expressed by the research community.

> *"Given the growing numbers of multi-radio consumer devices, mobile network operators seek to leverage spectrum across diverse radio technologies, thus boosting capacity and enhancing quality of service."* [5]

> *"We envisage that future IoT devices will also incorporate multi-radio interfaces and power consumption for driving the multi-radio interfaces will be one of technical issues"* [6]

> *"To support devices with dynamic application requirements, multi-technology devices are required. In addition, the availability of multiple technologies is also important for mobile nodes that currently suffer from frequent connectivity outages. This means the ability to seamlessly switch from one technology to the other and to form even multi-hop LPWAN networks. "* [7]

Unfortunately, the creation of multi-modal devices does not only consist of adding additional radio chips into the hardware design. Due to the previously mentioned limitations of IoT devices, multi-modal platforms require an extensive redesign of the wireless stack, especially to the MAC layer. Typically, a MAC protocol dictates how and when the shared wireless medium can be accessed in order to minimize the possibility of packet collisions. When using multi-modal devices, each available interface should be controlled by its own MAC protocol. Most current architectures sequentially activate the available MAC protocols, so no two protocols can be activated at the same time. This behavior has some key problems: (i) it is impractical to efficiently perform simultaneous transmissions or receptions at two or more interfaces, (ii) without complex orchestration mechanisms it is impractical to know if a given interface of a node is active, and therefore listening for incoming connections, and (iii) all state information is lost between each interface switch, which can cause unwanted overhead (e.g. the loss of IEEE 802.15.4 Time Synchronized Channel Hopping (TSCH) mode synchronization information). These issues are the main reason to step away from sequentially activated interfaces, in exchange for parallel execution of multiple active MAC protocols. The concept might seem straight forward at start, but a Pandora's box of unexplored issues appears. They are mostly related to the fact that MAC protocols can interrupt one

another, and thereby influence MAC protocol timings/behavior. These interruptions can amount for a significant time loss, due to the slower MCU clock speed of IoT platforms. Especially for time-sensitive MAC protocols such as Time Division Multiple Access (TDMA), TSCH or Low Power Listening (LPL), this may have undesirable consequences, resulting in packet loss, increased delay, and increased power consumption.
This chapter details current approaches on how to implement MAC protocols on multi-modal devices, as well as giving a vision of how performance gains can be achieved. More specifically the chapter contains the following contributions:

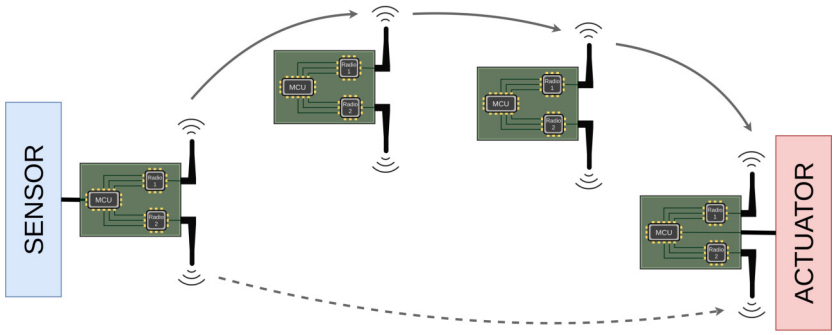
1. The problems related to multi-modal MAC design are elaborated;
2. The current state-of-the-art design approaches are detailed;
3. Several possible future solutions are proposed, in order to fix the multi-interface issues;

The remainder of this chapter is structured as follows: firstly Section 4.2 gives an overview of use cases which could benefit from the use of multi-modal hardware platforms. Next, Section 4.3 lists the possible design approaches which can be followed to implement multi-modal MAC protocols. Lastly, Section 4.4 concludes the chapter.

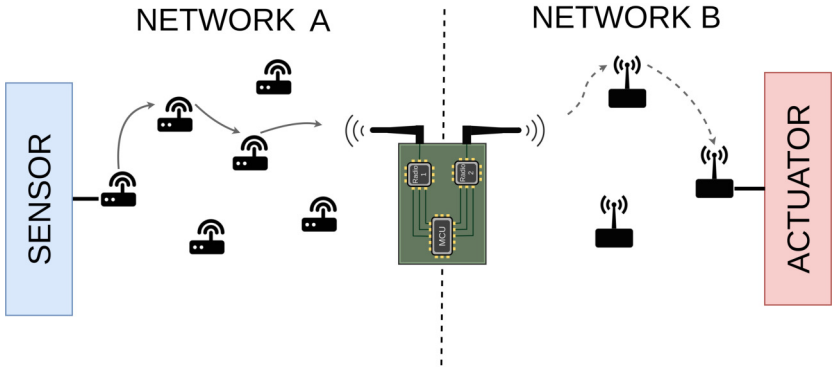
4.2 Use cases and scenarios requiring multiple MAC protocols

Through the use of multi-modal wireless platforms, some advanced use cases become possible to achieve. This section provides three use cases which would benefit by making use of these concepts, each of which has a visual representation if in Figure 4.2.

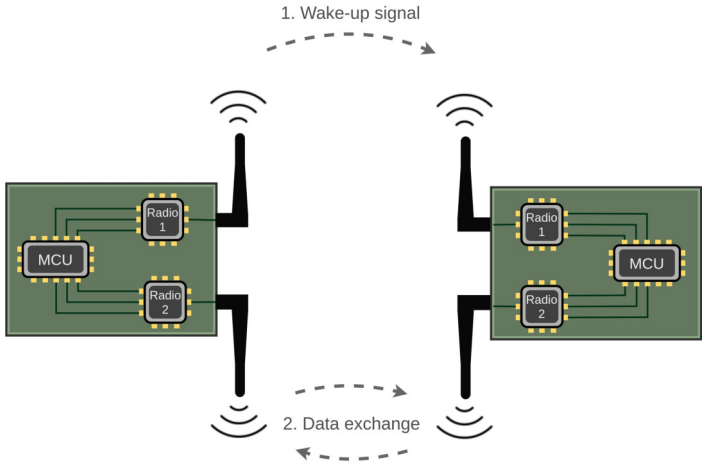
- **During the lifetime of a wireless device, it might encounter a variety of networking conditions and application requirements**: a periodic monitoring service might require close range, low energy, high throughput communication (e.g. available through LPWAN technology), but an alarm signal might require low latency and long range communication (e.g. available via Bluetooth Low Energy (BLE)). Intelligent optimization techniques can be used in order to select the optimal radio technology per given situation [8]. In Figure 4.2a the sensor uses the multi-hop close range radio for non-critical application traffic, and the long range radio to directly send the data to the actuator with a resulting lower latency.
- **Interconnection of different WSN's who are not necessarily using the same wireless technology via a mesh network.** Per network there should



(a) All nodes are deployed with two interfaces. The choice of interface is based on the applications requirements (in this case based on the latency).



(b) Two networks are interconnected via a border router. Since each network has their own radio technology, the border router should have two interfaces available in order to forward the data packets from one network to another.



(c) Multi-modal radio platforms can also be used to optimize energy consumption: the secondary lower current consumption radios are used as wake-up signal for the primary, more power hungry, high performance radio.

Figure 4.2: Use cases which can benefit from the use of multi-modal radio platforms.

be one (or more) devices who are able to communicate via two links, in order to connect the different networks (see Figure 4.2b). This enables continuous connectivity between heterogeneous objects (or networks), and it enables relay of data from several networks to a central server by using a multi-modal IoT gateway. The implications on the network layers is a subject which has already been extensively studied in [9].

- **Multi-modal radio platforms can be used in order to minimize the energy consumption of the platform.** By including a secondary low-power, wake-up radio, it is possible to completely turn off the primary, power consuming, but high performance radio. Only when the device receives a trigger on the wake-up radio, or when it has a packet to transmit, it will turn on the primary communication radio.

4.3 Multi-modal MAC design approaches

As explained earlier, MAC protocols are highly dependent on time accurate execution of instructions. The occurrence of interrupts, which delay the time critical execution, can have a negative impact on the medium access and can lead to packet loss, loss of battery capacity, increased delays, etc. When these interrupts are not related to the MAC layer, this could be fixed by giving the MAC layer interrupts priority over the rest of the network stack. When using multi-modal devices, each radio requires a MAC protocol to coordinate its medium access. Now issues may arise when two overlapping Interrupt Service Routines (ISR's) need to be executed, each of them originating from a different MAC protocol. The question of who to give preference over the other is not straight forward, and this chapter investigates the mechanisms to cope with this issue.
This section gives an overview of possible approaches/architectures to achieve multiple MAC protocols running on a single hardware platform. The (dis)advantages will be clearly listed, so an optimal decision can be made based on the requirements put forward by the MAC designer. Experiments were conducted between two emulated devices, showing a wireless performance comparison between the various solutions: Table 4.2 lists the maximum achieved throughput and execution latency (duration between the scheduled and eventual execution time of the code blocks) per MAC protocol.

4.3.1 Switch between multiple pre-installed MAC protocols

Most current multi-modal software architectures can only activate a single MAC/PHY-combination each given moment [10–15]. Because no two protocols are activated in parallel, no additional implementation effort is required on the side

Table 4.2: Maximum throughput results comparing the various multi-modal MAC solutions. Two radio interfaces are being used to set-up unidirectional traffic between two emulated nodes: (i) a subGHz interface at 31.5kbps (TI CC1200) and (ii) a 2400MHz interface at 250kbps (TI CC2538). The results can be compared to a baseline test, where only a single interface was active.

Solution	Protocol	Interface	Throughput [kbps]		Execution latency [μs]	
			Average	Stdev	Average	Stdev
Baseline (one active interface)	CSMA	2400MHz	106.80	14.21	0	0
	TDMA	subGHz	23.69	3.25	0	0
(a) Sequential protocols (4.3.1)	CSMA	2400MHz	52.73	13.79	0	0
	TDMA	subGHz	11.91	3.47	0	0
(b) Dedicated TDMA - Default (4.3.2)	TDMA	2400MHz	16.30	1.32	0	0
		subGHz	16.41	1.61	0	0
(c) Dedicated TDMA - Shared slots (4.3.2)	TDMA	2400MHz	22.57	1.87	0	0
		subGHz	23.10	1.95	0	0
(d) Code blocks - No priority (4.3.3)	CSMA	2400MHz	55.38	6.21	6573.14	13231.33
	TDMA	subGHz	17.92	3.37	679.52	1222.14
(e) Code blocks - TDMA priority (4.3.3)	CSMA	2400MHz	44.06	5.49	7326.5	15187.83
	TDMA	subGHz	23.65	2.44	96.33	304.82
(f) Slot based (4.3.4)	CSMA	2400MHz	84.39	0.84	0	0
	TDMA	subGHz	21.13	1.06	0	0

of the MAC protocols themselves. However, in order to switch between the available protocols, a MAC controller is needed which decides on the active protocol based on pre-defined logic. For example such a switch can occur if another protocol performs better given the current networking conditions and/or the application requirements.

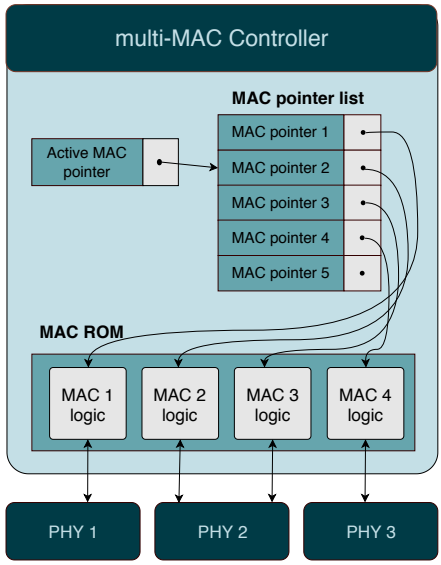


Figure 4.3: Controller logic in order to have multiple MAC protocols installed on the device. The MAC controller has an interface in order to control the active MAC protocol at each instance.

Sequentially activated protocols have a number of advantages, which still make them applicable for some use cases: (i) some of the multi-modal platforms only take into account a single, shared antenna into the hardware design [16], hereby only allowing a single Physical Layer (PHY) to be active at each moment, (ii) it is a comprehensible and easy implementable solution, and (iii) the time execution is assured since there can be no overlapping interrupts between pre-installed MAC protocols. However, sequential MAC protocols have a number of drawbacks. Firstly, the overhead of protocol switches is non-negligible because the context information is lost every time a disable occurs. Therefore the protocol needs to be (partly) re-initialized, and the context information reloaded. In the case of slot-based protocols such as TDMA or TSCH this can take a significant amount of time since the initialization is only finished as soon as a synchronization beacon is received. Secondly, there is no possibility of parallel execution, and therefore there is no optimal use of the available physical throughput of the radios. This can be observed in Figure 4.4 and Table 4.2 for which an experiment was conducted

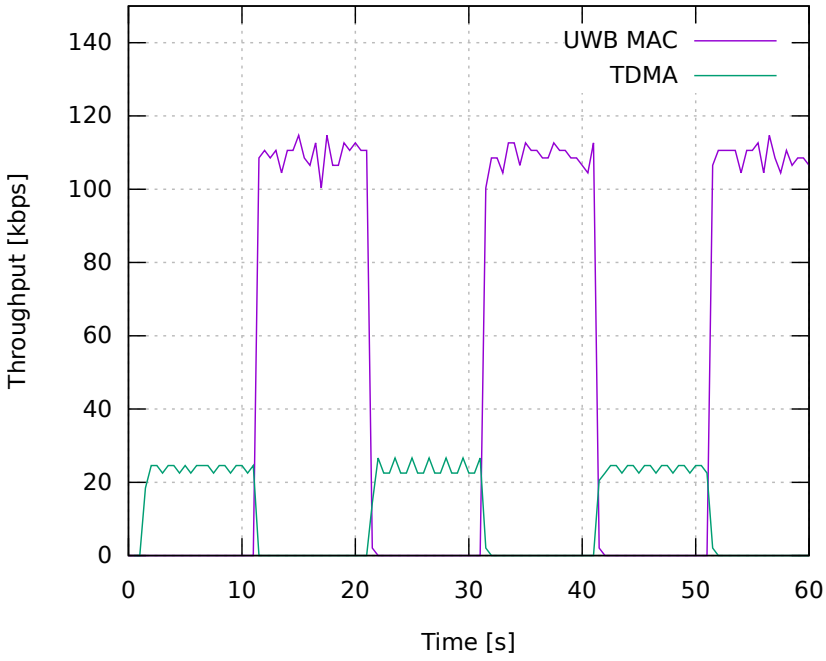


Figure 4.4: An experiment using two different sequentially activated MAC/PHY-combinations (TDMA - subGHz 31.5kbps (TI CC1200) and CSMA - 2400MHz 250kbps (TI CC2538)). Every ten seconds, a MAC protocol switch occurs. The PHY layers are not able to transmit/receive simultaneously, and must wait to be activated.

where two MAC protocols shared the time equally. Lastly, it introduces overhead to the network as wireless nodes need to negotiate which MAC/PHY-combination to use throughout the network.

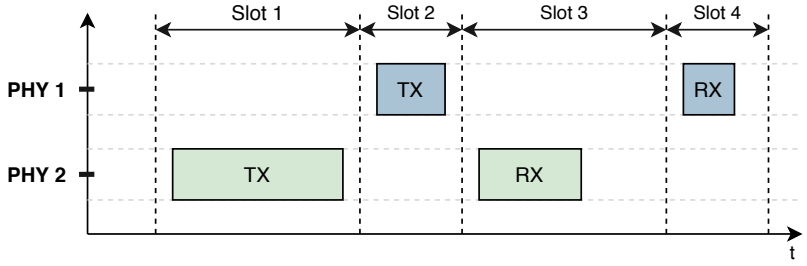
4.3.2 Single MAC using multiple interfaces

To achieve multi-modal software architectures, it is possible to use a single dedicated MAC protocol which controls all available radio interfaces. This guarantees timely execution of MAC instructions by separating the radio interactions with MAC instruction logic. Slot-based protocols such as TDMA or TSCH, are a well suited fit as they have the possibility to divide time into time slots. A scheduling algorithm typically assigns these time slots to individual wireless nodes, to either transmit or receive data. By appending a PHY layer selection as meta-data to the slots, the wireless device can determine on which radio interface to perform the action. The scheduling algorithm can schedule individual radio technologies within the context of a super-frame, so medium access cannot be unexpected or

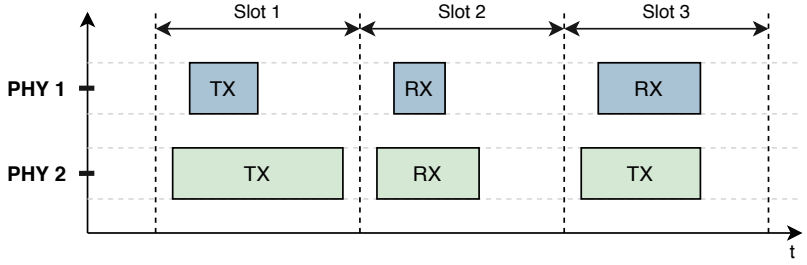
overlapping. Figure 4.5a shows an example TDMA superframe allocation, where radio technology one is allocated slot 2 and 4, and technology two is assigned slot 1 and 3. To optimize the time usage, the slot duration is altered depending on the maximum transmission duration of the active radio technology. In related work, this mechanism is for example used to minimize the energy consumption of a secondary power-hungry Ultra Wide Band (UWB) radio interface [17–19]. By only activating the UWB radio in dedicated time slots, the overall energy consumption of the solution can be diminished. Even though the solution delivers full code execution separation, still some drawbacks can be identified. Firstly, the assignment of slots requires an extension of current slot-based scheduling algorithms. Per slot negotiation of the active technology is required, which inevitably results in computational and medium overhead. Secondly, the solution does not use the complete capacity of the wireless radios, as only a single radio can be activated per slot, resulting in lower throughput (see Table 4.2 (b)).

The TDMA-based solution can be extended to allow for parallel execution of multiple interfaces, by scheduling multiple transmission/reception opportunities within a single slot. Practically, this is achieved by (i) preparing all selected radio interfaces for TX/RX-operations, (ii) starting the TX/RX-operations on dedicated offsets within the time slots, and (iii) waiting for all operations to be finished to start further processing. The duration of the slot should be long enough so all medium interactions, radio turnarounds and processing can be finished within the context of the slot (while taking into account some margin). While not negating the possibility of overlapping interrupts, the slot structure can handle the impact as the possible interrupts sources are known prior to execution. As the duration of the slot is large enough, all interrupts can be handled before the execution of the next slot. An example of the extension is shown in Figure 4.5b. Note that the MAC designer is not limited to which operation (RX or TX) to start on each individual radio interface, as each combination is viable. Thus, the solution actually consists of independent sub-TDMA's per radio interface, only sharing the same time reference. The scheduling algorithm thereby becomes less complex, as they can be performed on a per-interface basis, only ensuring that synchronization information is exchanged over all available interfaces. Additionally, the solution also leads to a possible throughput increase compared to non-shared slots (see Table 4.2 (c)). A negative aspect could be a possible discrepancy between the physical throughput of the PHY layers. If one of them is noticeably slower compared to the others (e.g. a 1.2kbps sub-GHz interface versus a 250kbps 2400MHz interface), the faster one will suffer as the slot duration will be big while still only a single packet can be transmitted. This could be solved by sending multiple data packets within the context of a single slot, again increasing the complexity of the solution.

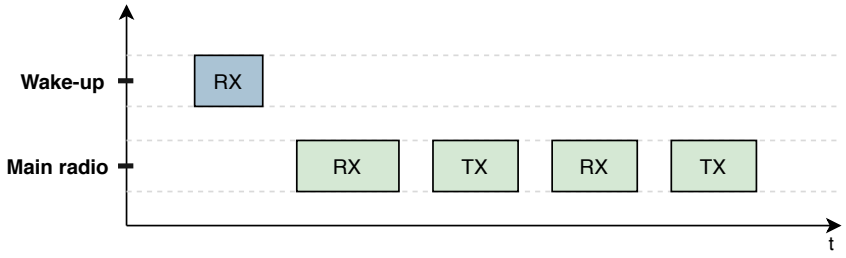
Another example of a single MAC protocol in combination with multiple radio interfaces consists of the use of a WuR. Power-hungry devices become equipped



(a) Each slot of the TDMA protocol holds a single transaction (TX or RX). There is no overlap between transmitting and receiving data.



(b) Parallel execution on multiple interfaces. The transmission and reception state is started on dedicated time offsets within the time slot.



(c) Through the use of a secondary Wake-Up Radio (WuR), the primary communication radio can remain in a power saving state. Only when a wake-up request is received does the primary radio need to be activated.

Figure 4.5: A single TDMA protocol, controlling two (or more) interfaces.

with a secondary WuR, which has a near-negligible power consumption in receive mode. Due to this key characteristic, the radio can indefinitely keep listening for any incoming communication messages or signals. If a node wants to set up communication, it first transmits a message over the WuR interface. After receiving the signal, the core more power-hungry radio can be turned on and perform data communication. An example of a WuR is shown in Figure 4.5c, where a wake-up signal on the WuR is followed by a bi-directional message exchange.

In conclusion, radio access happens within the context of a single (somewhat more complex) MAC protocol. However, this enforces the use of the same MAC protocol on all devices. This results in problems when of-the-shelf devices are included into the network, where there is no control over the MAC layer. Additionally, it is not always required to have all the features that this ”more advanced“ MAC protocol supports. Some protocols are more (power) efficient compared to the described TDMA solution, which can be power hungry due to the continuous synchronization effort, especially on long term deployments.

4.3.3 Code division in non-interruptable blocks

As stated before, the main issue with multi-modal MAC protocols is the possibility of overlapping interrupts resulting in increased timing delays. For example, if the execution of a TSCH slot is interrupted by another MAC protocol, the remaining slot execution is postponed. This can result in slot specific timings such as transmission/reception opportunities not being met. Another problem can be identified in Carrier Sense Multiple Access (CSMA) based protocols, where a positive Clear Channel Assessment (CCA) check should be followed by an immediate transmission attempt. If the delay between the CCA check and the actual transmission is too large, the medium can be occupied again due to another node starting a transmission, resulting in packet collisions. A multi-modal MAC protocol should hence negate the possibility of interrupts within critical moments during protocol execution.

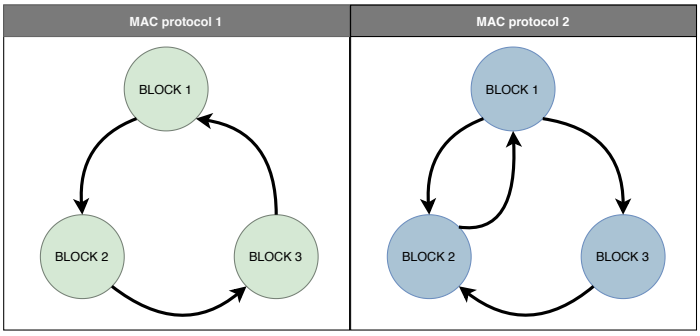
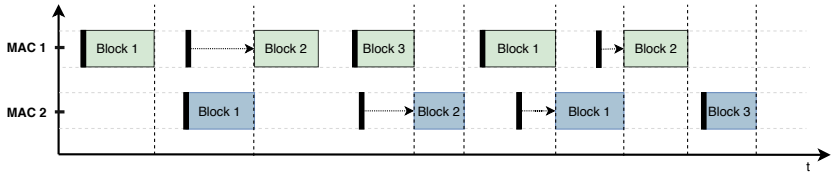


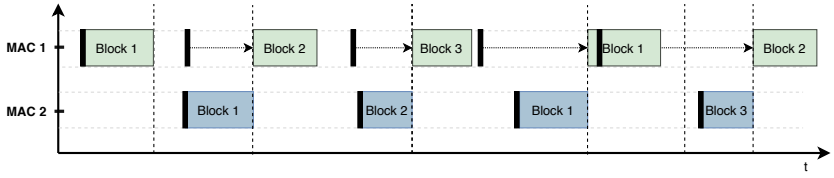
Figure 4.6: The non-interruptable code blocks make use of a stateful architecture, where each code block execution is represented by a different state.

To fulfill this need, another possible multi-modal MAC protocol approach consists of splitting the protocols in non-interruptable code blocks. These non-interruptable code blocks do not allow to be interrupted during execution time, thereby guaranteeing their internal timings. For example, in the case of TSCH a code block could be equivalent to a single time slot, or in the case of CSMA it could consist of a

CCA check followed by a packet transmission. The structure of MAC protocols thereby become state diagrams, where depending on the current state a different non-interruptable code block gets executed. An example representation of two MAC protocols is shown in Figure 4.6. This approach requires a multi-modal MAC controller or scheduler which dictates which block is activated at each given moment. The controller only passes interrupts to the intended MAC protocol, when one of their code blocks is activated, otherwise the context information is stored for later use. An example execution is showcased in Figure 4.7a. Note that sometimes the execution of a code block is postponed due to another block being active.
In Table 4.2 (e), it can be observed that both protocols have a largely decreased throughput compared to the baseline, and a varying amount of execution latency. In TDMA, and in extension all time sensitive protocols, the throughput loss can be accounted to the execution latency, as radio interactions are performed too late compared to their designated time offsets: e.g. if a receiver turns on its radio 1ms too late, it will likely be too late to receive the start of frame of a possible incoming packet.



(a) No priority between the code blocks.



(b) The code blocks are executed where MAC2 is executed with a higher priority compared to MAC1.

Figure 4.7: Examples using the non-interruptable code block architecture.

To improve results for the time-sensitive protocols, prioritized execution was introduced where each MAC protocol is given a priority. This mechanism postpones a code block if its execution overlaps with a different MAC of higher priority (see Figure 4.7b). To calculate possible execution overlaps, it is required to know beforehand how long a code block (maximally) requires to execute. By now giving a higher priority to time-sensitive protocols, the execution latency can be reduced, as the execution is not delayed anymore due to other active code blocks. The same

experiment was conducted between CSMA and TDMA, but now TDMA had a higher priority (see Table 4.2 (e)). For TDMA this resulted in a more stable result, as the throughput increased and the execution latency decreased, whereas the CSMA performance dropped since its execution was sometimes delayed due the TDMA being active.
The described mechanism, both with and without prioritized execution, showed to be a viable solution to have any combination of MAC protocols running on the same multi-modal device. Therefore it is also able to communicate with off-the-shelf devices, if the same MAC protocol is used. However, it still has a number of drawbacks, possibly holding it back from widespread adoption:

1. the performance is highly dependent on the set of installed protocols, as simultaneous execution of two (or more) time-sensitive protocols is still detrimental for the lower priority protocol;

2. the solution is not scalable towards a higher number of installed protocols.

4.3.4 Slot-based instruction execution

Another approach to achieve in parallel executed MAC protocols is slot-based instruction execution. Similarly like slot-based techniques such as TDMA or TSCH, the time is divided into time slots. However, instead of assigning these time slots to individual wireless nodes for medium interactions, the time slots are assigned to MAC protocols. Within the slot, the active MAC protocol has the sole access rights over the processing unit, which means that it will not be interrupted in its execution. After the time slot has finished, the MAC protocol must wait until the next assigned time slot. It is important to know that the slots are divided on a per-device basis, and hence no inter-device synchronization is required. An example of the concept can be observed in Figure 4.8, where two different MAC protocols are executed simultaneously (protocol 1 is transmitting a data packet, while protocol 2 is receiving a data packet).
The MAC protocol developer is free to perform any operation within the context of the time slot, however the execution should be finished within the duration of the time slot. Hence, the operations should have a deterministic behavior so the duration is known prior to execution. Additionally the duration of the time slots should be limited, since it delays the execution of the other protocols. Therefore, we advise to only execute a single instruction within the time slot, to minimize execution delay for other protocols (e.g. copy a packet to a buffer, give the transmission signal, perform a calculation, etc.). To determine the slot duration, the MAC developer should measure the individual instruction timings, and take the maximum as time slot duration.
One could argue that the slotted approach leads to even bigger execution delays (depending on the slot allocation), compared to the occurrence of unexpected in-

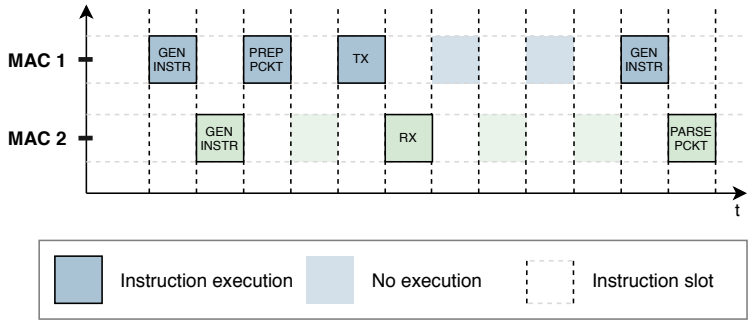


Figure 4.8: The MAC execution is divided in time slots, which get assigned to the different protocols. The active protocol can execute a single instruction within the context of the slot.

terrupts. However, the execution delay is deterministic, and can be taken into account during the design of the MAC protocol. The resulting protocols will thus be less performant, however their internal timings can be met, resulting in stable and expected performance. The slot based execution performance for both protocols did not reach the single protocol performance level, however it outperformed the code division in non-interruptable blocks by quite a big margin (see Table 4.2 (f)). Additionally, some other advantages can be identified: (i) it does not require negotiation between the installed protocols as the execution times are fixed, (ii) it does not require inter-device negotiation to determine which MAC protocol needs to be activated, and (iii) the solution is fully scalable towards any number of MAC protocols, however this results in increased MAC protocol timings.

4.3.5 Multi MCU hardware platforms

Due to the ever decreasing price of modern IoT processors (starting from a couple of Euros), combining several processors on a single platform is starting to become a possibility. Some commercially available chips such as the TI CC1352R already deploy two processors on a single device: one single powerful processor for application processing (Cortex M4), and a less powerful processor responsible for peripheral control (Cortex M0). Whereas the previous subsections focused on achieving multi-modal MAC protocols by separating their execution via software, it is also a viable option to run each different MAC protocol on its dedicated MCU. As the execution of each MCU is independent compared to the others, the execution of MAC protocols cannot mutually interfere with each other. As a result, the performance per protocol should be the same as if it were to be run on a single radio platform.

To control the different hardware components, a MAC manager is required which interfaces with the different MCU's. Most modern processors offer an on-chip bus protocol to allow communication towards other processors, in addition to easy integration of shared memory and on-chip peripherals. One such example is the AHB Lite interface on Cortex-M processors. At least, each MCU/MAC combination should offer the possibility to (i) enable/disable the protocols, (ii) pass packets for transmission, (iii) collect received packets, and (iv) offer getters/setters for internal MAC specific parameters.
The main drawback of a multi-MCU approach is the energy consumption of the device which increases dramatically, resulting in a smaller battery lifetime. For example the energy efficient LPC1100 based NXP processors consume more or less 13mA in active mode (7.3mA in sleep and 3uA in deep sleep mode). However, the solution is still applicable in non-battery powered nodes (e.g. a gateway), or when the processor can reside in deep sleep for the majority of (e.g. in low throughput use cases or duty cycled protocols).

4.4 Conclusion

In wireless (sensor) networks, different wireless standards enforce the use of a set of network protocols which work optimally in given networking conditions, or to accomplish a specific task. However, realistic deployments have dynamic requirements and changing networking conditions, resulting in difficulties when selecting an optimal wireless standard. This chapter advises to make use of multi-modal platforms, which deploy multiple radio chips in the hardware design. To coordinate network access, each of the radio interfaces should be controlled by a dedicated MAC protocol. Since most hardware designs only have a single processing core, the available MAC protocols share the same execution context. This results in protocols being able to interrupt each other, in the worst case leading towards broken MAC protocol timings and by consequence a degraded MAC protocol performance.
This chapter gave an overview of techniques to overcome the previously mentioned issues: (i) sequential activation of the protocols, (ii) a dedicated MAC protocol controlling all available interfaces, (iii) splitting protocols into non-interruptable code blocks, (iv) splitting protocol execution in to time slots, and (v) adding additional MCU's to the hardware design. Each of these techniques had a number of (dis)advantages compared to the others (summarized in Table 4.3). In conclusion, multiple MAC protocols can be deployed on the same device, however it still requires adaptations to the MAC controller software and/or the hardware design. The offered advantages of multi-modal devices outweigh the additional design effort, as it enables devices to cope with adaptive networking conditions and changing applications, and it enables seamless connectivity in heterogeneous networks.

Table 4.3: Comparison of the different solutions, in a number of key areas. Positive aspects are marked with ✓, while negative aspects are marked with ✗.

	Parallel ex.	**Off-the-shelf comp.**	**Performance**	**Battery efficiency**	**Network overhead**	**Scalability**
(a) Sequential protocols	✗	✓	✓	✓	✗	✓
(b) Designated protocol	✓	✗	✓	✓	✗	✓
(c) Code blocks	✓	✓	✗	✓	✓	✗
(d) Slot-based	✓	✓	✓	✓	✓	✗
(e) Multiple cores	✓	✓	✓✓	✗	✓	✓

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. *Wireless sensor networks: a survey*. Computer networks, 38(4):393–422, 2002.

[2] W. P. A. N. W. Group et al. *IEEE Std 802.15. 1-2005 Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)*. White Paper, 2005.

[3] J. Gutierrez et al. *IEEE standard for part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (lr-wpan)-draft d16*. IEEE-2002, 2002.

[4] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer. *A comparative study of LPWAN technologies for large-scale IoT deployment*. ICT express, 5(1):1–7, 2019.

[5] S. Andreev, M. Gerasimenko, O. Galinina, Y. Koucheryavy, N. Himayat, S.-P. Yeh, and S. Talwar. *Intelligent access network selection in converged multi-radio heterogeneous networks*. IEEE wireless communications, 21(6):86–96, 2014.

[6] Z. Abbas and W. Yoon. *A survey on energy conserving mechanisms for the internet of things: Wireless networking aspects*. Sensors, 15(10):24818–24847, 2015.

[7] E. De Poorter, J. Hoebeke, M. Strobbe, I. Moerman, S. Latré, M. Weyn, B. Lannoo, and J. Famaey. *Sub-GHz LPWAN network coexistence, management and virtualization: an overview and open research challenges*. Wireless Personal Communications, 95(1):187–213, 2017.

[8] J. Famaey, R. Berkvens, G. Ergeerts, E. De Poorter, F. Van den Abeele, T. Bolckmans, J. Hoebeke, and M. Weyn. *Flexible multimodal sub-gigahertz communication for heterogeneous internet of things applications*. IEEE Communications Magazine, 56(7):146–153, 2018.

[9] S. Bouckaert, E. De Poorter, B. Latré, J. Hoebeke, I. Moerman, and P. Demeester. *Strategies and challenges for interconnecting wireless mesh and wireless sensor networks*. Wireless personal communications, 53(3):443–463, 2010.

[10] C. Doerr, M. Neufeld, J. Fifield, T. Weingart, D. C. Sicker, and D. Grunwald. *MultiMAC-an adaptive MAC framework for dynamic radio networking*. In First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005., pages 548–555. IEEE, 2005.

[11] K.-C. Huang, X. Jing, and D. Raychaudhuri. *Mac protocol adaptation in cognitive radio networks: An experimental study*. In 2009 Proceedings of 18th International Conference on Computer Communications and Networks, pages 1–6. IEEE, 2009.

[12] S. Mudriievskyi and R. Lehnert. *Adaptive layer switching for smart grid applications in power line communications*. In 2016 International Symposium on Power Line Communications and its Applications (ISPLC), pages 115–120. IEEE, 2016.

[13] W. Wang, C. Dong, H. Wang, and A. Jiang. *Design and implementation of adaptive MAC framework for UAV ad hoc networks*. In 2016 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), pages 195–201. IEEE, 2016.

[14] M. Sha, R. Dor, G. Hackmann, C. Lu, T.-S. Kim, and T. Park. *Self-adapting mac layer for wireless sensor networks*. In 2013 IEEE 34th Real-Time Systems Symposium, pages 192–201. IEEE, 2013.

[15] G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, F. Gringoli, and I. Tinnirello. *MAClets: active MAC protocols over hard-coded devices*. In Proceedings of the 8th international conference on Emerging networking experiments and technologies, pages 229–240. ACM, 2012.

[16] J. M. Batalla, G. Mastorakis, C. X. Mavromoustakis, and J. Zurek. *On cohabitating networking technologies with common wireless access for home automation system purposes*. IEEE Wireless Communications, 23(5):76–83, 2016.

[17] J. Bauwens, N. Macoir, S. Giannoulis, I. Moerman, and E. De Poorter. *UWB-MAC: MAC protocol for UWB localization using ultra-low power anchor nodes*. To be submitted, 2020.

[18] N. Macoir, J. Bauwens, B. Jooris, B. Van Herbruggen, J. Rossey, J. Hoebeke, and E. De Poorter. *UWB Localization with Battery-Powered Wireless Backbone for Drone-Based Inventory Management*. Sensors, 19(3):467, 2019.

[19] M. Ridolfi, S. Van de Velde, H. Steendam, and E. De Poorter. *WiFi ad-hoc mesh network and MAC protocol solution for UWB indoor localization systems*. In 2016 Symposium on Communications and Vehicular Technologies (SCVT), pages 1–6. IEEE, 2016.

5

UWB-MAC: MAC protocol for UWB localization using ultra-low power anchor nodes

This chapter utilizes the multi-radio concepts elaborated throughout Chapter 4 for a novel multi-radio Medium Access Control (MAC) protocol. The UWB-MAC protocol is used to determine ranges between mobile tags and fixed infrastructure nodes. The protocol offers high power efficiency for the infrastructure nodes, which are assumed to be battery-powered. This is achieved by combining two different radio interfaces: a sub-GHz radio as communication interface, and an Ultra Wide Band (UWB) radio as ranging interface. A first implementation was created using the Time Annotated Instruction Set Computer (TAISC) framework, previously introduced in Chapter 2.

⋆ ⋆ ⋆

J. Bauwens, N. Macoir, S. Giannoulis, I. Moerman, and E. De Poorter.

Submitted in Elsevier Ad Hoc Networks

Abstract Indoor localization systems allow for innovative Industry 4.0 applications such as tracking of assets, people, or robots. Due to its cm-level accuracy, the UWB technology seems to be a perfect fit as an enabler for these advanced

use cases. Most current UWB research papers and commercial offerings assume that battery-powered mobile tags communicate with non-energy constrained infrastructure devices. However, in many deployments it is not possible to offer the required power cabling at the correct locations to provide energy to all infrastructure nodes. To this end, this chapter proposes a novel power-aware MAC protocol that uses a probed, secondary wake-up radio to maximize the battery lifetime on the anchor nodes. In order to show that battery powered infrastructure nodes are feasible, our solution accurately analyzes the long-term energy consumption of the infrastructure devices. Depending on the update rate and using a high capacity battery (10.4Ah), it is possible to achieve a battery lifetime between 1 to 16 years, which is comparatively up to four times better compared to the current state-of-the-art TDMA-based solutions.

5.1 Introduction

Localization systems, from which the Global Positioning System (GPS) is the most well-known, have already found their way in day-to-day life, enabling several important use cases such as asset tracking systems, locating cars and/or people, aviation, etc. However, these space-based satellite systems are very sensitive towards signal obstruction, creating coverage problems when no direct line of sight to at least three satellites is available. By consequence in indoor environments, but also in dense city environments with high-rise buildings, no accurate localization system can be provided using GPS based systems [1]. As such, GPS systems are not suitable for (indoor) use cases which require a seamless positioning system for indoor navigation, proximity detection to points of interest, proximity based advertising, healthcare applications, indoor asset tracking, autonomous drones/robots/AGVs, etc. [2]

To remedy this, over the years numerous indoor localization solutions have been investigated. Originally, solutions focused on traditional wireless technologies, for example Received Signal Strength Indicator (RSSI) based techniques using Bluetooth Low Energy (BLE) and Wi-Fi or proximity based mechanisms such as Near Field Communication (NFC). During the last number of years, the use of the UWB technology is starting to gain interest in the research world, as well as in industry due to its intrinsic much more accurate localization capabilities. Commercial implementations of this technology deliver a ranging error below 30cm [3]. These systems typically rely on fixed infrastructure with known positions ("anchor nodes") which need to be deployed within the environment. By determining the time-of-flight of the Radio Frequency (RF) signal between the mobile tag and the anchor nodes, it is possible to calculate the position of the mobile tag.

Most of these solutions assume that the anchor nodes are powered by the mains or Power over Ethernet (PoE), and the mobile tags are battery con-

strained devices. However, in many industrial environments this assumption is not realistic, as most existing buildings do not have the required power outlets and/or cabling to provide energy to all required anchors. The same is valid for non-building environments such as ships, trains, airplanes, etc. which do not take into account the requirements of localization infrastructure in their design. This makes the initial deployment cost of new localization infrastructure high, since it does not only consist of the anchor hardware, but also the cost of cabling. Additionally, although many mobile tag devices are indeed powered using a battery, these mobile tags also have the means to be charged regularly: an Autonomous Guided Vehicle (AGV) or a robot needs to be docked and charged often due to the power demands of the motors, and typical smartphone users charge their phone daily.

As such, **many deployments would benefit from solutions where the anchors are powered by (large-capacity) batteries.** This offers a lot of flexibility and ease of installation, as a new set-up can be deployed in a matter of hours. Unfortunately, most current state-of-the-art UWB scientific papers focus either on (i) localization accuracy [4–7], (ii) maximum achievable update rate [7, 8], or (iii) battery efficiency of mobile tag nodes [9, 10]. The anchor nodes typically use an always-on approach where anchors continuously listen for incoming ranging requests. Even when using large-capacity batteries, this means that the anchor nodes would have a battery lifetime counted in days, rather than years or even months. The main premise of this chapter is that anchor nodes could conserve energy by disabling their radio during stand-by periods when no mobile tags or range requests are present. To this end, **this chapter proposes a novel power-aware MAC protocol that supports additional signaling to wake up the UWB anchors, whenever ranging is required.**

The main contributions of this chapter are as follows.

- We propose UWB-MAC, an innovative contention based multi-tag MAC protocol whereby anchor nodes are only activated when mobile tags with range requests are present.
- A low complexity anchor node selection and wake-up algorithm is described to scale the solutions towards large deployments where multiple areas need to be covered.
- The performance of the RF protocol is experimentally and theoretically compared to state of the art solutions, showing that our work significantly decreases the energy consumption of the UWB infrastructure nodes.

The remainder of the chapter is structured as follows. First Section 5.2 describes the current state-of-the-art regarding current MAC protocols using the UWB RF technology. Next, Section 5.3 describes the high-level concept of the protocol,

describing the division in several protocol phases. Section 5.4 gives a detailed overview of every protocol phase. Section 5.5 evaluates the solution, focusing on the achievable energy consumption and update rate. Finally, Section 5.6 explains possible future extensions of the work, followed by Section 5.7 which concludes the chapter.

5.2 State-of-the-art

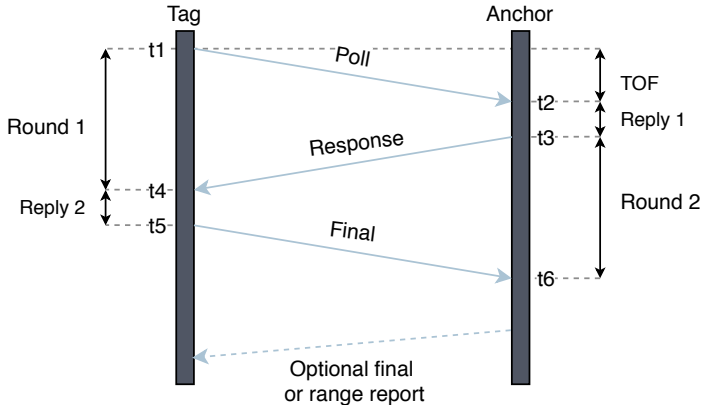
This section covers the state of the art of current UWB solutions, split between (i) the localization techniques to acquire either positions or ranges between nodes, and (ii) the MAC protocol logic which allows for communication between the devices.

5.2.1 UWB localization techniques

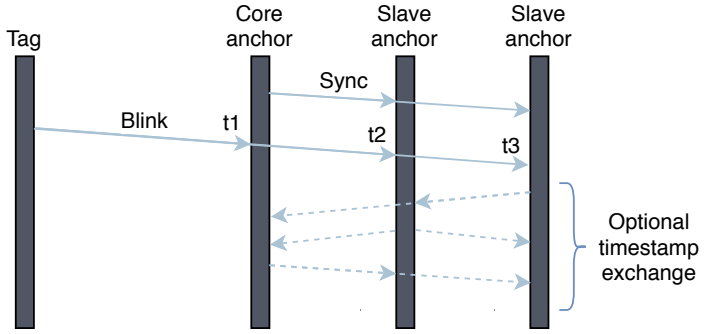
As stated in the introduction, UWB is an emerging technology in the context of indoor localization systems. Due to its high bandwidth and short communication pulses, it is extremely well suited to eliminate multi-path effects. By being able to very precisely determine the timestamp of the direct path, the time of flight of the signal can be deducted. In related literature, there are two commonly used techniques to achieve localization determination: Two Way Ranging (TWR) and Time Difference of Arrival (TDoA).
The TWR technique uses a two-way (poll and response) or three-way UWB message exchange (poll, response and final) to determine time of flight of the RF signal between tag and anchor. The exchange is visually represented in Figure 5.1a. By determining the transmission (t1, t3 and t5) and receive timestamps (t2, t4 and t6), the Time Of Flight (TOF) can easily be determined by using Equation 5.1 [11]. The final distance can be calculated by multiplying the TOF with the speed of radio waves. Only the node who receives the final message can calculate the range, which is typically on the anchor node. For the tag to know the distance to the anchor, either another UWB final message needs to be transmitted so the tag can also calculate the distance, or the anchor can report the calculated distance to the tag.

$$TOF = \frac{t_{round1}t_{round2} - t_{reply1}t_{reply2}}{t_{round1} + t_{round2} + t_{reply1} + t_{reply2}} \tag{5.1}$$

The key advantage of TWR is that no precise time synchronization between the anchors is required. However, TWR suffers from increased battery drain on the mobile tags, since at least three TWR packet exchanges per anchor node are necessary to calculate its distance towards the tag. By consequence also the the maximum achievable update rate is negatively impacted.
When using the TDoA scheme, the mobile tag only needs transmit a dedicated "blink" message towards the anchors (see Figure 5.1b). The receiving anchors



(a) Overview of a Two Way Ranging Two Way Ranging message exchange.



(b) Overview of a Time Difference of Arrival message exchange

Figure 5.1: Two typical techniques to determine position of mobile tags and/or the distance between a tag and anchor

determine the receive timestamp of the message, and subsequently exchange the timestamps over a wired back-end. However, since this chapter assumes a lack of wired infrastructure, the timestamps need to be exchanged using the RF medium. A position can now be calculated based on the time differences between the acquired timestamps, which is mathematically the same as finding the intersection of several hyperbola in 2D-space. The key advantage of this approach is the battery lifetime of the mobile tags, since their only source of consumption is the transmission of the short blink message. However, the overhead is shifted towards the anchor nodes, which need periodic additional message exchanges to (i) achieve sub-nanosecond synchronization to calculate accurate positioning, and (ii) to exchange the timestamps. Thus, since the focus of this chapter is to realize low-power infrastructure nodes, the TDoA approach is less suitable.

5.2.2 MAC protocols for UWB

A MAC protocol dictates how and when a radio is allowed to either access the medium, or when it should start listening for incoming transmissions. Due to their large bandwidth, most UWB radios are more power consuming compared to other RF technologies. As such, a MAC protocol that limits the amount of time the radio is enabled or in transmit/receive mode, allows energy to be conserved. Already several of these MAC protocols have been proposed, although most of these fail to meet the use case requirements stated in our introduction [18, 19]. An overview of several MAC protocols is given in Table 5.1.

For UWB transmissions, the traditional approach of Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) in the IEEE 802.15.4 standard is unusable, since no carrier sense can be performed on an UWB signal [20]. This is due to the low transmission power of the UWB signal which is even below the noise floor. Therefore, the IEEE 802.15.4a standard advises the use of the Aloha protocol, which transmits data with a random back-off as soon as the data is available, for both TDoA and TWR [12–14]. Unfortunately this is not a scalable approach, since the existence of multiple tags in the environment will result in ranging failures due to packet collisions [21]. Using an Aloha approach, the anchor nodes can not predict when a mobile tag will send a packet. As such, from an energy perspective, the Aloha protocol is efficient for the tags since the UWB radio is only enabled during a ranging sequence, but less for the anchor which have to permanently listen for incoming connections.

In order to minimize the occurrences of packet collisions, a slotted approach such as Time Division Multiple Access (TDMA) can be used. Each tag gets a dedicated time slot, in which it has unique access rights to perform a ranging sequence with the anchor nodes (either using TDoA or TWR), thereby negating the possibility of a collision [15, 16]. Additionally, anchor nodes can conserve energy due to the scheduled nature of the protocol: when no interaction is required, the UWB radio can remain in low power mode. However, in this approach additional overhead is introduced for (i) accurate time synchronization of the network and (ii) negotiation of the slot structure. To limit the energy cost of this approach, several solutions make use of a hybrid concept whereby an additional more power efficient radio (BLE, Wi-Fi, sub-GHz, etc.) is used to perform all non-ranging related communication [7, 17]. Even though the slotted approach is more power efficient, it has some drawbacks. Firstly, anchors still need be be permanently synchronized to a ”main“ synchronization anchor. Such multi-hop synchronization is not straightforward for large coverage areas. Moreover, synchronization needs to be maintained even when there is no tag in proximity to perform a ranging sequence with. Hence, the TDMA super-frame will remain empty and the synchronisation effort thus means an unnecessary loss of battery capacity. Secondly, these solutions require a complex slot allocation mechanism to (i) perform anchor selection,

Table 5.1: An overview of state-of-the-art approaches towards MAC protocol design using UWB radios. Compared to existing approaches, our solution significantly decreases the energy consumption of the anchor nodes (at the cost of higher energy consumption of the mobile tag).

Solution	Technique	Energy efficient tag	Energy efficient anchor nodes	Experimental validation	Theoretical validation
Aloha [12]	TDoA and TWR	✓✓	✗✗	✗	✗
TDoA (regular) [13, 14]	TDoA	✓✓✓	✗✗	✓	✗
TDoA (slotted) [15]	TDoA	✓✓	✗	✓	✓
TDMA (regular) [16]	TWR	✓	✓	✓	✓
TDMA (hybrid Wi-Fi) [17]	TWR	✓✓	✓✓	✓	✗
TDMA (hybrid subGHz) [7]	TWR	✓✓	✓✓	✓	✓
Our solution (UWB-MAC)	TWR	✗	✓✓	✓	✓

and (ii) to allocate dedicated multi-tag slots over multiple (potentially overlapping) TDMA cells.
The advantages and disadvantages of the above discussed techniques are summarized in Table 5.1. The UWB-MAC solution described in this chapter improves on the TDMA concepts by only performing synchronization when an actual ranging sequence needs to be performed. If that is the case, a small TDMA superframe (sufficient for a single localization exchange) that has only a local significance is scheduled. As such, we do not require complex multi-hop synchronization and only synchronize when localization is actually required. As a result, our approach significantly decreases the energy consumption of the anchor nodes.

5.3 UWB-MAC: high-level concept

Typically UWB deployments consist of two device types: (i) the fixed position infrastructure nodes which are denoted as "anchor" nodes, and (ii) the mobile nodes with a to be determined position which are called "tags". When considering the typical energy consumption of Internet of Things (IoT) platforms, most energy is consumed by wireless medium interactions and Micro Controller Unit (MCU) operations. The goal of UWB-MAC is to make sure that both radio and MCU of the anchor nodes reside in the least power consuming state, for the largest possible amount of time. To this end, the MAC protocol utilizes two distinct radio types: (i) a typical low power radio for negotiations (communication radio) and (ii) an accurate but more power hungry UWB radio for accurate ranging (ranging radio). This assumption is not unreasonable since many commercial and academic radio modules already combine multiple radio chips [22, 23]. Within a wireless network stack, the MAC layer dictates how the radio (and partially the MCU), and thus the most energy consuming component is used. For UWB-MAC, the goal is to ensure both radios from the anchor nodes to sleep during the majority of time.
The general concept for this protocol is visualized in Figure 5.2. The UWB-MAC protocol focuses on de-centralized, low-power operation in several distinct phases:

- **Phase 0**: during initialization, the mobile tags scan for any neighboring tags (see Section 5.4.1). If no neighbors are detected, the tag becomes the synchronization master and can initiate a contention phase. If a neighbor is detected, it synchronizes and joins its contention phase;

- **Phase 1**: the mobile tags contend for the access rights to the anchor nodes (see Section 5.4.2). The winning node is the first one to send a "wake-up" message over the communication radio interface after a random wait period. The wake-up packet contains the timing information for the next contention phase;

- **Phase 2**: the tags wake up the anchor nodes which are in close proximity over the same radio link (see Section 5.4.3). This trigger message or "probe" contains a schedule for the UWB ranging messages;
- **Phase 3**: the anchors acknowledge to which tag they will perform a ranging process (see Section 5.4.4);
- **Phase 4**: a slot based ranging process is started, where the distances between tag and anchor nodes is determined, via the well-known two-way ranging transmission scheme (see Section 5.4.4);
- **Phase 5**: in the final phase, the tag can optionally report the acquired ranges to a border router node, which subsequently uploads the ranges to a cloud service (see Section 5.4.4);

Note that anchor nodes are in their default sleep state until woken up by a tag. If no nearby tags are actively waking up the anchor nodes, they will remain in this state. By consequence they do not participate in phase 0 and 1. The detailed operation of each of the distinct phases will be elaborated in the next section.

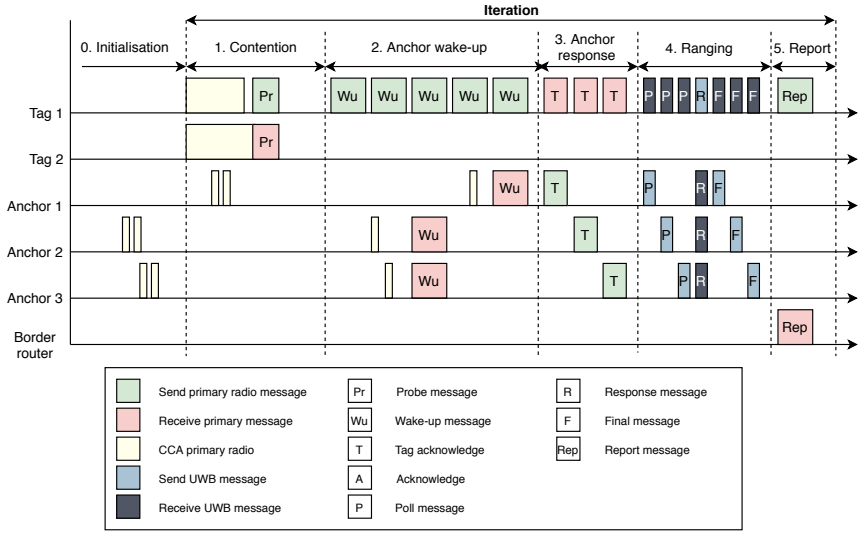


Figure 5.2: This figure shows the stages of the UWB-MAC protocol, consisting of (i) a contention phase between the mobile tags, (ii) an anchor wake-up phase where the "winning" tag can wake-up neighbouring anchor nodes, (iii) an optional anchor selection phase where anchor nodes respond to the tags, (iv) the final ranging phase where the ranges between tags and anchors are determined, and (v) an optional reporting phase. The primary radio is the communication radio, used to set-up a connection to the anchor nodes, before using the secondary UWB radio to perform ranging.

5.4 UWB-MAC: protocol specifications and analysis

The next subsections go into detail of every step of the protocol process. For each phase, detailed calculations are provided to determine the timing constraints and to calculate the energy consumption of the battery-powered anchor nodes. Please note that during a standard iteration, only phase 1 to 5 are active. Phase 0 is only activated when (re-)joining the network.

5.4.1 Phase 0: Initialization

When a tag first joins the network, it must be aware of which other tags are in its environment through the use of an initialization scanning period. The communication radio looks for any incoming probe packets, while the ranging radio is turned off. This detection mechanism is necessary to avoid future collisions on either the primary communication interface or the secondary ranging interface. By being aware of the environment, tags can 'negotiate' the medium access, and thereby minimise the possibility of further packet collisions.
When the scanning period is completed, two situations are possible:

1. If no close-by tags are detected, the tag becomes a synchronization master who dictates when every protocol iteration is initiated. For this purpose, the tag transmits a broadcast "probe" message, containing the timing information indicating the exact start of phase 1 of the protocol.

2. If one or more close-by tags are detected, marked by receiving a probe message, both radios are turned off until the start of next phase 1 iteration (dictated by the contents of the probe message).

Equation 5.2 dictates the minimal scanning period in phase 0, in order guarantee the reception of a probe.

$$t_{phase\_0} = t_{scan} >= t_{contention} + t_{iteration} \tag{5.2}$$

The scanning duration should be at least a complete iteration, in addition to the duration of the contention phase. The addition is required since the probe packet is transmitted randomly somewhere within the contention period.

5.4.2 Phase 1: Contention-based joining process

During **phase 1**, all tags turn on the communication radio to contend between each other for the exclusive access rights over the infrastructure nodes, while the ranging radio is still disabled. Only the winning tag may wake up the anchors. The others need to wait until a next iteration of the contention phase, which is repeated every $t_{iteration}$ period. The contention period is further detailed in Figure 5.3.

For contention, each tag picks a random duration between 0 and t_{max_r}, during which a scan for medium activity is performed. If no probe frame was received within the chosen duration, the tag sends its own probe frame, thereby notifying the environment that it "won" this contention round by choosing the shortest random period. The winning tag can now advance to phase 2. Tags who receive a probe, either in the initialization phase or during a contention round, determine that the transmitter of the probe has exclusive access to the anchor nodes to perform a ranging sequence. Hence, a low power mode is initiated which lasts for the rest of the current protocol iteration.

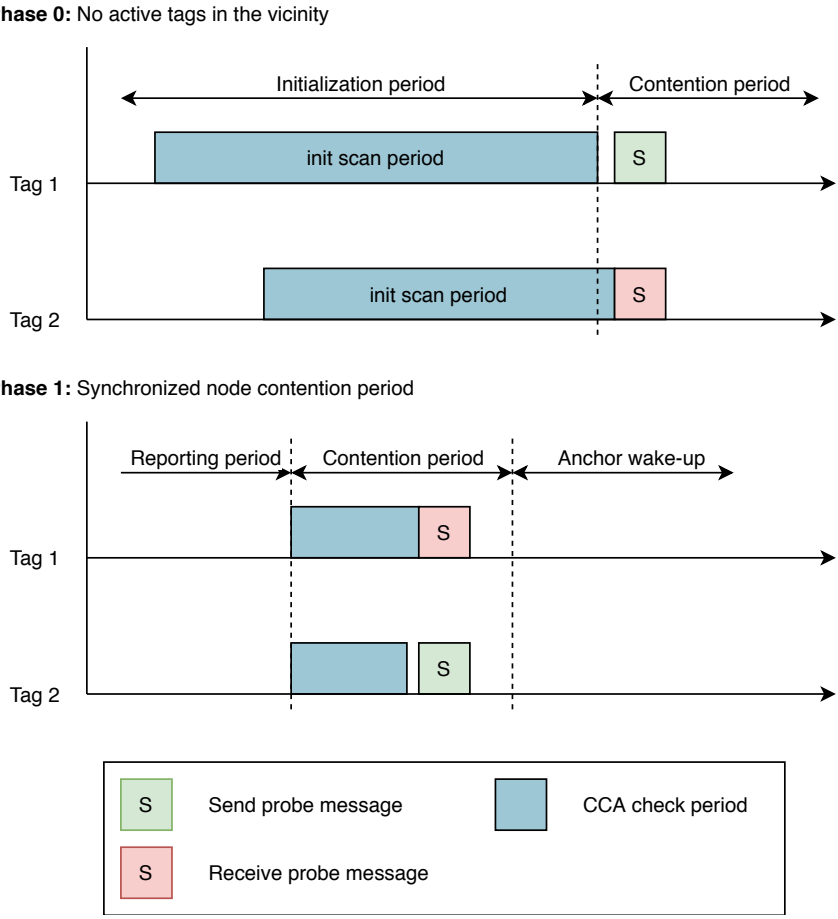


Figure 5.3: Phase 1: contention. (a) In situation 1, no active nodes are in the vicinity. The first tag to start sending (here tag 1), and therefore becomes the synchronization master. (b) In situation 2, the tags have already been synchronized, and contend between each other for the access to the anchors.

In large-scale deployments, tags that are too far from a group of synchronized tags are not able to receive the probe message anymore. Therefore, they determine their own time reference, and consequently multiple groups of synchronized tags are formed. Unfortunately, this makes the joining process more complex since tags can receive probing messages of different tag groups. After the initialization phase, the tag should join the tag group with the highest RSSI value. By periodically desyncing, and thereby restarting the initialization phase, the neighbor information will remain fresh and an optimal choice of tag group can be made. This will also solve the issue of mobility, where the closest tag group could be continuously changing.
Equation 5.3 formulates the minimal duration for the contention phase, which is mostly limited by the maximum random wait period t_{max_r}.

$$t_{phase\_1} = t_{contention} > t_{max\_r} + t_{probe} \tag{5.3}$$

Choosing a high value for t_{max_r} negatively impacts the achievable range update rate, whereas choosing a low value increases the probability of collisions when multiple tags send their probe simultaneously. Another factor to take into account is that switching between radio modes also requires a predetermined, radio-chip dependent time duration. Between the last positive Clear Channel Assessment (CCA) check, and the eventual probe transmission, the radio performs a receive-to-transmit turnaround with duration $t_{turnaround}$. If another tag concludes that the medium is free within this period, a collision still occurs. Equation 5.4 expresses the probability where multiple tags choose a colliding random value, with *n* the number of competing tags. One should tweak this equation in order to achieve an acceptable probe collision probability with a minimal t_{max_r}.

$$p_{probe\_collision} = 1 - \left(\frac{t_{turnaround}}{t_{max\_r}}\right)^n \frac{\left(\frac{t_{max\_r}}{t_{turnaround}}\right)!}{\left(\frac{t_{max\_r}}{t_{turnaround}} - n\right)!} \tag{5.4}$$

Until now, most energy is consumed by the mobile tags either by listening for or transmitting the probing message on the primary communication interface. The anchors remain in low power mode and are thus very energy efficient.

5.4.3 Phase 2: Low-power wake-up mechanism for anchor nodes

During **phase 2**, the tag that won the contention during the last phase will now send a wake-up packet over its communication radio. Upon receiving this wake-up packet, the anchor nodes know they have to activate their (highly energy consuming) ranging radio. The challenge is that the anchor nodes would have to turn

on their communication radio continuously to receive incoming wake-up packets from the surrounding tags, thereby consuming a significant amount of energy. For example, the operational lifetime of the anchor nodes in work from [7] is often largely limited by the energy consumption of its continuously-on communication radio, rather than the energy consumption of the ranging radio (see Section 5.5)
To decrease the wake-up times of the anchor nodes, inspiration can be drawn from Radio Duty Cycling (RDC) protocols such as LPL, ContikiMAC, etc [24]. These protocols allow for ultra low power receiver nodes, by periodically performing CCA checks to verify if a data transmission is occurring (see Figure 5.4). The receivers, in this case the anchor nodes, perform at least two consecutive CCA checks in order to check if there is any activity on the radio channel. The transmitter (the tag) sends a burst of packets on its communication interface in the hope that at least one is received by the anchor node. These so called wake-up packets contain the timing information for the start of next phases. If activity is detected during the CCA check period, the communication radio remains in receive mode, until the full wake-up packet has been received. The communication radio can now safely return to low power mode, and waits until the ranging sequence commences for further interactions. If no activity has been detected, the radio is turned off until the next CCA check sequence in t_{cycle_time}. Only during the CCA checks, the radio is put in a power consuming state. Note that the RDC is the default state of the anchor nodes, which is also active when (i) no tags are nearby, and (ii) during phase 0 and 1.

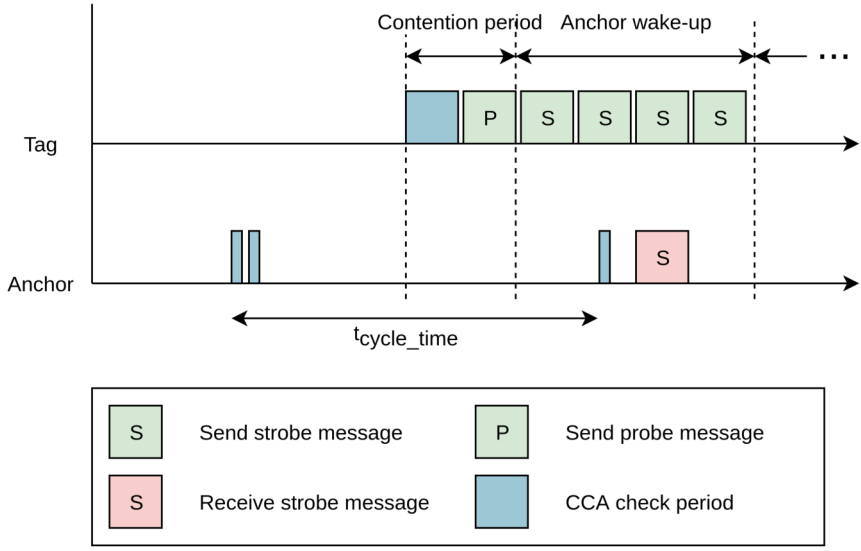


Figure 5.4: Strobing mechanism to perform a low-power anchor wake-up sequence

The constraints under which this duty cycling protocol works (e.g. minimum packet duration t_s, interval between CCA checks t_c, minimum wake-up sending duration t_{wakeup_time}, etc.) have already been documented in research [24]. To be compatible, this chapter makes use of the same symbols. The main impacting factor for the energy consumption of anchor nodes is the cycle time t_{cycle_time}. Within eqs. (5.5) and (5.6) the duration of phase 2 is expressed (t_{phase_2}), in addition to the constraints on the cycle time. By increasing the cycle time, the relative impact of either the CCA checks or receiving a wake-up packet, becomes less compared to the full cycle time duration. However, increasing the cycle duration decreases the possibilities of the tags to start ranging sequences, resulting in a lower ranging update rate. Hence tweaking t_{cycle_time} and t_{wakeup_time} is a trade-off between the ranging update rate, and the battery consumption of the anchor nodes.

$$t_{phase\_2} = t_{strobing} = (t_{wakeup\_tx} + t_i) \cdot n_{wakeup\_packets} \tag{5.5}$$

$$t_{strobing} >= t_{cycle\_time} + 2\,(2t_r + t_c) \tag{5.6}$$

The impact of changing the t_{cycle_time} is non-negligible for the tags. More wake-up packets ($n_{wakeup\_packets}$) need to be transmitted in order to guarantee successful reception on the anchor nodes. When using a sub-GHz radio as communication interface, this can be problematic when trying to meet the sub-GHz duty cycling requirements. Using a non-duty cycle limited, e.g. a 2.4GHz IEEE802.15.4 radio, might be necessary when t_{cycle_time} becomes too large.

5.4.4 Phase 3, 4 and 5: Ranging sequence

The complete ranging sequence consists of two parts: (i) the anchor selection, which is a two-way handshake where tag and anchor both acknowledge their intention to perform the ranging phase, and (ii) the two-way ranging phase where the effective ranges between tag and anchor are determined. Optionally, the tag can afterwards choose to send its range towards a core anchor node connected to a cloud service.

5.4.4.1 Anchor response

Throughout the second phase, tags performed a wake-up sequence in order to wake up neighboring anchor nodes. However, due to the un-centralized workings of the protocol, the tag remains unaware of which anchors have woken up to perform a ranging sequence. A selection scheme between tag and anchor is necessary, in order to negotiate the UWB medium access **(phase 3)**. A first possibility uses the ranging information from a previous ranging sequence, in order to assess the proximity from tags to anchors. This mechanism requires no additional energy

usage of tag and anchors, but is error prone. If the tag has moved too much from its original position, or if anchors did not hear the wake-up message, the anchors will not respond in the ranging phase. A more energy consuming mechanism, is for the anchors to transmit either a link-layer acknowledgment or a dedicated packet. This is a confirmation that (i) the anchor heard the wake-up message from the tag, and (ii) that the anchor will perform a ranging sequence in the next protocol phase.

$$t_{phase\_3} = n_{anchors} * (t_{s2tx} + t_{anchor\_response} + t_{rx2s}) \tag{5.7}$$

By combining proximity with the transmission of an Acknowledgment (ACK), a more optimal solution can be achieved. The tag determines which n anchors have the highest probability of being close to its position. Their node-ID's will be appended to the wake-up/probe message in a sorted list. Anchors use the offset within the list as an index for a slotted anchor response scheme. Within their allocated slot the anchor sends a response to the tag. When the tag has no prior information about its location, no initial anchor selection can be made. This is solved by sending an empty node-ID list. If an anchor receives such a message, a random index is calculated (between 0 and n). The handshaking is thus performed in this randomly allocated time slot. By doing so, there is still a random chance that multiple anchors will react within the same time slot. However, if at least one anchor is able to perform a correct handshake, this will offer enough localization information to perform a better localization guess in the next ranging sequence.

5.4.4.2 Ranging sequence

After the anchor selection, both tag and anchor gathered enough knowledge to perform a ranging sequence **(phase 4)**. This ranging sequence uses a similar two-way-ranging approach as described in the related work section (see Section 5.2). Instead of the beacon-based synchronization, this solution uses the timing information embedded into the strobe/probe-packets. By consequence all nodes have the same concept of time. To minimize the impact on energy consumption of the anchor nodes, the typical TWR roles are reversed, as the anchors do not need to reside in the energy consuming Receive (RX)-mode as much. The anchors initiate the ranging sequence by sequentially transmitting a unicast UWB "poll" message to the selected tag, based on their previously acquired ranging index. The tag responds by broadcasting a single UWB "response" message. Finally, the anchors sequentially send the "final" UWB message to the tag. Now the tags have all information necessary to calculate the distance towards each anchor point. The anchors are no longer responsible for sending the calculated range to the tag and/or core anchor point, thereby conserving energy. The tag can optionally conclude the protocol iteration by transmitting the range to the core anchor via its primary communication interface, depending on the application **(phase 5)**.

$$t_{phase\_4} = t_{s2tx} + (n_{anchors} * 2 + 1) * t_{uwb} + t_{rx2s} \tag{5.8}$$

$$t_{phase\_5} = t_{s2tx} + t_{report} + t_{rx2s} \tag{5.9}$$

5.5 Validation

Within this section, the performance of the novel UWB-MAC protocol is evaluated. The evaluation criteria are mainly focused on energy consumption and achievable update rate for the anchor nodes, however the energy consumption for tags is also quantified. For evaluation purposes, a model has been created providing the necessary evaluation output, based on all configurable input parameters. For both radios and MCU, widely used components were used: a TI CC1200 as primary radio, a DecaWave DW1000 as ranging radio, and an ARM Cortex M3 MCU. The anchors become equipped with a large capacity battery pack in order to maximize the battery capacity. The fixed hardware configuration is summarized in Table 5.2, while other more protocol specific settings are summarized in Table 5.3. Table 5.4 gives an overview of the protocol timings achieved with the previously mentioned settings.

Table 5.2: Component configurations for evaluation

Component	**Type**	**Setting name**	**Setting value**
Primary radio	TI CC1200	Center Frequency	868MHz
		Datarate	50kbps
		Modulation	2-GFSK
		Deviation	25kHz
		TX Power	10dBm
		Voltage	3V
Ranging radio	DecaWave DW1000	Mode	Mode 3
		Center Frequency	6489.6MHz
		Bandwidth	499.2MHz
		Datarate	110kbps
		PRF	16MHz
		Preamble	1024 symbols
		TX Power	-9.3dBm
		Voltage	3V
MCU	ARM Cortex M3	Frequency	16MHz
		Voltage	3V
Battery	Generic	Voltage	3.7V
		Capacity	10.4Ah
			38.48Wh

Table 5.3: Protocol settings used for evaluation (unless stated otherwise)

Protocol	Parameter name	Parameter value
UWB-MAC	Cycle time	100000
	Selected anchors per sequence	4
	Contending tags	4
	Probe payload size	14 bytes
	Strobe payload size	30 bytes
	Anchor response payload size	14 bytes
	UWB payload size	24 bytes
	Report size	20 bytes
TDMA	Selected anchors per sequence	4
	Contending tags	1
	Beacon payload size	14 bytes
	UWB payload size	24 bytes
	Report size	14 bytes

Table 5.4: Default protocol timings

Event type	Event	Duration
Packets	Probe	3520 μs
	Wake-up	6080 μs
	Anchor response	3520 μs
	UWB	3252 μs
	Report	4480 μs
Phases	Phase 1	13520 μs
	Phase 2	103280 μs
	Phase 3	26744 μs
	Phase 4	66032 μs
	Phase 5	5000 μs

The importance of low-power anchor nodes has already been explained thoroughly in previous sections. Therefore, this section gives a detailed evaluation of the energy consumption of the novel UWB-MAC protocol. The goal is to determine (i) what are the sources of energy consumption, (ii) what is the typical battery lifetime of the solution, (iii) how do configuration/application parameters influence the battery lifetime, and (iv) how does it compare to the current state-of-the-art solution and in which situations does it behave better.
Due to the high energy consumption of currently available UWB ranging radio's it is important to minimize the "on"-time by enabling a low(er) power mode when possible. For the UWB-MAC protocol, this was achieved through the use of a duty cycled primary radio. For evaluation purposes, a sub-GHz radio was chosen given its flexibility in configuration options, and its long range transmission capabilities. Figure 5.5 displays the energy consumption per phase of the protocol, as well as the source of the consumption.

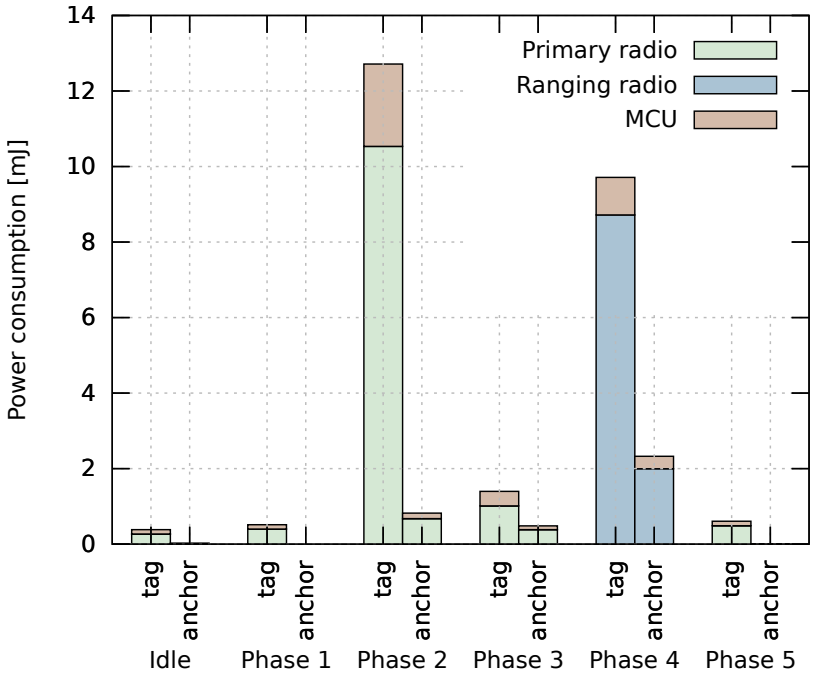


Figure 5.5: Energy consumption of each protocol phase, with a cycle time of 100ms. In addition, the the source of energy consumption is also plotted (primary radio, UWB radio, or MCU)

When no operation is required (**idle**), a minimal energy consumption can be achieved for both anchor (0.006mJ per cycle) and tag (0.266mJ per protocol it-

eration). The tags consume more energy since they need to transmit/receive a probe to uphold the tag synchronization, while the anchor only needs to perform a CCA check sequence. During **phase 1 and 2** the tag performs a number of consecutive strobe/probe transmissions on the primary interface, while the anchor only consumes energy due to the strobe reception. In **phase 3**, the anchors send a single packet. **Phase 4** (the ranging phase) is the main culprit of energy consumption for the anchor nodes, since the ranging radio needs to be woken up. During the last phase (**phase 5**), the anchor again has no operation.

Figure 5.6 compares the cumulative energy consumption per cycle during stand-by mode or ranging mode (for anchors and tags), between UWB-MAC and a state-of-the-art TDMA solution [7]. During a stand-by cycle of TDMA a single synchronization beacon is received, followed by an empty superframe. The active cycle is a single synchronization beacon, followed by a two-way-ranging superframe. It is apparent that, both for UWB-MAC and TDMA, the anchor consumption remains low if the anchor is kept in idle mode, or alternatively formulated if the number of ranging requests is kept to the minimum. However, the UWB-MAC anchor consumption in stand-by mode drastically outperforms the TDMA counterpart, due to the duty-cycled nature of the protocol. During a ranging mode, the energy consumption of the anchor nodes is more or less the same between the two protocols. The low energy consumption of the anchor nodes, comes at the cost of the mobile tags, as they consume far more energy in order to wake up the anchor nodes.

The goal of this work was to prolong the battery life of infrastructure-less anchor nodes. For this purpose the lifetime of a large-capacity battery was quantified in Figure 5.7, and compared to traditional always-on anchor nodes, and the TDMA solution. In all cases, the UWB-MAC protocol outperforms its competitors. Again, the advantageous nature of the solution is more apparent when the update rate is low. With higher update rates, TDMA and UWB-MAC convert to more or less the same battery lifetime, since the relative impact of the ranging communication starts to outweigh the impact of the synchronization communication.

The main drawback of the solution is the achievable update rate, as can be seen in Figure 5.8. This is a direct result of the large timing overhead of synchronization phases (phase 2 and 3). The update rate can be improved by lowering the protocol cycle time, but the energy consumption thereby also increases. In all cases TDMA and nullMAC can achieve a higher update compared to UWB-MAC. As a result, this protocol should not be chosen in use cases which require a high(er) update rate.

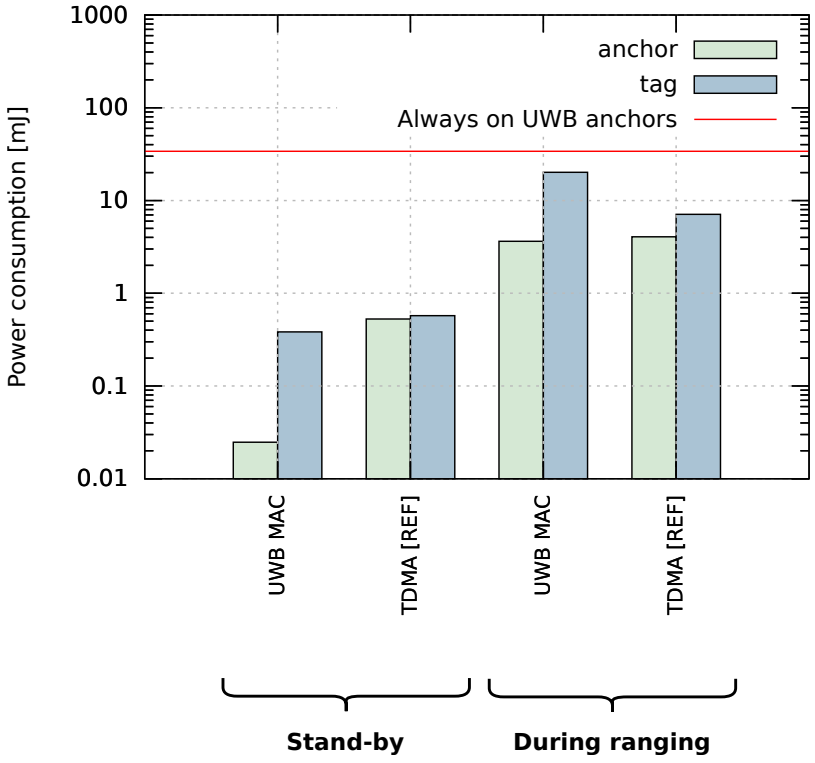


Figure 5.6: Comparison of energy consumption between UWB-MAC and TDMA [7] for a single protocol iteration. Especially when the anchor resides in stand-by mode, UWB-MAC outperforms always-on UWB anchors and the TDMA solution. Take note that the vertical axis is on a logarithmic scale

5.6 Future work

Several extensions of this work are still possible, which have the potential of further improving the protocol performance.

5.6.1 Low-power wake-up radios

Making use of duty-cycling protocols is a well-known mechanism to reduce the energy consumption for battery constrained devices. However, there exists an alternative which could be more efficient: a Wake-Up Radio (WuR). Power-hungry devices become equipped with a secondary WuR, which has a near-negligible energy consumption in receive mode. Due to this key characteristic, the radio can indefinitely keep listening for any incoming communication messages or signals.

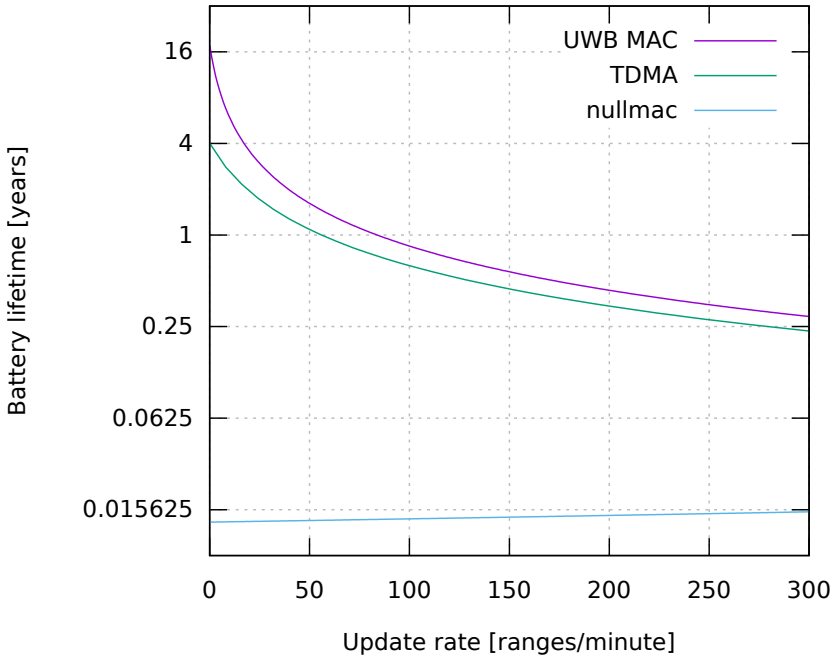


Figure 5.7: This figure shows a comparison between anchor battery lifetimes for three different solutions, in terms of the requested update rate. As can be observed, especially for low update rates, a large battery lifetime can be achieved through the use of the UWB-MAC protocol.

If a node wants to set up communication, it first transmits a message over the WuR interface. After receiving the signal, the core more power-hungry radio can be turned on and receive data communication. The inclusion of a WuR into the UWB-MAC protocol, could (i) allow for a higher update rate due to the reduced time overhead, and (ii) dramatically reduce the energy consumption for tags and anchors.

During the last decade, wake-up radios have been the subject of extensive research, where multiple designs were proposed. A lot of research papers make use of "fictional" wake-up radios, in order to verify their design [25]. However, lately they start to get included into modern radio designs (e.g. TI RF430F5978 or ScioSense AS3933). Still wake-up radios have some other disadvantages, holding them back for the adoption in modern hardware designs: (i) they are typically very low range (10-20m) due to a reduced receiver sensitivity, (ii) they are have a limited throughput, making them mostly unsuitable for further data communication, (iii) there is a lack of a stable hardware design, and (iv) there is a lack of a universal communi-

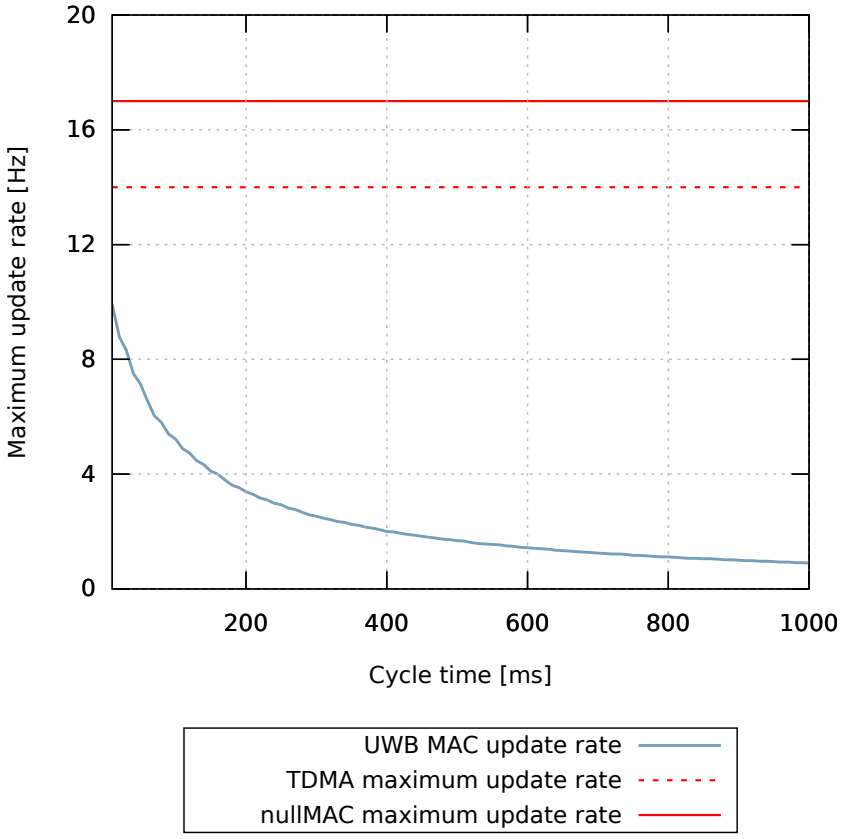


Figure 5.8: Achievable update rate with a given cycle time. By decreasing the cycle time, a higher update rate can be achieved (at the cost of battery lifetime). In any case, the achievable update rate is lower than the TDMA and nullMAC solution.

cation standard to the WuR.

5.6.2 Dynamic reconfiguration

Throughout the evaluation section, a static configuration of both radios was used. The solution would profit from dynamic re-configuration of the radios to further decrease the effectiveness and energy efficiency. For instance, by choosing a higher Modulation and Coding Scheme (on either radio), the air time of the RF signal can be reduced. This directly improves the energy consumption on a per packet basis. However, the achievable distance of the signal is negatively impacted.

5.7 Conclusion

This chapter promised to dramatically improve the battery efficiency of the infrastructure or anchor nodes, compared to the state-of-the-art. For this purpose an innovative contention based solution was created. By only periodically polling the medium for incoming transmissions, the anchors could remain in a low-power mode for the majority of the time. Only in the presence of a (strobing sequence of a) mobile tag, do they become active in order to perform ranging sequences. The energy efficiency of the anchor nodes comes with some trade-offs. Firstly the battery of the mobile tag nodes drains much more rapidly due to their increased responsibility of waking up the anchors. However, in a large number of use cases the tag nodes have the possibility to be charged regularly, or are attached to large capacity batteries, making the battery drain less of an issue. Secondly, the achievable update rate is lower, compared to other state-of-the-art solutions. In the end, a solution integrator should make a trade-off to which solution best suits their specific use case. If a solution is needed which delivers battery-efficient anchor nodes, with a low to medium high update rate, the UWB-MAC protocol fits those needs.

References

[1] M. G. Wing, A. Eklund, and L. D. Kellogg. *Consumer-grade global positioning system (GPS) accuracy and reliability*. Journal of forestry, 103(4):169–173, 2005.

[2] A. Basiri, E. S. Lohan, T. Moore, A. Winstanley, P. Peltola, C. Hill, P. Amirian, and P. F. e Silva. *Indoor location based services challenges, requirements and usability of current solutions*. Computer Science Review, 24:1–12, 2017.

[3] A. R. J. Ruiz and F. S. Granja. *Comparing ubisense, bespoon, and decawave uwb location systems: Indoor performance analysis*. IEEE Transactions on instrumentation and Measurement, 66(8):2106–2117, 2017.

[4] S. N. A. Ahmed and Y. Zeng. *UWB positioning accuracy and enhancements*. In TENCON 2017-2017 IEEE Region 10 Conference, pages 634–638. IEEE, 2017.

[5] N. Macoir, M. Ridolfi, J. Rossey, I. Moerman, and E. De Poorter. *MAC Protocol for Supporting Multiple Roaming Users in Mult-Cell UWB Localization Networks*. In 2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM), pages 588–599. IEEE, 2018.

[6] R. Dalce, A. Van den Bossche, and T. Val. *Reducing localisation overhead: a ranging protocol and an enhanced algorithm for UWB-based WSNs*. In 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), pages 1–5. IEEE, 2015.

[7] N. Macoir, J. Bauwens, B. Jooris, B. Van Herbruggen, J. Rossey, J. Hoebeke, and E. De Poorter. *Uwb localization with battery-powered wireless backbone for drone-based inventory management*. Sensors, 19(3):467, 2019.

[8] B. Choi, K. La, and S. Lee. *UWB TDOA/TOA measurement system with wireless time synchronization and simultaneous tag and anchor positioning*. In 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), pages 1–6. IEEE, 2018.

[9] A. Costanzo, D. Dardari, J. Aleksandravicius, N. Decarli, M. Del Prete, D. Fabbri, M. Fantuzzi, A. Guerra, D. Masotti, M. Pizzotti, et al. *Energy autonomous UWB localization*. IEEE Journal of Radio Frequency Identification, 1(3):228–244, 2017.

[10] P. Mayer, M. Magno, C. Schnetzler, and L. Benini. *EmbedUWB: Low Power Embedded High-Precision and Low Latency UWB Localization*. In 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), pages 519–523. IEEE, 2019.

[11] Decawave. *The implementation of Two Way Ranging with the DW1000*, 2015.

[12] E. Karapistoli, F. Pavlidou, I. Gragopoulos, and I. Tsetsinas. *An overview of the IEEE 802.15.4a Standard*. IEEE Communications Magazine, 48(1):47–53, 2010.

[13] M. J. Kuhn, M. R. Mahfouz, J. Turnmire, Y. Wang, and A. E. Fathy. *A multi-tag access scheme for indoor UWB localization systems used in medical environments*. In 2011 IEEE Topical Conference on Biomedical Wireless Technologies, Networks, and Sensing Systems, pages 75–78. IEEE, 2011.

[14] J. Tiemann, F. Eckermann, and C. Wietfeld. *Multi-user interference and wireless clock synchronization in TDOA-based UWB localization*. In 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), pages 1–6. IEEE, 2016.

[15] J. Tiemann, Y. Elmasry, L. Koring, and C. Wietfeld. *ATLAS FaST: Fast and simple scheduled TDOA for reliable ultra-wideband localization*. In 2019 International Conference on Robotics and Automation (ICRA), pages 2554–2560. IEEE, 2019.

[16] F. Legrand, I. Bucaille, S. Héthuin, L. De Nardis, G. Gaincola, M. Di Benedetto, L. Blazevic, and P. Rouzet. *UCAN's ultra wide band system: MAC and routing protocols*. In International Workshop on Ultra Wideband Systems, 2003.

[17] M. Ridolfi, S. Van de Velde, H. Steendam, and E. De Poorter. *WiFi ad-hoc mesh network and MAC protocol solution for UWB indoor localization systems*. In 2016 Symposium on Communications and Vehicular Technologies (SCVT), pages 1–6. IEEE, 2016.

[18] M. S. I. M. Zin and M. Hope. *A review of UWB MAC protocols*. In 2010 sixth advanced international conference on telecommunications, pages 526–534. IEEE, 2010.

[19] X. Shen, W. Zhuang, H. Jiang, and J. Cai. *Medium access control in ultra-wideband wireless networks*. IEEE Transactions on Vehicular Technology, 54(5):1663–1677, 2005.

[20] M. Kok, J. D. Hol, and T. B. Schön. *Indoor positioning using ultrawideband and inertial measurements*. IEEE Transactions on Vehicular Technology, 64(4):1293–1303, 2015.

[21] M. Ridolfi, S. Van de Velde, H. Steendam, and E. De Poorter. *Analysis of the scalability of UWB indoor localization solutions for high user densities*. Sensors, 18(6):1875, 2018.

[22] DecaWave. *DWM1001 Datasheet*. (Accessed on 05/25/2020).

[23] B. Van Herbruggen, B. Jooris, J. Rossey, M. Ridolfi, N. Macoir, Q. Van den Brande, S. Lemey, and E. De Poorter. *Wi-PoS: A Low-Cost, Open Source Ultra-Wideband (UWB) Hardware Platform with Long Range Sub-GHz Backbone*. Sensors, 19(7):1548, 2019.

[24] A. Dunkels. *The contikimac radio duty cycling protocol*. 2011.

[25] M. Magno, S. Marinkovic, B. Srbinovski, and E. M. Popovici. *Wake-up radio receiver based power minimization techniques for wireless sensor networks: A review*. Microelectronics Journal, 45(12):1627–1633, 2014.

Part III

Enabling sustainable operation of MAC protocols and coexistence strategies

6

Coexistence between IEEE 802.15.4 and IEEE 802.11 through cross-technology signaling

To ensure the longevity of Internet of Things (IoT) solutions, an important challenge is the coexistence with other wireless solutions occupying the same frequency bands. As the interference, originating from such technologies can be harmful, the general Quality of Service (QoS) of IoT networks can be severely impacted with unwanted effects on metrics such as packet error rate, latency, and battery consumption. Therefore, this chapter offers a coexistence technique in order for IEEE 802.15.4 networks and IEEE 802.11 networks to coexist with each other, using a time slotted Time Division Multiple Access (TDMA) approach. For the IoT devices, a proof of concept implementation was developed using the Time Annotated Instruction Set Computer (TAISC) framework (Chapter 2).

⋆ ⋆ ⋆

J. Bauwens, B. Jooris, P. Ruckebusch, D. Garlisi, J. Szurley, M. Moonen, S. Giannoulis, I. Moerman, E. De Poorter.

Accepted in IEEE INFOCOM (CNERT workshop), May. 2017.

Abstract When different technologies use the same frequency bands in close proximity, the resulting interference typically results in performance degradation. Co-

existence methods exist, but these are often technology specific and requiring technology specific interference detection methods. To remove the root cause of the performance degradation, devices should be able to negotiate medium access even when using different technologies. To this end, this chapter proposes an architecture that allows cross-technology medium access by means of a TDMA scheme. In order to achieve cross-technology synchronization, which is required for the TDMA solution, an energy pattern beacon is transmitted. The use of energy patterns is sufficiently technology agnostic to allow multiple technologies to negotiate between each other. The feasibility of the solution is experimentally demonstrated in a large scale testbed using 50 Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 and IEEE 802.11 devices, demonstrating a successful cross-technology TDMA synchronization rate of over 90%.

6.1 Introduction

Currently a significant number of wireless technologies use the unlicensed 2.4GHz Industrial, Scientific and Medical (ISM) frequency bands for wireless communication. The increasing number of devices including laptops, smartphones, IEEE 802.15.4 nodes, bluetooth devices, etc. using the same limited medium has caused a large number of interference issues. Without medium access negotiation, a throughput loss of up to 30% for IEEE 802.11 and 60% for IEEE 802.15.4 is possible when both technologies coexist in the same environment [1]. Within the same technology interference can easily be minimized by designing Medium Access Control (MAC) protocols that intelligently allocate the medium to individual devices. For cross-technology MAC protocols the choices are limited: agreements on medium usage between different technologies is difficult due to a lack of direct communication possibilities between devices. Several attempts were made at creating a more advanced architecture that allows cross-technology communication. These architectures were not scalable due to overhead, or were limited to only a couple of technologies [2, 3].
In this chapter an architecture is proposed that combines the use of energy detection and a backbone for cross-technology communication between IEEE 802.11 and IEEE 802.15.4 devices which results in fair use of the limited wireless medium. The focus lies on IEEE 802.11 and IEEE 802.15.4, but the concepts and architecture described are also applicable for other technologies.
The contributions of this chapter are as follows:

1. cross-technology synchronization phase based on energy detection;
2. cross-technology communication which enables channel access control in different technologies;
3. multi-platform set-up and evaluation of the solution in a large scale testbed.

The chapter is organized as follows. First section 2 overviews the current state of the art in terms of minimizing cross-technology interference. Afterwards, section 3 gives an overview of how the cross-technology TDMA scheme proposed in this chapter can be used for minimizing cross-technology interference. Next, section 4 explains the cross-technology controller architecture and how it is vital for the usage in the TDMA protocol. This is followed by section 5, which is a general evaluation of the solution. Finally, section 6 concludes the chapter.

6.2 State of the art

This section gives a short overview of related work that aims to improve cross-technology coexistence. The technique that is easiest to implement, and thus most commonly used, is selecting channels to achieve minimal frequency overlap by different technologies. This solution consists of detecting channel loads and choosing the 'best channel' accordingly [4]. Due to the increasing number of devices using the same wireless medium it becomes less feasible to select a channel that has a lower degree of occupancy. Choosing a better channel is also not an obvious choice due to the unpredictable nature of the wireless medium and its channel usage.
Attempts at creating a model of white space in a wireless medium have been made. A node use of this model to access the medium at the most optimal moments [5]. Unfortunately the same problem arises as when changing channel by detecting channel load: medium access is highly unpredictable and can change from one moment to another making the use of a model, in most cases, too unreliable.
Several research groups make use of energy detection mechanisms in their architectures. ESense [6] proposes an alphabet system based on packet length for unidirectional communication between different wireless technologies. Each character in the alphabet is mapped on a specific packet length. Detecting the packet length is done by measuring the time difference between the rising and downward flank of the energy pattern. Subsequently, based on the measured packet duration, the corresponding character is determined. Having a large alphabet raises problems with uniqueness: due to the usage of packet length and not a real 'pattern', a regular data packet might be mistaken for a cross-technology packet. To counter this the cross-technology packet is repeated several times, which results in a wastage of channel time in a medium that already might be saturated. WizSync [7] is a second system that makes use of energy detection but contrary to the concepts described in this chapter, existing IEEE 802.11 beacons are used which are created by default access points. The major advantage of this technology is that it makes use of existing hardware drivers, the drawback that it results in a fixed schedule due to the lack of control over the beacon interval. In a third solution [8] the medium is statically divided between IEEE 802.11 and IEEE 802.15.4 nodes. Within their own time frame the nodes can use a technology specific MAC protocol (e.g. CSMA in IEEE

802.11 and TSCH in IEEE 802.15.4). The IEEE 802.11 nodes start a timer when a IEEE 802.15.4 packet is detected, the nodes will reactivate after this timer has stopped. This solution lacks scalability options as the configuration is static during the whole lifetime of the devices. This chapter improves upon the above concepts by using unique cross-technology energy beacons that allow to reconfigure the devices after deployment.

6.3 Cross-technology interference mitigation

In order to achieve a fair distribution of the medium, a cross-technology MAC protocol is necessary. Two kinds of MAC protocols can be distinguished: contention based and non-contention based. In a contention based approach, devices will sense the medium and if found idle will try and send a packet. If the medium is found occupied a back off will be performed before a new attempt can be made to transmit the packet. The problem with such a mechanism is twofold: the first issue arises from a differing transmission power (typically 0dBm for IEEE 802.15.4, 20dBm for IEEE 802.11). A IEEE 802.11 node that is far enough to not sense the transmissions of a IEEE 802.15.4 sensor node can start sending and blow away all the transmission attempts of the sensor node. A second issue results from the turnaround time of the radio (128µs for IEEE 802.15.4, 5µs for IEEE 802.11), as the sensor node will always draw the shorter straw when trying to compete with a WiFi node [9]. As such contention based approaches mostly unfairly benefit IEEE 802.11 devices.

A non-contention based approach will result in a fairer solution: each node gets it's own share of the medium during which only that node can access it. The most commonly used approach is TDMA where channel access is divided in time slots. To avoid transmission collisions, all nodes should be tightly synchronized to the same reference time which is typically kept by one or more central nodes. TDMA is a proven mechanism that can provide high throughput and is typically very robust. Despite the availability of a TDMA protocol in IEEE 802.11 as well as in IEEE 802.15.4, a cross-technology TDMA is not easy to implement due to a strong dependence on the distribution of timing information. This timing information can only be transmitted to nodes of the same technology and thus a novel approach needs to be taken to achieve synchronization between wireless technologies.

A radio chip might not be able to interpret a beacon from a different technology, but it can detect when the medium is being accessed by means of energy detection. Even more importantly it can detect when a switch between idle and occupied has occurred on a channel. This mechanism in combination with timing information can be used for synchronization purposes. A central node sends beacons which are simple energy patterns containing a sequence of channel access and idle periods

with very specific timings.

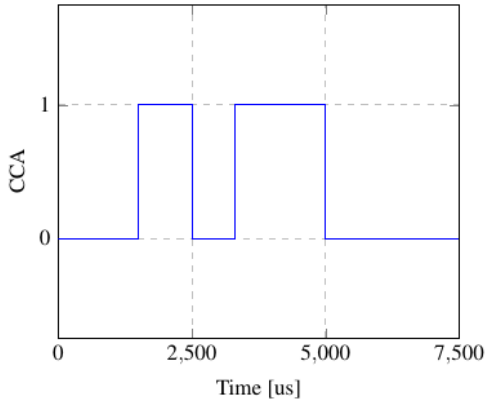


Figure 6.1: Cross-technology beacon transmission

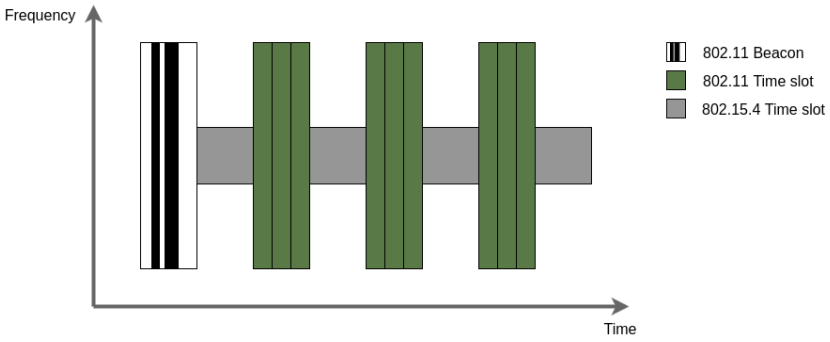


Figure 6.2: Cross-technology synchronization and TDMA schedule. Each IEEE 802.15.4 slot is followed by three IEEE 802.11 slots

Different approaches from coding theory can be used to define custom beacons with different meanings allowing full cross-technology protocol implementations, e.g. to allow negotiation between different devices. Advanced implementations can even include error correction codes [10]. Figure 6.1 shows an example implementation of such a beacon slot used in the TDMA solution. It is of utmost importance that the pattern should be unique enough to not be labeled as interference, hence the up times shouldn't coincide with commonly used packet lengths [6]. For this implementation, following TDMA slot durations were used: a IEEE 802.11 slot corresponds with a slot duration of 2500µs, while the IEEE 802.15.4 slot duration is 7500µs. These durations were chosen by taking into account the maximum packet air time for each technology as well as the slot durations being a multiple

of each other. The decision was made to use a IEEE 802.11 device to generate the beacon for a number of reasons: (1) higher data rate which allows a more fine-grained control of the beacon, (2) higher maximum transmission power (100mW or 20dBm to 1mW or 0dBm), (3) a single IEEE 802.11 channel overlaps with nine bluetooth channels and four IEEE 802.15.4 channels which allows to synchronize multiple channels with a single channel Wi-Fi access point.
To allow other technologies synchronize on the IEEE 802.11 beacon, Clear Channel Assessment (CCA) is used. In general the CCA status is used as an indication if a particular channel is available for transmission or not. The CCA check is based on the Received Signal Strength Indicator (RSSI) and a configurable threshold. An update is generated every four clock cycles (for IEEE 802.15.4 nodes). The MAC protocol will be notified when there has been a change in status of the CCA accompanied by the time stamp on which this change has occurred. With some simple subtractions it can be calculated how long the CCA was high/low and if this corresponds with the beacon generated by the access point. Some margins have to be taken into account: a IEEE 802.11 radio is more precise so there might be some variation on the measured beacon durations.
To minimize data loss a number of improvements can be used, for example it is not always necessary to synchronize on a beacon every superframe. Based on the amount of clock drift on a particular device in combination with the time since the last synchronization a deadline can be derived on which the node has to be synchronized again.

$$time\_since\_sync * clock\_drift = drift < drift\_max$$

If it wasn't possible to distinguish a beacon within the synchronization slot of the superframe, but the deadline hasn't passed yet, execution will be continued as if the synchronization has succeeded. If the deadline has passed the node will inevitably have to wait until the beacon has been detected. With this easy technique it is possible to lower the CCA threshold and by consequence cover a larger area.
A second improvement are intra-technology synchronization nodes that are responsible to synchronize the nodes within their own network. These nodes, in their turn, synchronize on a central node in the same way that was described in previous sections. The synchronization nodes can be put in close proximity to the central node with a high CCA threshold thus minimizing the chance that interference will have an impact on synchronization. Nodes being synchronized within their own technology have the added advantage that they are less prone to interference.

6.4 Cross-technology communication for medium access negotiation

The previous section illustrated how a cross-technology TDMA solution can be used in order to achieve a fair distribution of the medium. In the example scheme (in Figure 6.2) it was shown that the superframe contains a sequence of time slots where each of the IEEE 802.11 slots was followed by a single IEEE 802.15.4 slot. One could argue that the medium access is more or less fair by giving each technology half of the medium access time, but this static configuration might not always reflect the needs of the network. A dynamic slot structure will, in most cases, more optimally make use of the medium. For example a sensor device might not need a slot every superframe: the time allocated might be more suited to give to a node that, at that moment, needs more slots for being able to send all its data over the network. Such nodes need to request extra slots to send data in.
To allow for schedule negotiation the WiSHFUL architecture was used: this architecture supplies UPI's (Unified Programming Interface) for local and remote protocol configuration and monitoring according to the WiSHFUL vision [11]. Using this architecture, nodes can be addressed independent of their originating technology using technology-independent, unified interfaces[1].

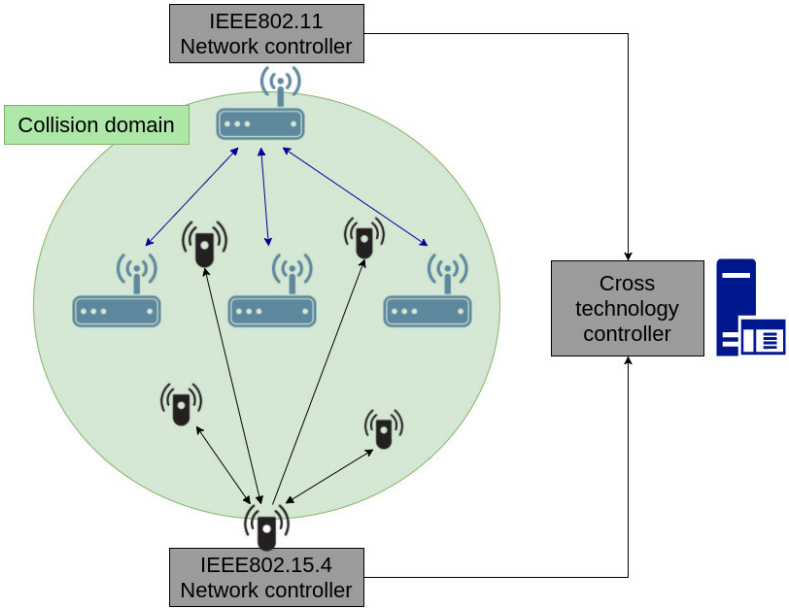


Figure 6.3: Cross-technology controller framework

[1]Cross-technology negotiation could also be implemented using custom beacons with other energy patterns, at the cost of reduced available air-time.

In Figure 6.3 a network of devices is displayed within the same collision domain. These devices are all connected via a WiSHFUL control network. At the top level a cross-technology controller can be found. It communicates with the different technology-specific controllers over a backbone network (to make network-wide changes spanning multiple technologies). In the context of this chapter it is responsible for keeping track of time slot allocation for each technology without knowledge about specific node needs. It accepts requests from the technology specific network controllers for a change in number of slots. A new slot layout is calculated which can subsequently be distributed to all technology specific network controllers.
A technology specific network controller acts as an intermediary between the cross-technology controller and the actual nodes. It keeps track of the nodes inside their network and their requirements and allocates available resources accordingly. These resources can be requested from the cross-technology controller. It should be noted that it is of utmost importance to not only request resources, but allocated resources should be released once they are not needed anymore. The last level of controller logic houses in the nodes itself. Depending on the application needs, the number of necessary resources varies. When allocated resources don't map correctly on the applications needs anymore, the node should ask the network controller to allocate resources differently.

6.5 Evaluation

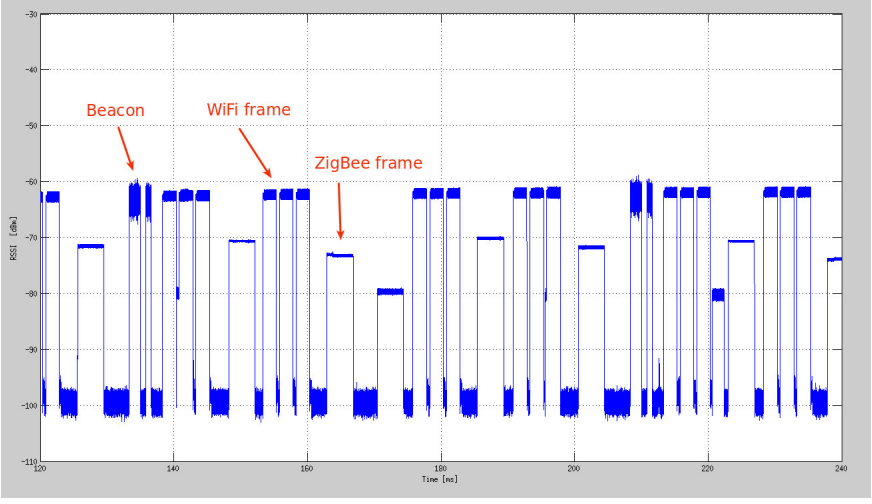


Figure 6.4: Cross-technology synchronization and TMDA schedule

The TDMA solution has been implemented and tested extensively. This section describes how these tests were conducted and which hardware was used.

6.5.1 Used hardware and software

The evaluation was performed in the imec w-iLab.t wireless testbed [12]. It allows for flexible testing of the functionality and performance of wireless networking protocols and systems in a time-effective way, by providing hardware and the means to install and configure firmware and software on (a selection of) nodes, schedule automated experiments, and collect, visualize and process results. This testbed is equipped with a large number of wireless nodes with IEEE 802.11a/b/g/n, IEEE 802.15.4 and IEEE802.15.1 interfaces. The embedded PC's are connected by a wired interface for management purposes. A Linux distribution is available by default (although this can be changed). It is up to the user to define the behavior of the embedded PC's by installing software and/or scripts on the nodes. As such, the embedded PCs can be used for a very broad set of experiments. The testbed allows access to the embedded PC's individually via Secure SHell (SSH) by a web-based testbed interface, so the binaries and scripts can be spread and installed on multiple nodes at once.
Following hardware was used to conduct the experiments:

- IEEE 802.11: PC Engines ALIX;
- IEEE 802.15.4: RM090 sensor node connected to a Zotac;
- Interference: Zotac with 802.11abgn Wifi card

The experiment orchestration was carried out with WiSHFUL local controllers running on the ALIX and Zotac devices, totalling 50 nodes spread throughout the testbed environment which covers $1200m^2$. Each node was loaded with the TDMA solution. The global control program, which controls the different nodes and collects node statistics, can be run on the experimenters own device as well as one of the testbed nodes. For the experiments conducted in this chapter, the global controller was run on a Zotac node inside the testbed. The results were logged from the global controller to a MySQL database. The control programs are publicly available[2].
To evaluate the feasibility and performance of the solution described in this chapter, two hardware platforms were modified. For the IEEE 802.11 nodes the Wireless MAC Processor (WMP)[3] was used [13]. The WMP implementation allows easy generation of the beacon pattern by generating packets of predefined length

[2]https://github.com/wishful-project
[3]https://github.com/ict-flavia/Wireless-MAC-Processor

at specific moments. For the IEEE 802.15.4 nodes, a TDMA has been implemented in TAISC[4], which is an architecture for easy MAC protocol creation [14]. By modifying the existing TDMA protocols, it became possible to detect the energy beacon. It has to be noted that both the WMP, as well as TAISC, are open source. The final TDMA solution is given in Figure 6.2. A beacon slot is followed by the rest of the superframe in which a number of slots is divided among the different nodes in the network independent of which technology the node belongs to.

6.5.2 Performance evaluation

6.5.2.1 Analysis of the TDMA solution

Figure 6.4 shows the cross-technology TDMA protocol that results from the cross-technology negotiation between 50 IEEE 802.15.4 and IEEE 802.11 nodes. The figure is obtained by recording RSSI samples using a Universal Software Radio Peripheral (USRP) node. The USRP was configured to listen on 2440Mhz which corresponds with channel 18 on IEEE 802.15.4 and channel 6 on IEEE 802.11. The beacon, which is being transmitted every 75ms, can clearly be distinguished by the two short consecutive peaks. Following the beacon is the rest of a superframe which consists of a number of IEEE 802.11 and IEEE 802.15.4 slots which are allocated to specific nodes. None of the nodes sends outside of its slot, by consequence the channel access is guaranteed. The resulting solution is a highly synchronized TDMA in which each node gets its own fair share of the medium.

6.5.2.2 Analysis of the synchronization

At first it was attempted to find the optimal CCA threshold to detect the energy beacon. For this purposes all nodes reported the number of synchronization beacons detected within a given time interval and with a given CCA threshold. Figure 6.5 shows the ratio of number of syncs detected relative to the number of transmitted beacons. If the CCA is chosen too low only nodes close to the beacon transmitter will be able to synchronize, if chosen too high the beacon might become indistinguishable from interference.
For the two evaluated technologies the optimal CCA threshold is $-90dBm$ which allows for a synchronization success rate of 90.5%. Since the clock drift is sufficiently low to ensure synchronized operation during multiple superframes, this synchronization rate is sufficient to allow robust long-term operation even when some cross-technology synchronization frames are not detected.
Another factor that impacts the quality of the TDMA solution is the distance to the beacon transmitter. Figure 6.6 shows that the synchronization decays linearly with

[4]https://github.com/WirelessTestbedsAcademy/wishful-contiki-tree

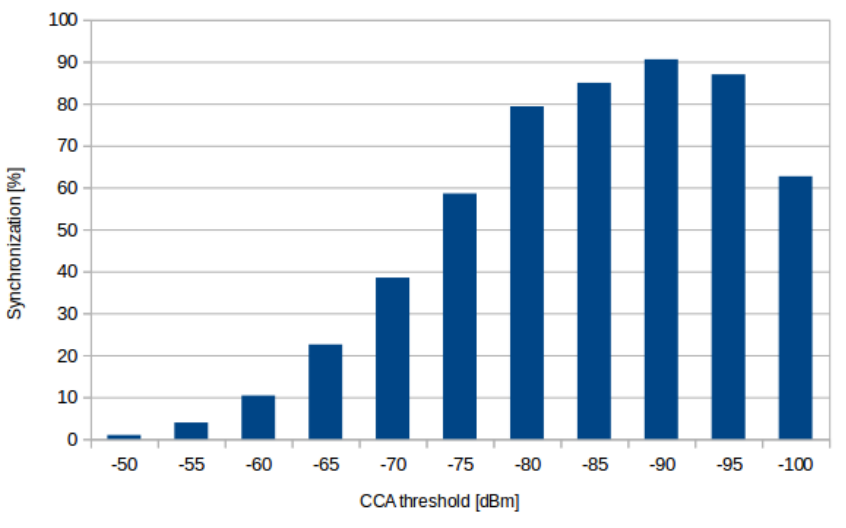


Figure 6.5: Impact of CCA threshold on synchronization

distance to the transmitter. This is to be expected as the RSSI gets lower and thus the chances to get interfered get higher.

6.5.2.3 Analysis of the impact of external (non-controllable) interference

In an ideal world the solution proposed in this chapter would be used on all possible devices and the possibility of external interference will be negated. Since not all devices will simultaneously make the switch to the proposed TDMA solution, external interference will likely still be present. The previous tests were conducted in a semi-shielded testbed facility, and as a result the influence of external interference was minimal. Interference/data originating from nodes outside the network has a big influence in the smooth functioning of the proposed solution. If such interference falls directly inside a beacon slot, nodes might not be able to distinguish the beacon. To successfully receive the energy pattern beacon, the CCA threshold should be set at an optimal value. If chosen too high the beacon will be labeled as noise on nodes that are further from the central node. Therefore, the CCA threshold has a direct impact on the size of the network that can be covered by a single beacon sender. If on the other hand the value is chosen too low, it might pick up energy not originating from nodes inside the network. The pattern can thus become indistinguishable from background noise. Consequently an ideal value does not exist and a trade-off has to be made between network size and the number of times the nodes can safely not receive the synchronization beacon.
To emulate the effects of interference a number of Wi-Fi nodes were selected

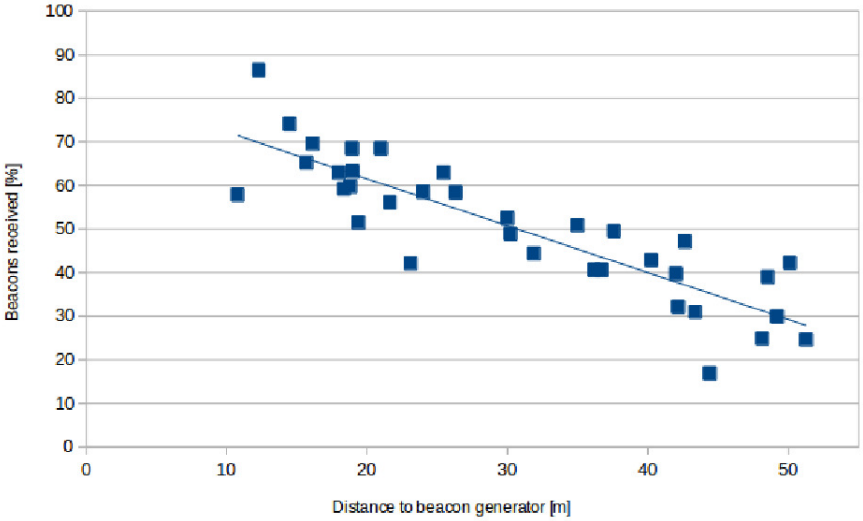


Figure 6.6: Impact of distance on cross-technology synchronization success rate

throughout the environment which use the regular Carrier Sense Multiple Access (CSMA) protocol as described in the IEEE 802.11 standard and have no knowledge about the TDMA used in the other nodes. Figure 6.7 illustrates the effects of interference generated by the IEEE 802.11 nodes. The optimal CCA threshold has shifted to -75dBm. The number of detected beacons has shrunk dramatically, so it can be concluded that non-controllable external interference is a big issue for the described solution.
The resulting work can be demonstrated in real-time, using the WiSHFUL Portable Testbed or the Wilab-t testbed facility. We hereby refer to the demo abstract for IEEE INFOCOM CENRT 2017 [15].

6.6 Conclusion

Cross-technology interference occurs when multiple devices use the same frequency bands without the possibility to successfully negotiate medium access. This chapter focused around mitigating cross-technology interference by means of a simple TDMA mechanism based on an energy pattern beacon. It was shown to be possible to generate such a beacon in a way that other technologies were able to synchronize on it. A shared TDMA scheme was devised with a cross-technology synchronization phase. Initially the TDMA schedule was configured statically. To cope with dynamic application requirements, the solution was extended with a cross-technology management framework which enable channel access control

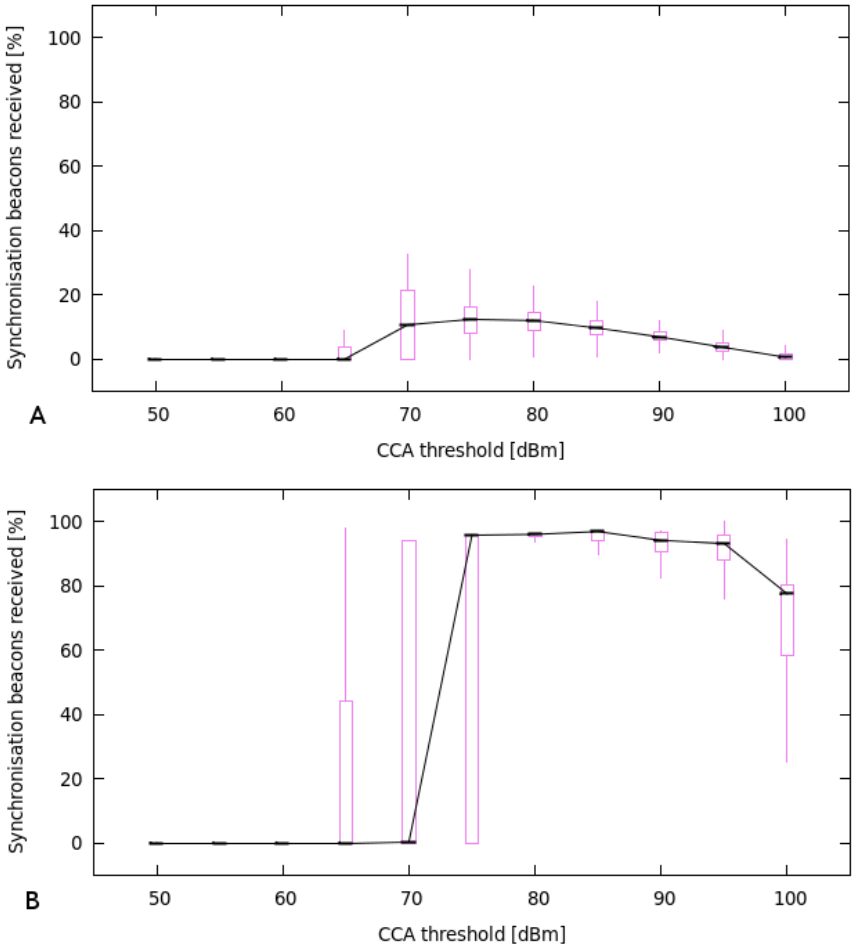


Figure 6.7: CCA threshold impact on synchronization (a) with external interference and (b) without external interference

in different technologies by dynamically allocating time slots to specific technologies/nodes. The cross-technology synchronization was experimentally tested on real hardware in the Wilab-t testbed facility. It was determined that the nodes were able to synchronize more than 90% of the time. In addition, optimal CCA thresshold values were determined both with and without external interference. The resulting solution allows for robust communication when the wireless medium is being shared by multiple technologies.

Acknowledgment

This work was partially supported by project SAMURAI: Software Architecture and Modules for Unified RAdIo control, and European Commission Horizon 2020 Programme under grant agreement no. 645274 (WiSHFUL).

References

[1] R. A. Saeed, S. Khatun, B. M. Al, and M. Khazani. *Performance analysis of ultra-wideband system in presence of IEEE802. 11a and UMTS/WCDMA frequency bands*. In Information and Communication Technology, 2007. ICICT'07. International Conference on, pages 117–122. IEEE, 2007.

[2] R.-C. Wang, R.-S. Chang, and H.-C. Chao. *Internetworking between ZigBee/802.15. 4 and IPv6/802.3 network*. SIGCOMM Data Communication Festival, 2007.

[3] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler. *Trio: enabling sustainable and scalable outdoor wireless sensor network deployments*. In Proceedings of the 5th international conference on Information processing in sensor networks, pages 407–415. ACM, 2006.

[4] M. S. Kang, J. W. Chong, H. Hyun, S. M. Kim, B. H. Jung, and D. K. Sung. *Adaptive interference-aware multi-channel clustering algorithm in a ZigBee network in the presence of WLAN interference*. In Wireless Pervasive Computing, 2007. ISWPC'07. 2nd International Symposium on. IEEE, 2007.

[5] J. Huang, G. Xing, G. Zhou, and R. Zhou. *Beyond co-existence: Exploiting WiFi white space for Zigbee performance assurance*. In Network Protocols (ICNP), 2010 18th IEEE International Conference on, pages 305–314. IEEE, 2010.

[6] K. Chebrolu and A. Dhekne. *Esense: communication through energy sensing*. In Proceedings of the 15th annual international conference on Mobile computing and networking, pages 85–96. ACM, 2009.

[7] T. Hao, R. Zhou, G. Xing, M. W. Mutka, and J. Chen. *Wizsync: Exploiting wi-fi infrastructure for clock synchronization in wireless sensor networks*. IEEE Transactions on mobile computing, 13(6):1379–1392, 2014.

[8] P. De Valck, I. Moerman, D. Croce, F. Giuliano, I. Tinnirello, D. Garlisi, E. De Poorter, and B. Jooris. *Exploiting programmable architectures for WiFi/ZigBee inter-technology cooperation*. EURASIP Journal on Wireless Communications and Networking, 2014(1):212, 2014.

[9] N. C. Tas, C. Sastry, and Z. Song. *IEEE 802.15. 4 throughput analysis under IEEE 802.11 interference*. In Proceedings of the International Symposium on Innovations and Real Time Applications of Distributed Sensor Networks, 2007.

[10] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes*. Elsevier, 1977.

[11] C. Fortuna, P. Ruckebusch, C. Van Praet, I. Moerman, N. Kaminski, L. DaSilva, I. Tinirello, G. Bianchi, F. Gringoli, A. Zubow, et al. *Wireless software and hardware platforms for flexible and unified radio and network control*. In European Conference on Networks and Communications (EUCNC 2015), pages 712–717, 2015.

[12] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester. *The w-iLab. t testbed*. In International Conference on Testbeds and Research Infrastructures, pages 145–154. Springer, 2010.

[13] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli. *Wireless MAC processors: Programming MAC protocols on commodity hardware*. In INFOCOM, 2012 Proceedings IEEE, pages 1269–1277. IEEE, 2012.

[14] B. Jooris, J. Bauwens, P. Ruckebusch, P. De Valck, C. Van Praet, I. Moerman, and E. De Poorter. *TAISC: a cross-platform MAC protocol compiler and execution engine*. Computer Networks, 107:315–326, 2016.

[15] J. Bauwens, B. Jooris, P. Ruckebusch, D. Garlisi, J. Szurley, M. Moonen, S. Giannoulis, I. Moerman, and E. De Poorter. *Demo Abstract: Cross-technology TDMA synchronization using energy pattern beacons*. In Computer Communications Workshops (INFOCOM WKSHPS), 2017 IEEE Conference on, pages 948–949. IEEE, 2017.

7

Over-the-Air Software Updates in the Internet-of-Things: An Overview of Key Principles.

While previous chapters focused on achieving a high Quality of Service (QoS) for current Internet of Things (IoT) networks, this chapter elaborates how to ensure that IoT networks can cope with the demands of the future. To offer a high level of sustainability, a step-by-step approach is proposed to integrate software updates in IoT solutions. This approach allows to integrate novel Medium Access Control (MAC) protocols or novel coexistence techniques into current IoT networks.

⋆ ⋆ ⋆

J. Bauwens, P. Ruckebusch, S. Giannoulis, I. Moerman, and E. De Poorter.

Published in IEEE Communications Magazine, Sept. 2019.

Abstract Due to the fast pace at which the IoT evolves, there is an increasing need to support over-the-air software updates for security updates, bug fixes and software extensions. To this end, multiple over-the-air techniques have been proposed each covering a specific aspect of the update process, such as (partial) code updates, data dissemination and security. However, each technique introduces an overhead, especially in terms of energy consumption, thereby impacting the op-

erational lifetime of the constrained battery powered devices. Up until now, a comprehensive overview describing the different update steps and quantifying the impact of each step is missing in scientific literature, making it hard to assess the overall feasibility of an over-the-air update. To remedy this, our article (i) analyzes which parts of an IoT operating system are most updated after device deployment, (ii) proposes a step-by-step approach to integrate software updates in IoT solutions, and (iii) quantifies the energy cost of each of the involved steps. The results show that besides the obvious dissemination cost, other phases such as security also introduce a significant overhead. For instance, a typical firmware update requires 135.026mJ, of which the main portions are data dissemination (63.11%) and encryption (5.29%). However, when modular updates are used instead, the energy cost (e.g. for a MAC update) is reduced to 26.743mJ (of which 48.69% for data dissemination and 26.47% for encryption).

7.1 Introduction

The IoT refers to the trend to include small, cheap and/or energy efficient wireless radios in everyday objects. IoT solutions are already digitizing an increasing amount of functionalities of modern day society, impacting application areas such as health care, surveillance, agriculture, personal fitness, and home and industry automation. This trend will lead to a further increase in (i) the number of devices per person and (ii) the number of devices per square meter, thereby introducing the need for well designed and maintainable IoT solutions.
However, the specific device limitations, fast technology evolution and the increasing pace at which new devices are rolled out in difficult to reach areas, raise questions concerning the long term sustainability of previously installed IoT networks. For instance, security issues or bugs are often detected post deployment, thereby hindering operational IoT networks. Moreover, already deployed devices cannot take advantage of new features and/or optimizations, or even adapt to new application requirements. A recent industry study showed that the frequency of field updates will significantly increase in the coming years, even with the possibility of monthly software updates [1].
Despite this increasing interest in over-the-air updates, scientific literature discussing the impact on energy consumption is limited. For example, [2] calculates the energy cost for data transmission, but ignores security and reliable dissemination. Similarly, the operational impact of software updates on code versioning are not discussed. This chapter offers a remedy by providing the steps, which make use of state-of-the art techniques, required for enabling over-the-air software updates, while discussing the impact on constrained devices.
In summary, this article contains the following contributions and insights:

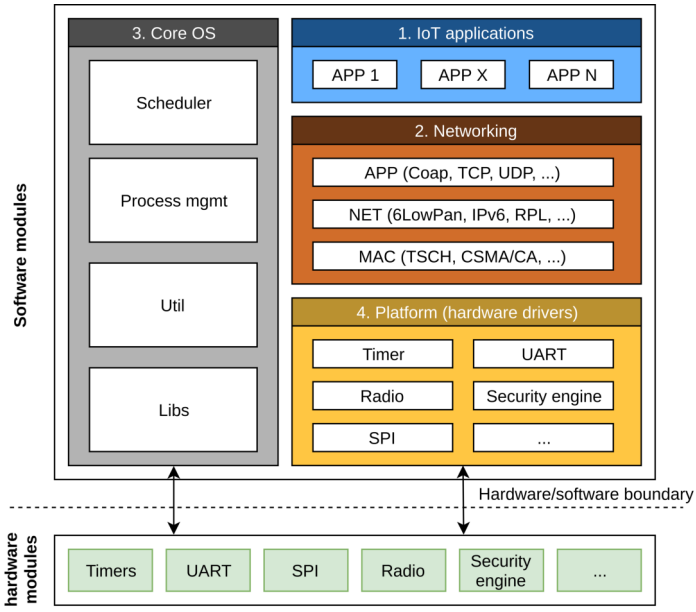
- An analysis concerning the distribution of software development effort in

different parts of widely used IoT operating systems.

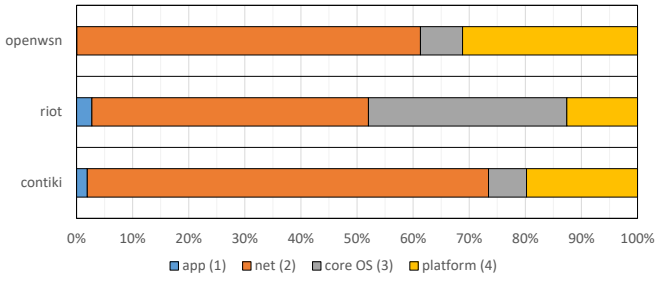
- A comprehensive overview of the key steps in an over-the-air update process.
- A quantification of the energy overhead per deployment phase, showing the relative energy impact.
- A discussion on the impact of updates on operational processes, such as the versioning approach used for software modules.
- A list of future research directions that could enhance the potential of over-the-air updates.

7.2 Analysing update requirements in IoT operating systems

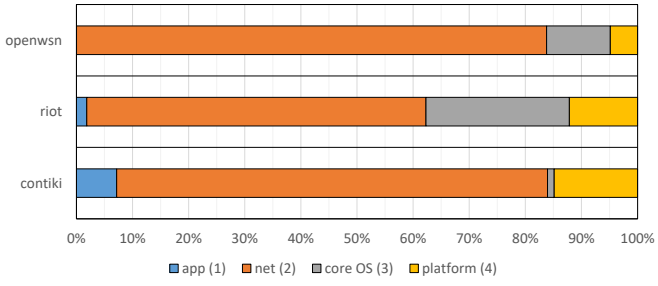
It is important to recognize parts of IoT solutions which evolve quickly and are hence more likely to require software updates. Figure 7.1a depicts the wireless stack of a typical sensor application, containing the software modules and their interaction with the (non-upgradable) hardware modules. The software modules are divided into four blocks: (i) application software, (ii) network protocol stack software, (iii) Operating System (OS) core software, and (iv) platform hardware driver software. Contrary to traditional software systems, the IoT applications are relatively simple (i.e. sense or actuate) and therefore small in size. Most logic resides in the various network protocols enabling end-to-end communication with IoT devices. The memory usage and Git commit history for each block is calculated for different IoT operating systems on a Zolertia Re-Mote, which is a typical constrained IoT device (ARM Cortex-M3 32 MHz clock speed, 512 KB ROM and 32 KB RAM), as depicted on Figure 7.1b and 7.1c (in %). **The network protocol stack comprises a significant portion of the firmware, occupying between 50% and 72%.** The complexity of the network stack is the main reason for the larger code size, which has a direct consequence on the development effort. Moreover, since standards are frequently updated, there is a continuous push to include the latest features in the code base. This in contrast to the core OS and platform code as illustrated in Figure 7.1c, showing the Git commit statistics for two consecutive releases of three IoT operating systems. The statistics include all code lines changed in the software modules required to build a firmware on the Zolertia Re-mote. **Between 60% and 84% of the code changes are related to the network protocol stack.** The rate at which new standards are being proposed seems to be increasing, and therefore the wireless stack will likely not achieve a completely "stable" state in the near future [3].



(a) The typical software components IoT stack (and the hardware interface), divided in application code (1), networking functionality (2), OS core functionality (3), and platform specific code (4).



(b) Relative memory size per component (%).



(c) Relative Git commit statistics of the different components, indicating the percentage of code lines changed between OS software releases (between V2018-01 and V2018.04 for RIoT, RB1.4 and RB1.8 for OpenWSN, and REL3.1 and the august 2018 master branch for Contiki).

Figure 7.1: Git commit statistics and memory usage of three IoT operating systems.

Although an IoT network operator would be mostly interested to enable application updates, Figure 7.1b demonstrates that other code blocks actually comprise a larger portion of the firmware, and are hence at higher risk of containing bugs. Without network protocol updates, it is impossible to guarantee optimal performance during the operational lifetime of the device. A sustainable IoT solution therefore adopts a continuously running development process, taking into account the rapid rate in which technology and business requirements change. In this process, software updates are essential in keeping an IoT solution up to date for the following reasons:

- Protocol and standard version updates, improving efficiency.
- Critical bug fixes and security updates, increasing availability and security.
- New applications, providing additional functionalities;
- Integration with third party IoT systems (hardware or software), extending the scope.
- Adopting new communication standards and protocols, improving performance and interoperability.

However, because the network stack on constrained devices is currently included in the OS, it can only be upgraded by means of a full firmware update, consuming a substantial amount of energy. Given its significance and the rapid change rate, we argue that partial updates of a network stack should also be possible, thereby lowering the energy cost for protocol updates.

7.3 Software update process

This section provides the step-by-step process required to enable over-the-air software updates in a secure and reliable manner, visually represented in Figure 7.2. The procedure is initiated by downloading an updated module from a software repository. First, during the "Software (SW) Module Management" phase, the code is verified offline. This phase includes (i) a compilation step, automatically adding linker metadata to the resulting binary module; (ii) a compatibility analysis step, checking compatibility with the already deployed modules maintained in a binary module repository; (iii) a functional verification step, verifying the module in a simulation or digital twin network mirroring the actual network. On successful completion of the first phase, the "secure software rollout" phase can commence, which includes (i) a security step, encrypting and signing the binary module; (ii) a dissemination step, transferring the binary module to the devices; (iii) an activation step, installing and making operational the binary module. Each of these steps is discussed in more detail in the next sections.

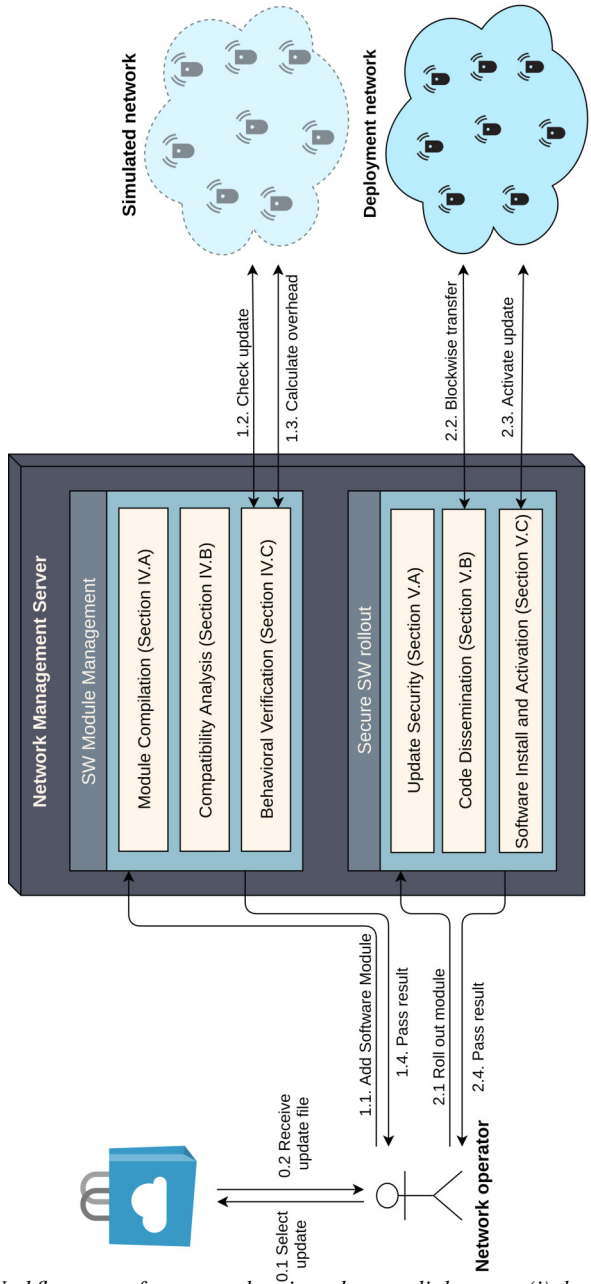


Figure 7.2: Workflow to perform over-the-air updates, split between (i) the management of software modules, and (ii) the secure software rollout.

7.4 Phase 1: software module management

Updating the software on a remote wireless device is error prone: due to unforeseen code interactions the updated code might have adverse effects on performance, or even result in unstable operations. This could necessitate a rollback to a previous stable version. Since a wireless update is an intensive process in terms of medium usage, computational overhead and energy consumption (see Section 7.5), it is important to automatically assess the validity of the update upfront, before actually performing the change and possible wasting valuable resources. For this purpose, a network operator will first pass the software modules to the SW module management services which include three distinct steps, each explained in the remainder of this section: (i) module compilation, (ii) compatibility analysis, and (iii) qualitative predeployment behavioral verification.

7.4.1 Software Module Compilation

Module compilation ensures that the software is transformed from source code into an efficient binary format that can be distributed to a running system. Three different over-the-air software compilation methods are available for constrained devices as discussed in [2]:

- Firmware based approaches replace the entire image. All source code is compiled into a single image and installed on each device. Binary differential patching techniques (e.g. sending only the firmware differences) can be used in order to reduce the image size that needs to be transferred.
- Dynamic linking approaches require a linker on the constrained device to install or update software modules in an active system. The linker relocates code (data) section to the allocated ROM (RAM) memory regions and resolves undefined references to already present modules.
- Prelinking approaches offload the task of the dynamic linker to a more powerful server. An offline linker relocates and resolves undefined references before the modules are disseminated to the devices. This approach requires complete knowledge of the code and data memory location on each device.

The latter two approaches, that is dynamic and pre-linking, are modular and can be further categorized by their binding models [4], defining how code blocks are linked post deployment to the external functionality (functions, shared memory, etc.) provided by other modules:

- A linker that uses a strict binding model statically links code blocks to each other, replacing undefined symbols in one code block with the correct physical address of another code block. Because the physical addresses are hard-

coded in memory, it is practically impossible to relink modules after installation if other modules are updated.

- A linker that uses a loosely coupled binding model employs an indirect function call mechanism and jump tables to redirect function calls between code blocks. By manipulating the jump tables, it is possible to update code blocks in the entire firmware even after installation. This comes at the cost of an increased jump table management complexity and extra memory usage.

The choice of the update method and binding model has a considerable impact on the possible update scenarios (e.g. which code blocks can be updated) and the cost of the update in terms of bandwidth, latency and energy:

- Firmware updates allow to replace the entire code base but requires most bandwidth and, consequently, energy. The latency is also very high, especially because a reboot is required, potentially losing running network state.
- In all cases, pre-linking outperforms dynamic linking because the resulting binary is smaller. This comes at the cost of additional computational complexity and requires that all devices have exactly the same firmware.
- A strict binding model only allows to update/add application level code blocks (i.e. blue part of Figure 7.1a) while a loosely coupled binding model [4] can update all code blocks, for example when including the network protocols.

7.4.2 Compatibility analysis

Because software modules are often developed independently of each other, some versions could prove incompatible with each other, leading to a degraded or broken network. As such, a versioning system needs to verify the compatibility of the different software modules. In case modular software updates are supported (e.g. only a single protocol or application is updated), the compatibility check system should also be applied on a module level.
This validation process can be split up into several sub-processes (see Figure 7.3). First, a compatibility check verifies if the software module can run on the target hardware platform. This is denoted as *'platform compatibility'*. Second, compatibility between the different software modules installed on a single device is checked. This process is referred to as *'inter-module compatibility'*. Last, the *'network compatibility'* ensures that multiple versions of the same software module can co-exist within the same network (e.g. multiple versions of IPv6 on different devices).
On traditional component upgradable software systems, such as Open Services Gateway Initiative (OSGi) [5], only inter-module compatibility is included. This

is not sufficient when applying partial update methods because network layers can be updated separately. For instance, an updated MAC layer could rely on information exchanges (e.g. enhanced beacons in the IEEE-802.15.4 TSCH mode) that are not yet available in older versions of the Physical Layer (PHY) layer, rendering the new MAC version incompatible. To counter this, the central management server keeps track of the software version(s) installed on each device and ensures compatibility. A possible approach utilizes a matrix for keeping track of compatibility, for which an example is shown in Figure 7.3. This compatibility matrix can be updated by performing tests ranging from static code verification, over simulations, to analysis in a real life deployment.

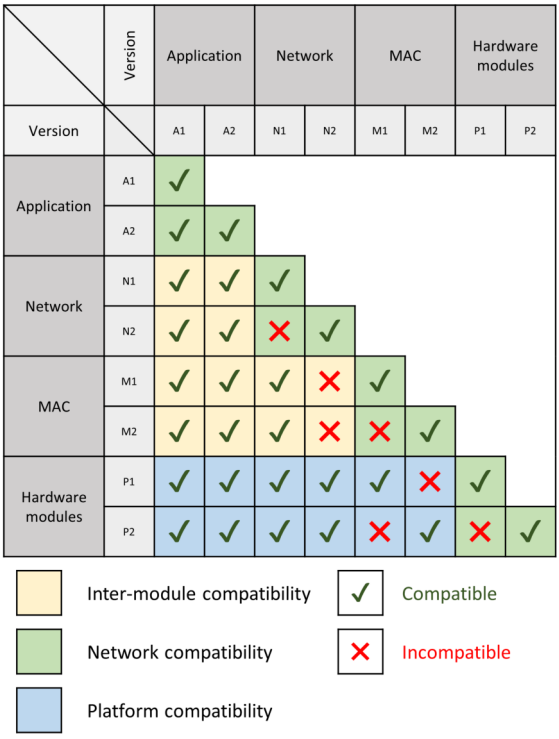


Figure 7.3: An example matrix showing the compatibility between devices and network layers.

7.4.3 Pre-deployment behavioral verification

While the previous verification step primarily focuses on compatibility of software versions, this section describes a methodology to also investigate if the update actually improves the network performance and stability. This is necessary because, contrary to typical software systems, a software update in constrained IoT net-

works reduces the battery lifetime and cannot be easily reverted. A qualitative analysis provides insights in the network operation and verifies the impact on both the node local and network wide QoS after the update. For instance, it checks if the throughput and latency requirements are still fulfilled.
There are a number of ways in which a qualitative analysis can be performed. A sandbox or testbed enables testing the software in a controlled environment on real hardware [6]. While this gives accurate information on a node local level, it is notoriously hard to verify network wide interactions since a sandbox is different from the actual environment. To overcome this, often a network simulator is used (e.g. CupCarbon, Cooja, NS-3, etc. [7]), which provides a network wide view on the overall QoS. However, the results are always an estimation based on a particular channel model.
A simulated virtual environment can be further enhanced using the digital twin concept, used primarily in the context of manufacturing and warehousing [8]. The digital twin mimics the behavior of real physical objects, allowing to test new solutions and business processes in a non-invasive manner. In the context of over-the-air updates, a digital twin mimics a network of interacting IoT devices containing all node types and software combinations of the real life deployment. Moreover, the simulated environment is continuously updated with information from the actual network, improving the simulation model. If the digital twin network behaves as expected after the update, the actual software roll-out can be initiated.

7.5 Phase 2: Secure software rollout

After validating an upcoming software update, the deployment sequence can commence, including: (i) securing and authenticating the data transfer, (ii) disseminating the software update module(s), and (iii) installing the software module(s) and coordinating a simultaneous activation. Note that each step introduces a non-negligible overhead in terms of device memory, network traffic and power consumption. This could drain the batteries, decreasing the operational lifetime of the constrained devices. Therefore a trade-off should be made comparing the (possible) performance gains with the overhead of the update. To gain further insights regarding the overall energy cost of each step, the energy consumption has been measured for the Zolertia Remote device for both a modular as well as a full firmware update, as shown in Figure 7.4. The results were obtained by using the model for data dissemination and installation cost as defined in [2] and combined with extra measurements concerning the various security techniques. The aforementioned results were determined for a single hop topology and do not take into account possible packet loss and/or retransmissions. **Overall, a full firmware update requires about 135.026mJ, while a modular update only requires 26.743mJ.**

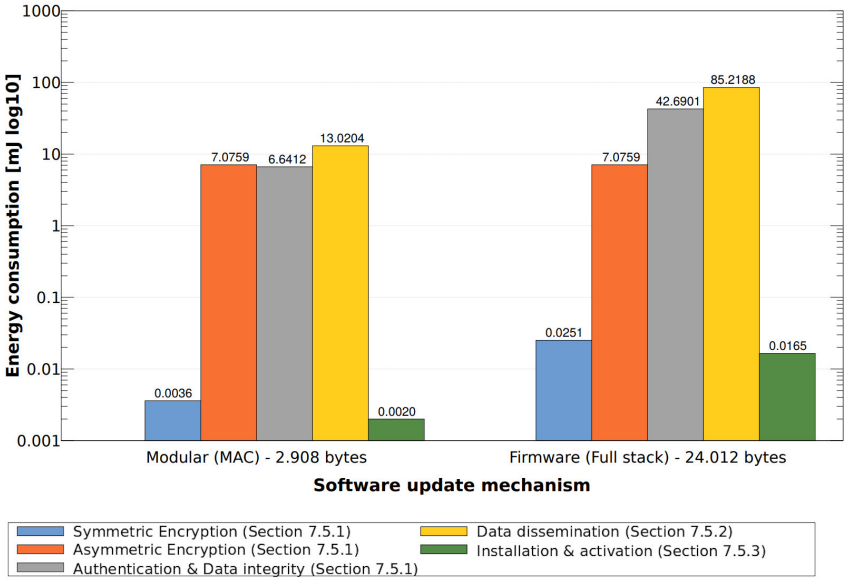


Figure 7.4: Energy consumption of the over-the-air update process for firmware updates (full stack) and partial updates (MAC layer), shown on a logarithmic scale.

7.5.1 Software update security

While wireless updates can be used to fix security vulnerabilities, the update process can also open possible exploits and enable malicious control over the software stack. Several techniques have been proposed in order to secure over-the-air updates, or data dissemination in general. For example, the authors of [9] analyzed how a data transfer can occur while maintaining the four major security aspects: confidentiality, integrity, authentication and availability. The software updates are typically provided by the manufacturer or the local network operator.

Although security is clearly beneficial, the impact on the device functionality as well as network performance should not be underestimated. The remainder of this section will elaborate which security measures are feasible and will quantify the overhead for the constrained end devices. These end nodes typically verify the origin of the update, and decrypt the packet contents.

In order to verify the data integrity and origin of traffic, a Hash-based Message Authentication Code (HMAC) can be appended to each data packet. However, HMAC is commonly used with SHA-256, which requires 32 bytes per packet or 25% of the 127 bytes IEEE 802.15.4 Maximum Transmission Unit (MTU). **Figure 7.4 (third column) shows the measured energy consumption for data integrity and authentication, requiring 6.641 (42.690) mJ for a modular update (full firmware update), which accounts for 24.83% (31.62%) of the total energy consumption.** The HMAC packet overhead is the main reason for the high energy

cost of authentication and data integrity, while the computational overhead for calculating the HMAC only constitutes to $\pm 0.01\,\%$ for both methods. The energy cost can be drastically lowered by appending a single HMAC for the entire update file on the last packet. On the other hand, this disables the possibility to verify data integrity on a per packet basis.
Besides authentication and data integrity, confidentiality is an equally important security aspect. Two flavours of data encryption methods exist: symmetric key encryption (e.g. AES), and asymmetric encryption (e.g. RSA or ECC). In general, symmetric key encryption is faster than the asymmetric counter part. For instance, on the IEEE 802.15.4 CC2538 radio chip it takes 3.4ms to decrypt a full firmware upgrade of 24012 bytes using AES-256, while the same decryption takes 12606.3ms using RSA-1024. On the other hand, symmetric keys offer less security, since the shared key enables unauthorized network access when compromised. Also using multicast traffic in combination with asymmetric encryption is difficult to realise, requiring a key sharing protocol. Therefore it is not advisable to solely rely on either symmetric or asymmetric encryption to guarantee confidentiality.
In order to achieve a high level of security with a decreased energy cost and a lower computational overhead, a combination of both methods can be used (e.g. the Datagram Transport Layer Security (DTLS) standardized protocol [10]). During the initial device bootstrapping, server and clients exchange certificates containing their public key. Per software update a new symmetric session key is generated, which is subsequently asymmetrically encrypted and transmitted via a 'Server Key Exchange'. Any further over-the-air traffic, related to the current software update, is encrypted using this session key. This mechanism offers the advantages of (i) minimizing the consequences of a compromised symmetric encryption key; (ii) enabling multicast during over-the-air update; and (iii) offering a good trade-off between performance and security. **When using DTLS like approaches, encryption still requires a significant amount of energy: 7.080mJ for a modular update and 7.101mJ for a full firmware update. Compared to the total update cost this accounts to 26.47% and 5.29% respectively.**

7.5.2 Code dissemination

Several techniques exist to disseminate an update to wireless devices, as surveyed in [11]. They focus on minimal energy consumption by applying efficient broadcast schemes and try to avoid flooding the network. More recently, with the rise of low power wide area networks, operating in the duty cycle restricted unlicensed sub-GHz bands, novel techniques [12] have been proposed to overcome the duty-cycle restrictions imposed by regulatory bodies such as Federal Communications Commission (FCC) and European Telecommunications Standards Institute (ETSI). Dissemination techniques that rely on unicast transmission schemes

cannot be applied in large scale networks due to these restrictions. For networks with such limitations, coordinated multicast techniques [13] should be applied that can initiate a over-the-air update session on groups of devices [14].
In most cases, software updates exceed the MTU of packets, hence fragmentation is required. A software dissemination protocol must make sure that all devices receive the entire update file. This process does not tolerate any lost packets or bit errors, as this results in corrupted binary code. To overcome this problem with minimal impact on energy and latency, special measures such as block (N)acks and caching on intermediate devices are required. Especially when employing multicast dissemination, retransmissions should also be grouped and multicasted to reduce overhead.
Figure 7.4 shows the energy usage for the constrained wireless end devices during an over-the-air operation. A large portion of the overall energy usage can be accounted to the dissemination, especially when considering that the HMAC message overhead is calculated separately. Also notable are the differences between full firmware and modular updates. **Overall, dissemination costs 13.020 (85.219) mJ for a modular (full firmware) update. Relative to the total energy cost of the update, this constitutes to 48.68% (63.11%).**

7.5.3 Software module installation and activation

After update dissemination, the software must be installed and activated on all devices. The installation procedure is different for each of the update methods (i.e. firmware based, dynamic linking and pre-linking) as discussed earlier in Software Module Compilation. The installation starts as soon as the server notifies that all devices have received and verified the complete update file. The result is reported back to the server after which the activation procedure can be initiated.
Activating software on a group of networked devices should happen simultaneously, especially when concerning network functionality. A failed or delayed activation on one or more devices could introduce protocol inconsistencies, resulting in network connectivity issues. Even worse, if the connection to the update server is broken, it is impossible to fix the issues remotely, making automatic rollback mechanisms a necessity. The devices need to restore the previous software version, either when demanded by the server, or when the connection to the server has been lost.
Figure 7.4 demonstrates that the installation and activation overhead is negligible, as installing only requires copying the relevant sections to Random-Access Memory (RAM) or Read-Only Memory (ROM). Note that when using a dynamic linker on the device, a small portion of CPU overhead is added. **Overall installing and activating an update requires 0.002 (0.017) mJ for a modular (full firmware) update, constituting only $\pm 0.01\,\%$ of the overall energy cost for both methods.**

7.6 Future research directions

When IoT systems become more mature, their ecosystems grow, often attracting third party developers that want to add custom software. This is a natural evolution that helps extending software systems beyond its originally intended scope. This will put forward many challenges that need to be tackled to achieve sustainable IoT networks. (i) The trustworthiness and origin of third party code needs to be verified. (ii) The solutions which are offered by research do not take into account the existence of multiple owners or owner groups, which has a deep impact on the properties of secure software dissemination protocols. (iii) Code isolation techniques should be developed in order to prevent attacks from inserting malicious code. (iv) Recent Software Defined Radio (SDR) platforms also allow partial updates of Field-Programmable Gate Array (FPGA) functionality and thereby the entire network stack, including the physical layer, becomes upgradable. (v) The recent trends towards Software Defined Network (SDN) approaches and visualization, allows networks to inject new "network rules" into the application layer, thereby influencing lower layer protocol behavior. These approaches could be extended by injecting not only rules, but even full software components or new network stacks at the application layer. (vi) Edge/fog-based architecture could be used to to more efficiently disseminate update data to the end devices.

7.7 Conclusions

In the fast growing world of the Internet-of-Things, networks are deployed in increasingly diverse application domains. In order to make IoT solutions truly sustainable, it is necessary to periodically update (parts of) the software post deployment. This article gave a comprehensive overview of the principles, necessary to implement a secure and efficient over-the-air software update mechanism, resulting in a step-by-step approach as summarized in Figure 7.2.

Two distinct phases were identified: (i) the software module management phase, and (ii) the secure software roll-out. The first phase is performed completely offline, in order to minimize the impact for the deployment network. Using the combination of a compatibility matrix and digital twin network it is possible to identify bugs, version incompatibilities or performance issues even before the update is executed. The second phase elaborated on the eventual roll-out of the software modules to the devices, quantifying the energy overhead per step (symmetric/asymmetric encryption, authentication, data integrity, data dissemination, installation, and activation) as can be seen in Figure 7.4. The results show that beside the obvious dissemination cost, the other steps also introduce a significant overhead especially for modular updates (i.e. 51.31% vs. 36.89% for a full firmware update). The use of HMAC for data integrity and authentication, and the use of Elliptic-Curve

Cryptography (ECC) for encryption occupy the main portion of this overhead. To conclude, it is possible to improve the sustainability of IoT solutions and calculate the possible overhead upfront. The results obtained in the second phase allow a network operator to estimate the cost of either a modular or a full firmware update in terms of energy. Thus, a trade-off can be made between this cost and the performance increase.

Acknowledgment

This work was supported by the FWO SBO IDEAL-IoT project (S004017N), the FWO EOS MUSE-WINET project (G0F4918N), and the H2020 ORCA project (732174).

References

[1] H. Guissouma et al. *An Empirical Study on the Current and Future Challenges of Automotive Software Release and Configuration Management*. In Euromicro Conf. on Software Engineering and Advanced Applications, pages 298–305. IEEE, 2018.

[2] P. Ruckebusch et al. *Modelling the energy consumption for over-the-air software updates in LPWAN networks: SigFox, LoRa and IEEE 802.15. 4g*. Internet of Things, 3:104–119, 2018.

[3] J. C. Cano et al. *Evolution of iot: An industry Perspective*. IEEE Internet of Things Magazine, 1(2):12–17, 2018.

[4] P. Ruckebusch, E. D. Poorter, C. Fortuna, and I. Moerman. *Gitar: Generic extension for internet-of-things architectures enabling dynamic updates of network and application modules*. Ad Hoc Networks, 36:127–151, 2016.

[5] P. Brada and J. Bauml. *Automated Versioning in OSGi: A Mechanism for Component Software Consistency Guarantee*. In Euromicro Conf. on Software Engineering and Advanced Applications, pages 428–435, 08 2009. doi:10.1109/SEAA.2009.80.

[6] S. De et al. *Test-enabled architecture for IoT service creation and provisioning*. In The Future Internet Assembly, pages 233–245. Springer, 2013.

[7] M. Chernyshev et al. *Internet of Things: Research, simulators, and testbeds*. Internet of Things Journal, pages 1637–1647, 2017.

[8] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn. *Digital Twin in manufacturing: A categorical literature review and classification*. IFAC-PapersOnLine, 51(11):1016–1022, 2018.

[9] F. Doroodgar, M. A. Razzaque, and I. F. Isnin. *Seluge++: A secure over-the-air programming scheme in wireless sensor networks*. Sensors, 14(3):5004–5040, 2014.

[10] S. L. Keoh, S. S. Kumar, and H. Tschofenig. *Securing the internet of things: A standardization perspective*. IEEE Internet of Things Journal, 1(3):265–275, 2014.

[11] X.-L. Zheng and M. Wan. *A survey on data dissemination in wireless sensor networks*. Journal of Computer Science and Technology, 29(3):470–486, 2014.

[12] L. Cheng et al. *Towards minimum-delay and energy-efficient flooding in low-duty-cycle wireless sensor networks*. Computer Networks, 134:66–77, 2018.

[13] B. Kim and K.-i. Hwang. *Cooperative downlink listening for low-power long-range wide-area network*. Sustainability, 9(4):627, 2017.

[14] J. Toussaint, N. El Rachkidy, and A. Guitton. *Performance analysis of the on-the-air activation in LoRaWAN*. In Information Technology, Electronics and Mobile Communication Conf., pages 1–7. IEEE, 2016.

8
Conclusion

"Be a yardstick of quality. Some people aren't used to an environment where excellence is expected."

–Steve Jobs (1955 - 2011)

Steve Jobs was a man who wanted to achieve a qualitative design of his products and services, even in parts that could be deemed as less important or invisible to customers. Even if a single aspect of the design is not how it should be, the user experience would suffer as a whole. This philosophy led to a highly successful line of products which go far beyond what people expected, alleviating the complete user experience. Within this PhD dissertation we aim to achieve a similar excellence in the world of Internet of Things (IoT), and even wireless networking in general. If one aspect of the wireless experience is not optimal (e.g. frequent connection drops), the user will be less inclined to continue using these wireless devices. Therefore, to make further strides in the adoption of IoT technologies, there is a requirement to achieve excellence in wireless connectivity.

Although IoT was already introduced a number of decades ago, still some challenges remain which stand in the way of achieving seamless connectivity. Most of these challenges arise because the world of IoT is rapidly evolving, resulting in an increasing number of wireless connections and wireless devices with various capabilities. The remainder of this chapter is structured in a number of subsections, each of which summarizes and concludes the work performed per challenge. The last subsection gives some future directions, which go beyond what was achievable within the context of a PhD research.

8.1 Challenge 1: Current MAC protocol architectures only offer limited portability towards different hardware platforms

Current IoT networks consist of a wide variety of platforms, each of which could differ in terms of hardware capabilities and performance. For these to communicate, there is a requirement for them to use the same Medium Access Control (MAC) protocol. Although a variety of MAC protocols optimized for different conditions or network requirements exists, nowadays MAC protocols can not easily be reused on other hardware platforms. As such, there is a need for cross-platform MAC protocols, which can be freely ported towards any platform. Additionally, even if a MAC protocol is available for multiple platforms, MAC timings depend strongly on the underlying hardware. The same MAC protocol implementation could therefore act differently per platform, resulting in unfair/unpredictable behavior when multiple platforms are combined in the same network.
In Chapter 2 we have demonstrated that the Time Annotated Instruction Set Computer (TAISC), a novel framework for hardware independent MAC protocol development and management, is capable of simplifying the development of new MAC protocols. This became possible through the use of a hardware independent language consisting of high level radio commands without losing the real time aspects of protocol execution. The main advantages of TAISC are (i) its intuitive MAC protocol design, (ii) its capabilities for cross-platform compilations, thereby promoting MAC protocol portability, (iii) its capability of over-the-air MAC upgrades, and (iv) accurate radio control due to its inherent time-awareness. By using timing information for all instructions, TAISC MAC protocols can automatically be fine-tuned for optimal energy consumption and spectrum efficiency.
Chapter 3 described in detail the challenges related to multi-platform MAC development. For each challenge a solution was proposed, which can be combined towards an overall methodology for the design of portable MAC protocols. Through the use of these steps it becomes possible to achieve a MAC design which (i) is more developer friendly, by moving the burden of time annotation towards the precompiler, (ii) is more efficient compared to traditional approaches, significantly improving MAC performance and energy efficiency, and (iii) allows efficient reuse and combination of MAC protocols over a wide range of IoT platforms without additional development efforts.

8.2 Challenge 2: Constrained devices can not efficiently use multiple radios on a single platform

When looking at wireless (sensor) networks, different wireless standards enforce the use of a set of network protocols. This set works optimally in given networking conditions, or to accomplish a specific application requirement. However, realistic deployments have dynamic application requirements and changing networking conditions, resulting in difficulties when selecting an optimal wireless standard. By combining radios (and their dedicated network standard) on a single multi-modal device, the advantages of each can be combined, thereby improving the overall wireless behavior. Due to the inherent limitations of IoT devices, multi-modal platforms require an extensive redesign of the wireless stack, especially to the MAC layer. Unfortunately, although a lot of research acknowledges the advantages related to multi-radio platforms, they do not elaborate how to perform the required redesign of the MAC layer.
Chapter 4 gave an overview of possible techniques to achieve multi-modal MAC protocols: (i) sequential activation of the protocols, (ii) the creation of a dedicated MAC protocol controlling all available interfaces, (iii) the splitting of protocols into non-interruptable code blocks, (iv) the splitting of protocol execution in to time slots, assigning each of them to a different protocol, and (v) adding additional Micro Controller Units (MCU's) to the hardware design. In conclusion, it is possible to have multiple MAC protocols running on the same device, however it still requires adaptations to the MAC controller software and/or the hardware design. The offered advantages of multi-modal devices still outweigh the additional design effort, as it enables devices to cope with adaptive networking conditions and changing applications, and it enables seamless connectivity in heterogeneous networks.
Chapter 5 uses the multi-radio MAC concepts to create an innovative contention based localization/ranging solution. Through the use of a dual radio set-up combining a power-hungry Ultra Wide Band (UWB) radio, and a lower power sub-GHz radio, a lot of energy is conserved. By only periodically polling the medium for incoming sub-GHz transmissions, the anchors could remain in a low-power mode for the majority of the time. Only in the presence of a mobile tag, do they become active in order to perform ranging sequences using the UWB interface. Still a couple of trade-offs can be identified: (i) the battery of the mobile tag nodes drains substantially faster due to their increased responsibility of waking up the anchors, and (ii) the achievable update rate is lower, compared to other state-of-the-art solutions. In the end, the solution still delivers extremely battery-efficient anchor nodes, with a low to medium high update rate.

8.3 Challenge 3: Current IoT networks have a limited support for coexistence strategies

When different technologies use the same frequency bands in close proximity, the resulting interference typically results in performance degradation for both technologies. Chapter 6 proposed an interference mitigation technique between Wi-Fi nodes and IoT nodes occupying the 2.4GHz spectrum. By means of a simple Time Division Multiple Access (TDMA) mechanism based on an energy pattern beacon, it was possible to fairly divide the medium. Initially the TDMA schedule was configured statically. To cope with dynamic application requirements, the solution was extended with a cross-technology management framework which enables channel access control in different technologies by dynamically allocating time slots to specific technologies/nodes. It was determined that the nodes were able to synchronize more than 90% of the time. In addition, optimal Clear Channel Assessment (CCA) threshold values were determined both with and without external interference. The resulting solution allows for robust communication when the wireless medium is being shared by multiple technologies.

8.4 Challenge 4: There is a lack of coherent strategies to update devices after deployment

In the fast growing world of IoT, networks are deployed in increasingly diverse application domains. In order to make IoT solutions truly sustainable, it is necessary to periodically update (parts of) the software post deployment. Chapter 7 gave a comprehensive overview of the principles, necessary to implement a secure and efficient over-the-air software update mechanism, resulting in a step-by-step approach. Two distinct phases were identified: (i) the software module management phase, and (ii) the secure software roll-out. The first phase verifies the validity of the software update in terms of compatibility and performance. It is performed completely offline, in order to minimize the impact for the deployment network. The second phase elaborated on the eventual roll-out of the software modules to the devices. It is important to note that beside the obvious dissemination cost, the other steps (especially security) also introduce a significant overhead.

8.5 Future directions

Although some major steps were taken in order to improve the wireless behavior of IoT devices, still some future directions can be identified which were not possible to tackle within the context of the PhD research track.

Towards an MAC development eco-system

Throughout this PhD dissertation, we made use of the TAISC framework which allowed developers to create MAC protocols in an intuitive and easy manner. Although the framework has a lot of strengths, still some caveats remain. Further extensions can still be made towards an even more capable system, which allows a large amount of freedom both for MAC developers, as well as network integrators/managers:

- Even though the strain of creating a new MAC protocol is minimized through the use of the TAISC framework, the complexity is still too big for network integrators who simply want to pick a network stack which performs best in their networking conditions. Therefore - similarly to smartphone application stores - we envision a future in which also MAC protocols and more generally IoT network protocols can be made available.
- For a network manager to select a specific MAC protocol out of the protocol store, it must be known what the performance will be on the platform(s) of choice. This requires the benchmarking of each protocol for each available platform, either via a testbed or in simulation. Since the duration of each TAISC instruction is known, static code analysis could be used to perform an adequate estimate of the MAC performance.
- Within the world of IoT, energy consumption is one of the most important metrics since it dictates how long a device can survive on a single battery charge. Existing MAC programming languages focus on instruction timings, rather than energy consumption. To make an estimate of the energy usage of a protocol, new manual measurements are required on each hardware platform. For this purpose, MAC development frameworks could be extended to include energy consumption models. By combining self-learning techniques with the consumption models, the battery lifetime of a device can be estimated automatically even at compile time.
- Throughout the last years batteries (which are typically expensive, sensitive to environmental changes, and hard to maintain) are starting to get replaced by energy harvesters which collect small amounts of energy out of the environment. Due to the limited energy storage capacity, the devices are restricted in their capabilities: actions which request too much energy should

not be executed. MAC architectures should therefore become energy aware, requiring decision making based on the remaining energy capacity and the energy consumption of an action (which can be calculated using the earlier mentioned consumption models).

Towards interference-less wireless communication

A methodology was proposed to limit the amount of harmful interference between Wi-Fi and IoT nodes, however this is not nearly enough to exclude all interference out of the wireless spectrum. As the world of tomorrow will increasingly rely on wireless connectivity, the number of wireless devices and thus also the wireless spectrum usage will sky-rocket. This puts a strain on current co-existence techniques, who will have difficulties coping with the increased data traffic. Additional effort is required to create novel techniques which can efficiently negotiate spectrum usage between the technologies and devices.
The inclusion of other frequency bands can positively impact the saturated wireless spectrum, however this impact will only be temporary. For example the sub-GHz Industrial, Scientific and Medical (ISM) band is at the moment of writing less crowded than the 2.4GHz bands, resulting in an illusion that the impact of interference is limited. However the interest in sub-GHz is growing fastly, so within a few years these networks can run into similar coexistence issues as 2.4GHz networks or even worse since (i) the available bandwidth is significantly smaller, and (ii) the collision domain of the technologies is significantly larger (a possible range up to 50km). As such, especially for industrial applications, there is a need for sub-GHz interference mitigation solutions that can ensure reliable communication for critical control loops.
On top of the increasing number of devices, we also see an increasing number of wireless technologies (with different RF characteristics) occupying the same wireless medium. This trend will require new techniques which (i) can identify which technology is interfering, and (ii) can restore fair spectrum usage between these technologies (e.g. through the use of collaboration protocols).

Towards fully sustainable IoT solutions

A sustainable IoT solution adopts a continuously running development process, taking into account the rapid rate in which technology and business requirements change. To fulfill the need for sustainable IoT solutions, we already proposed a methodology to update devices post-deployment. Still several additions can be made to the base philosophy:

- The trustworthiness and origin of third party code needs to be verified.
- The solutions which are offered by research do not take into account the

existence of multiple owners or owner groups, which has a deep impact on the properties of secure software dissemination protocols.

- Recent Software Defined Radio (SDR) platforms also allow partial updates of Field-Programmable Gate Array (FPGA) functionality (which is programmable hardware) and thereby the entire network stack, including the physical layer, becomes upgradable. For example your current mobile phone could simply be upgraded to a mobile technology, without the requirement to change your hardware.

The overhead of the update methodology was quantified, and it was shown that performing updates (even partial ones) has a deep impact on battery lifetime of devices. Sustainable IoT solutions could first attempt other less intensive optimization techniques, relying on remote monitoring and reconfiguration of the network stack already available on the device. Through reconfiguration, the solutions provided in Part I and Part II, could even be further improved:

- Within heterogeneous networks, the MAC protocol timings can be updated when a new device type enters, thereby restoring fairness within the network.
- Within multi-radio networks, optimization techniques can be used to select and switch to the best set of protocols.
- The UWB-MAC protocol would profit from dynamic reconfiguration of the radios to further decrease the effectiveness and energy efficiency. For instance, tweaking the Modulation and Coding Scheme (MCS) has an impact on the energy efficiency and achievable distance.

Schematic representation of a heterogeneous network consisting of a variety of single- and multi-radio devices.