



UNIVERSIDAD
DE MÁLAGA

IntegraDos: facilitating the adoption of the Internet of Things through the integration of technologies

PhD Thesis Dissertation in Computer Science

Cristian Martín Fernández

Dpto. Lenguajes y Ciencias de la Computación
University of Málaga

Supervised by:

Manuel Díaz Rodríguez
Bartolomé Rubio Muñoz

November, 2018

UNIVERSIDAD
DE MÁLAGA





UNIVERSIDAD
DE MÁLAGA

AUTOR: Cristian Martín Fernández

 <http://orcid.org/0000-0003-0988-591X>

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional:

<http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode>

Cualquier parte de esta obra se puede reproducir sin autorización

pero con el reconocimiento y atribución de los autores.

No se puede hacer uso comercial de la obra y no se puede alterar, transformar o hacer obras derivadas.

Esta Tesis Doctoral está depositada en el Repositorio Institucional de la Universidad de Málaga (RIUMA): riuma.uma.es

UNIVERSIDAD
DE MÁLAGA



D. Manuel Díaz Rodríguez, Catedrático de Universidad del Departamento de Lenguajes y Ciencias de la Computación de la E.T.S. de Ingeniería Informática de la Universidad de Málaga, y D. Bartolomé Rubio Muñoz, Profesor Titular de Universidad del Departamento de Lenguajes y Ciencias de la Computación de la E.T.S. de Ingeniería Informática de la Universidad de Málaga

Certifican

Que D. Cristian Martín Fernández, Ingeniero en Informática, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo nuestra dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulada

IntegraDos: facilitating the adoption of the Internet of Things through the integration of technologies

Revisado el presente trabajo, estimamos que puede ser presentado al tribunal que ha de juzgarlo, y autorizamos la presentación de esta Tesis Doctoral en la Universidad de Málaga.

Málaga, Noviembre de 2018

Fdo.: D. Manuel Díaz Rodríguez
Catedrático de Universidad del Departamento
de Lenguajes y Ciencias de la Computación.

Fdo.: Bartolomé Rubio Muñoz
Titular de Universidad del Departamento
de Lenguajes y Ciencias de la Computación.

UNIVERSIDAD
DE MÁLAGA





UNIVERSIDAD
DE MÁLAGA

“It’s lack of faith that makes people afraid of meeting challenges, and I believed in myself.”
(Muhammad Ali)



UNIVERSIDAD
DE MÁLAGA

Contents

List of Tables	xi
List of Figures	xv
Acronyms	xvii
1 Introduction	1
1.1 Motivation	3
1.2 Contributions	6
1.3 Outline of this Thesis	8
2 State of the Art and Related Work	11
2.1 Constrained Application Protocol (CoAP)	12
2.2 Run-time deployment and management	15
2.3 Device-decoupled applications and portability in the Internet of Things	18
2.4 Internet of Things and Cloud Computing Integration	22
2.5 Blockchain and Internet of Things Integration	27
3 Run-time Deployment of Physical Resources in the IoT	33
3.1 Problem Statement and Research Goals	34
3.2 Run-time Deployment and Management of CoAP Resources	34
3.2.1 CoAP Middleware	35
3.2.2 Smart Gateway	39
3.2.3 Web UI	41
3.3 Implementation	43
3.4 Evaluation	44
3.4.1 Smart Home case study	44



3.4.2	Memory Footprint	45
3.4.3	Data Transmission	45
3.4.4	Power Consumption	47
3.4.5	Communication Delay	49
3.4.6	Comparison with other proposals	51
4	Appdaptivity: An Internet of Things Device-Decoupled System	53
4.1	Problem Statement and Research Goals.	54
4.2	Appdaptivity System	55
4.2.1	Requirements	55
4.2.2	Approach and Design	55
4.2.3	System Deployments	58
4.2.4	CoAP Network	58
4.2.5	Portability Core (PoCo)	59
	4.2.5.1 Personalised Behaviours	60
	4.2.5.2 PoCo Components	62
4.2.6	User Applications	65
4.3	Implementation	67
4.3.1	PoCo and Interconnection	67
4.3.2	User Applications	68
4.3.3	CoAP Network	69
4.4	Results and Discussion	70
4.4.1	Underlying IoT Infrastructure	70
4.4.2	User Applications	72
4.4.3	Smart Cities: Portability of IoT Services in Different Districts	73
4.4.4	Appdaptivity in HomeLab	76
4.5	Differences between CoAP, the CoAP++ Framework and Appdaptivity	76
5	An Internet of Things and Cloud Computing Integration	79
5.1	Problem Statement and Research Goals	80
5.2	Integration Components	80
5.2.1	Cloud Platforms	81
	5.2.1.1 Batch Processing	81
	5.2.1.2 Distributed Databases	82
	5.2.1.3 Real-time Processing	84

5.2.1.4	Distributed Queues	84
5.2.1.5	Management, Monitoring and Deployment	85
5.2.2	Middleware for IoT	89
5.3	The Lambda-CoAP Architecture: an IoT and Cloud Computing Integration	94
5.3.1	Lambda Architecture	95
5.3.2	Smart Gateway	96
5.3.2.1	Lightweight Virtualisation	97
5.3.2.2	IoT Devices and Virtualisation Management Web UI	97
5.3.2.3	Edge Computing Framework	98
5.3.3	CoAP Middleware	99
5.4	Implementation	99
5.4.1	Lambda Architecture	99
5.4.2	Smart Gateway	100
5.4.3	CoAP Middleware	101
5.5	Evaluation	101
5.5.1	Lambda Architecture	102
5.5.2	Smart Gateway	105
5.5.3	CoAP Middleware	107
5.6	Challenges and Open Research Issues	108
6	On Blockchain and its Integration with the Internet of Things	111
6.1	Problem Statement and Research Goals	112
6.2	Blockchain	112
6.2.1	Challenges	114
6.2.1.1	Storage Capacity and Scalability	114
6.2.1.2	Security: Weaknesses and Threats	115
6.2.1.3	Anonymity and Data Privacy	116
6.2.1.4	Smart Contracts	118
6.2.1.5	Legal issues	119
6.2.1.6	Consensus	120
6.3	IoT and Blockchain Integration	122
6.3.1	Challenges in Blockchain-IoT Integration	127
6.3.1.1	Storage Capacity and Scalability	127
6.3.1.2	Security	128



6.3.1.3	Anonymity and Data Privacy	129
6.3.1.4	Smart contracts	130
6.3.1.5	Legal issues	131
6.3.1.6	Consensus	131
6.4	Blockchain platforms for IoT	132
6.5	Evaluation	134
7	Conclusions and Future Work	139
7.1	Conclusions	140
7.2	Future Work	142
7.3	Publications	142
7.4	Funding	145
A	Resumen	147
A.1	Introducción	147
A.1.1	Constrained Application Protocol (CoAP)	148
A.1.2	Motivación	150
A.1.3	Contribuciones	152
A.2	Despliegue y gestión de recursos físicos CoAP en tiempo de ejecución	154
A.3	Appdaptivity: un sistema para el desarrollo de aplicaciones portables y desacoplados de los dispositivos del IoT	156
A.4	Lambda-CoAP: una Integración de Internet de las Cosas y Cloud Computing	158
A.5	La Integración de Blockchain e Internet de las Cosas	160
A.6	Conclusiones y Trabajos Futuros	161
A.6.1	Conclusiones	161
A.6.2	Trabajos Futuros	163
	Bibliography	164



List of Tables

- 2.1 Integration proposals comparison 23
- 2.2 Blockchain applications 29
- 2.3 IoT-Blockchain applications 30
- 3.1 Smart Gateway API 40
- 5.1 Cloud Platforms comparison I 87
- 5.3 Cloud Platforms comparison II 88
- 5.4 IoT Middleware comparison 92
- 5.6 Lambda Architecture configuration used in the Λ -CoAP architecture 103
- 6.1 Bitcoin nodes and functionality 113
- 6.2 IoT devices to be used as blockchain components 127
- 6.3 Blockchain platforms for creating blockchain applications 133
- 6.4 Blockchain nodes evaluation on Raspberry Pi v3 (Synchronising (s), after synchronisation (as)) 135
- 7.1 ISI-JCR Journals publications that support this thesis 143
- 7.2 International conference publications that support this thesis 144
- 7.3 International workshop publications that support this thesis 144
- 7.4 National conference publications that support this thesis 145
- 7.5 ISI-JCR Journals publications that have been published during this research 145





UNIVERSIDAD
DE MÁLAGA

List of Figures

1.1	Internet of Things connected world	2
1.2	Thesis contributions overview	6
2.1	CoAP messages. (a) Non-confirmable CoAP GET request; (b) Request with piggy-backed response.	13
2.2	CoAP Message Format	14
2.3	Observing a resource in CoAP	15
3.1	Run-time deployment and management of CoAP resources architecture overview	35
3.2	Flow chart with the CoM behaviour on CoAP request	38
3.3	Screenshots of the Web UI	42
3.4	Smart Home case study	45
3.5	Memory footprint with different library types	46
3.6	Power consumption with different behaviours	48
3.7	Energy simulation with several operation rates over a year	48
3.8	Communication delay measures	50
4.1	Appdaptivity architecture.	57
4.2	Possible deployment configurations in Appdaptivity.	59
4.3	System interaction to actuate with a light actuator in a physical location.	63
4.4	PoCo configuration with building and location flows, access control, and different behaviours.	65
4.5	User application screenshots with some defined behaviours defined in the PoCo. (a) A chart, real-time values and a group of lights; (b) Pop-up window for interact with a light group.	66
4.6	Alert message sent from the PoCo to a user application.	68

4.7	Location discovery response time with respect to the CoAP end points discovered. .	71
4.8	Location discovery response time with respect to the PoCo components deployed. .	72
4.9	Response time with different numbers of user applications and the behaviour list of resources.	73
4.10	Smart-city use case translated into a Appdaptivity flow.	74
4.11	Smart-city use case deployment scenario.	76
4.12	RAM and Flash memory usage in the Zolertia RE-Motes and the Zolertia Orion router. (a) Memory average usage in the sketches of the Zolertia RE-Motes; (b) Memory usage in the sketch of the Zolertia Orion Router.	77
5.1	Cloud and IoT integration	81
5.2	The Λ -CoAP architecture: an overview	94
5.3	Lambda Architecture	95
5.4	Smart gateway with lightweight virtualisation, evolution from Chapter 3 to the Λ -CoAP architecture	98
5.5	Lambda architecture management with Apache Ambari	100
5.6	Edge computing framework web UI	101
5.7	Portable solution with the listed sensors and communication. (a) Schematic of the portable solution; (b) Portable solution integrated.	102
5.8	Ingestion throughput in the Lambda architecture: Record Size vs Throughput (MBs)	104
5.9	Ingestion throughput in the Lambda architecture: Record Size vs Throughput (records)	104
5.10	Average latency in the Lambda architecture with multiple message size	105
5.11	Average latency: cloud vs edge	106
5.12	Ingestion throughput (MBs): cloud vs edge	106
5.13	Ingestion throughput (records): cloud vs edge	107
5.14	Power consumption of the CoM in Arduino Mega 2560 and Moteino Mega with different sensors	108
5.15	Memory footprint of the CoM in Arduino Mega 2560 and Moteino Mega	110
6.1	Blockchain IoT interactions	124
A.1	El mundo conectado a través de Internet de las Cosas	148
A.2	Observando un recurso en CoAP	149
A.3	Visión general de las contribuciones de la tesis	152



A.4	Visión general del sistema de gestión y despliegue de recursos físicos en tiempo de ejecución	155
A.5	Arquitectura de Appdaptivity	157
A.6	Una visión general de la arquitectura Λ -CoAP	159



UNIVERSIDAD
DE MÁLAGA

Acronyms

6LoWPAN IPv6 over Low power Wireless Personal Area Networks.

ACL Access Control List.

AMQP Advanced Message Queuing Protocol.

API Application Programming Interface.

CoAP Constrained Application Protocol.

CoM CoAP Middleware.

DBMS Database Management System.

DDS Data Distribution Service.

DHT Distributed Hash-Table.

DoS Denial of Service.

DTLS Datagram Transport Layer Security.

FBA Byzantine agreement Federated Byzantine Agreement.

HTTP Hypertext Transfer Protocol.

IoT Internet of Things.

IPFS Inter Planetary File System.

JSON JavaScript Object Notation.

JVM Java Virtual Machine.

LA Lambda Architecture.

LPWAN Low Power Wide-Area Networks.

mDNS multicast Domain Name System.

MQTT Message Queue Telemetry Transport.

NFC Near Field Communication.

OIC Open Interconnect Consortium.

PBFT Practical Byzantine Fault Tolerance.

PIR Passive Infrared Sensor.

PoB Proof of Burn.

PoCo Portability Core.

PoS Proof of Stake.

PoW Proof of Work.

RDBMS Relational Database Management Systems.

RDD Resilient Distributed Dataset.

REST REpresentational State Transfer.

RPL IPv6 Routing Protocol for Low-Power and Lossy Networks.

SCHC Static Context Header Compression.

SOA Service Oriented Architecture.

SOAP Simple Object Access Protocol.

TCP Transmission Control Protocol.

TSD Time Series Daemon.

UDP User Datagram Protocol.

UI User Interface.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

VSM Virtual Sensor Manager.

WoT Web of Things.

WS Web Service.

WSDL Web Services Description Language.

WSN Wireless Sensor Network.

WWW World Wide Web.

1

Introduction

We are witness to an unprecedented and rapid evolution in electronics, wireless communication and miniaturisation technologies. This has contributed significantly to reducing the production costs of electronic products and embedded devices, thereby increasing the number of them in the Internet era. Nowadays, it is possible to acquire a personal computer for just 9 dollars [1], something unachievable a few years ago. These devices are usually devices connected to the Internet with capabilities to sense and actuate over the physical world, a paradigm also known as the Internet of Things (IoT) [2]. The IoT was probably introduced by Ashton [3] in 1999, and can be defined as a set of interconnected things (devices, tags, sensors, and so on) over the Internet, which have the ability to measure, communicate and act all over the world. The key idea of the IoT is to obtain information about our environment to understand and control and act on it. The IoT has been in a state of continuous growth over the last few years, and according to Cisco [4], the number of connected devices is predicted to reach 500 billion by 2030.

The IoT visualises a totally connected world, where things are able to communicate measured data and interact with each other. This makes a digital representation of the real world possible, through which many smart applications in a variety of industries can be developed. These include: smart homes, wearables, smart cities, healthcare, automotive, environment, smart water, smart

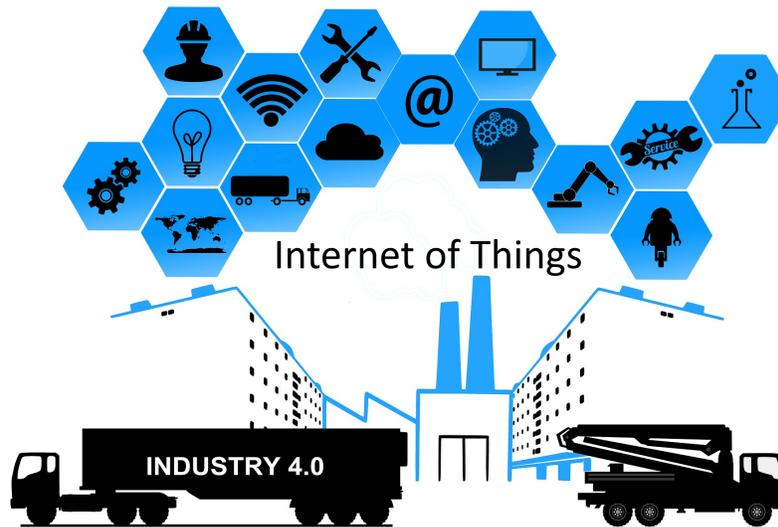


Figure 1.1: Internet of Things connected world

grid, etc. IoT solutions are being deployed in many areas, optimising production and digitising industries. Figure 1.1 shows some domains where the IoT is applied, connecting physical things to the Internet. IoT applications also have very specific characteristics, they generate large volumes of data and require connectivity and power for long periods. This, together with the limitations in memory, computer capacity, networks and limited power supply pose a great number of challenges.

Many IoT devices are designed to work for long periods and are battery-powered as they may be deployed in environments where there may not be mains electricity (e.g., environmental monitoring). To reduce the power consumption and production costs, these devices have limitations in terms of storage, processing and networking. Therefore, applying current standard ways to inter-communicate systems and exchange information such as web services is a non-viable task in most cases. Moreover, one of the main challenges, in the moment of designing an IoT application is the vendor lock-in issues derived from using multiple platforms without a common backbone standard. The Internet Engineering Task Force (IETF) has taken into account the aforementioned problems and has contributed with many standards intended to reduce the gap between resource-constrained devices and powerful ones and connect these devices directly to the Internet. One of its results is the IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [5], which enables the transmission of IPv6 packets on top of IEEE 802.15.4 networks. Routing is also a challenge in constrained networks since they have special features such as limited radio range, large number of nodes and limited resources. An IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL)

[6] was also released by the IETF to solve the routing issues in constrained networks. However, the advances in the media layers are not sufficient, since the heterogeneity still climbs to the upper layers. Taking note of the expansion of the World Wide Web (WWW) and its standard interconnection mechanisms (web services), the IETF designed, through its CoRE (Constrained RESTful Environments) working group, the Constrained Application Protocol (CoAP) [7], which provides a RESTful architecture similar to current web services, nevertheless it has been adapted to resource-constrained devices. CoAP is one of the promising protocols for the IoT and most of the contributions of this thesis are based on CoAP.

The serious constraints with respect to processing, storing and networking also originated integrations with paradigms like cloud computing, known as the Cloud of Things [8], which has attempted to solve such IoT limitations. Cloud computing enables an on-demand access to a pool of configurable resources providing virtually unlimited capabilities in terms of storage and processing power [9], which are the main drawbacks of the IoT. As a result, this paradigm has enabled application scenarios that require intensive computation over large volumes of data, such as health care and critical infrastructure monitoring. These resulting applications benefit from service availability and processing power at the expense of an increase of latency in the data propagation across the backhaul and cloud economic costs. The backhaul data latency cannot meet the requirements of applications that require time-constraints when the network throughput is limited. In addition, the next generation of applications will generate large amounts of data that would result in a bottleneck for mobile networks in cloud-based computing applications. For instance, the next generation of connected cars are expected to generate 1GB of data per second [10]. This will be difficult for current mobile networks to manage, even with the advances of the next generation networks (5G). Recent paradigms such as fog, edge and social dispersed computing [11] aim to reduce both latency and bandwidth in IoT cloud communication, moving the computation to what is available near to the generating sources. These paradigms will place the computation as close as possible to where it is being generated, thereby reducing the latency, a requirement for (near) real-time applications. Social dispersed computing goes beyond and aims at the participation of end users to improve the global benefit of the system.

1.1 Motivation

Despite the improvements in the latest protocols and paradigms for the IoT, the adoption of the IoT still poses a great number of challenges to be overcome before becoming a reality. Among these challenges can be identified the volatility and mobility of its applications, the limitations of its

devices, and the need for fast processing and reliable data. During the course of this thesis we have found four motivations that have inspired our research to facilitate the adoption of the IoT:

On the one hand, IoT applications are still designed to be coupled to final devices, which means that applications do not adapt to the continuous changes in the underlying infrastructure. In the case of inclusion of new devices in these applications or even their disconnection due to battery consumption, it is still necessary to reconfigure these applications and devices to adapt to these changes and to be part of the system. In the IoT where the number of devices grows in an unprecedented way and devices are very volatile, these limitations drag out its expansion. For instance, in environmental monitoring, device-decoupled applications could display over-the-fly measurements from sensors deployed based on the users' needs. Moreover, IoT applications are personalised, i.e., the underlying IoT infrastructure can be designed to work with certain end users. This greatly increases the complexity in the application development. For instance, imagine a personalised application for more than one hundred end users. The development for all the personalised logic can represent a higher development effort than the whole architecture itself. Last but not least, end users can be part of different IoT applications in separate scenarios. In this case, it would be desirable to have a control over these applications with the minimum configuration and using a common interface.

Apart from software portability, IoT solutions comprise many hardware components that can be exchanged or required in different environments depending on the user's needs. IoT devices can be divided into two categories in most IoT scenarios. On the one hand, the devices can constitute a black box with specific sensors which complicates their configuration, e.g., wearable products. Other IoT devices can be composed through configurable microcontrollers, enabling customisable environments to be designed. These IoT devices are closer to the vision of the IoT, as they are plug&play devices. However, the necessary tools and knowledge for programming and configuring microcontrollers are not accessible to everyone. In addition, this process should be done in run-time in order to avoid the service interruption of devices. This could also lead to power saving (critical in IoT devices), deactivating the unnecessary hardware components when required. Many application scenarios use a lot of sensors and actuators for their operation, such as smart cities and environmental applications, so this challenge could have a positive affect on many IoT scenarios.

On the other hand, the limitations of the IoT devices make it difficult to apply paradigms that convert raw data into useful information, as they require large capabilities in terms of processing and storage, such as big data. Paradigms like cloud computing have been integrated with the IoT to overcome its main limitations and provide these much needed capabilities. Even though these integrations can overcome such limitations, the current need to extract knowledge from the data in (near) real-time implies having to think beyond. Applications in critical environments, such as

structural health monitoring [12], not only require large computation capabilities but also a low latency response, otherwise the information could stop being useful. These scenarios, where the data propagation latency over the backhaul cannot meet requirements, lead to cloud computing, even though it may be unsuitable due to high latency. Current paradigms, such as edge and fog computing, should, therefore be considered to carry out the integration of the IoT with cloud computing and meet the low latency of these scenarios, without forgetting the limitations of the IoT devices.

Finally, the huge expansion of the IoT has to be supported by standard mechanisms and protocols as seen in order to reduce the existing heterogeneity in the field. This heterogeneity leads to vertical silos and reduces the adoption of the IoT. However, aside from the heterogeneity and integration challenges present in the IoT, the trustworthiness of its data is also an important issue to bear in mind. Nowadays, we trust in the information of financial entities and the government among others, but can we be sure that the information provided by them and by other external entities, such as IoT companies, has not been tampered with/altered/falsified in any way? This is a difficult question to answer in centralised architectures. Untrusted entities can alter information according to their own interests, so the information they provide might not be completely reliable. This brings about the need to verify that the information has never been modified. Although the IoT can facilitate the digitisation of the information itself, the reliability of such information is still a key challenge. In this sense, a new technology that was born as the first decentralised cryptocurrency has the potential to offer a solution to the data reliability problem: Blockchain. The integration with the IoT would perfectly complement both technologies. Application domains that manage sensitive data, such as health care and police surveillance, could leverage this integration to both acquire more information about their processes and save it in a reliable way.

These challenges have motivated the work of this thesis. Specifically, this thesis addresses the following research questions:

- (i) How can the management and deployment of physical resources in the IoT be facilitated?
- (ii) How can we intuitively define portable, adaptable, personalised and device-decoupled IoT applications?
- (iii) How can the IoT be integrated with cloud computing taking into account the vast amount of data generated in the IoT, low latency and the limitations of its devices?
- (iv) How can the IoT be integrated with Blockchain and what are the potential benefits and challenges of that integration?

1.2 Contributions

This thesis aims to investigate the aforementioned challenges. The main goal of this thesis (IntegraDos) is to facilitate the adoption of the IoT through the research challenges and integrations identified. This goal comprises four individual contributions. Figure 1.2 shows an overview of the contributions of this thesis. On the one hand, it has been addressed how the management of sensors and actuators in physical devices can be facilitated without programming and rebooting their systems (1). On the other hand, a system to program portable, personalised, adapted to changing contexts and intuitive IoT applications has been defined (2). Then, the integration components for an IoT and cloud computing integration have been analysed, resulting in the λ -CoAP architecture (3). Lastly, the challenges for an IoT and Blockchain integration have been analysed and the possibilities of IoT devices to incorporate blockchain components have been evaluated (4).

Specifically, the following contributions are provided in this thesis to respond to the thesis' research questions:

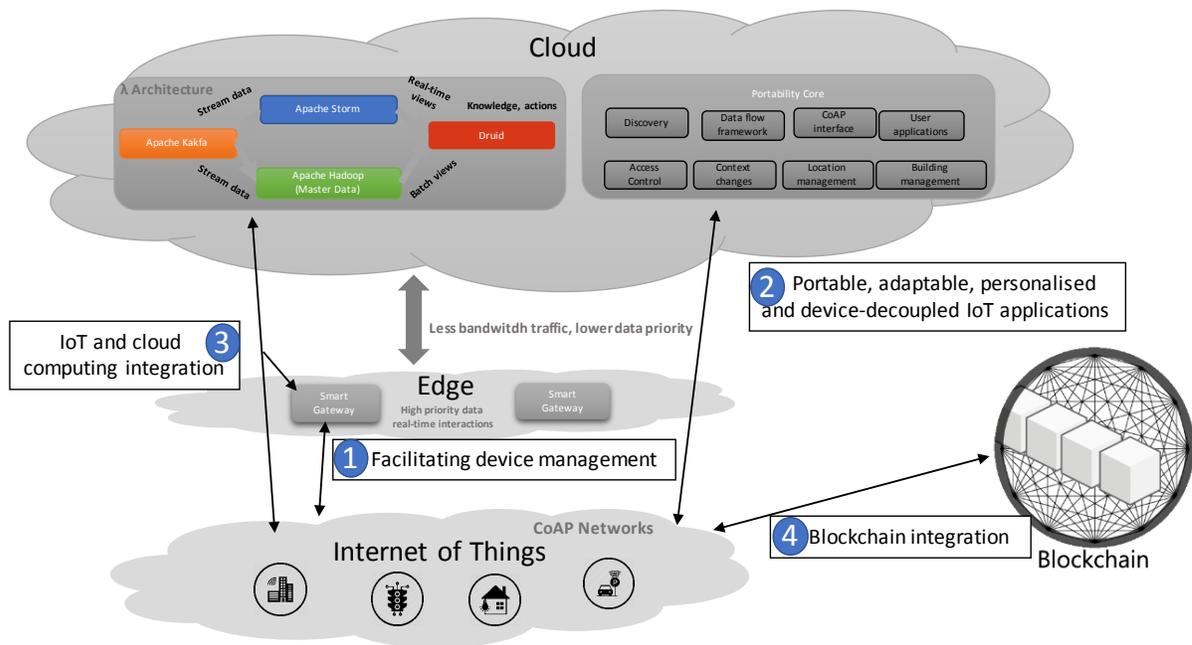


Figure 1.2: Thesis contributions overview

- (i) A run-time deployment and management system of physical resources that bridges the gap between end users and customisable environments. With this system, end users can incorporate new physical components in their systems without having to access and program the micro-

controller board. Application domains with a high density of hardware components such as smart cities will benefit from this system in the management of their components. The limitations of the devices to support libraries have been taken into account with the definition of an objective middleware for a specific environment.

- (ii) A framework that enables the development of portable device-decoupled applications that can be adapted to changing contexts called Appdaptivity. Through Appdaptivity, application developers can intuitively create portable and personalised applications, disengaging from the underlying physical infrastructure. This enables the development of applications that adapt to the continuous changes in the underlying infrastructure. Application developers just need to concentrate on the application logic, rather than the low-level details of the underlying physical infrastructure, which is managed by Appdaptivity. The portability of applications enables their mobility between different physical infrastructures with a minimal configuration. This contribution meets the requirements of the development of IoT applications in general, specifically those that have dynamic behaviours and plug&play components.
- (iii) A comprehensive study of the integration components for a cloud computing and IoT integration, including existing integrations and proposals. The study comprises from low level components for the IoT, such as IoT middleware, up to multidisciplinary cloud computing platforms. The analysis leads to the λ -CoAP architecture, which proposes an integration of cloud computing and the IoT. The purpose of the λ -CoAP architecture is to query, process and analyse large amounts of IoT data with low latency. This low latency is achieved in part thanks to an edge layer that has been incorporated to reduce both the latency and the bandwidth in cloud-IoT communications. Application domains that require large-scale data processing with a low latency response, such as structural health monitoring, could make use of this architecture to overcome their limitations problems.
- (iv) An exhaustive study on the challenges and open issues in the new disrupting technology for reliability, Blockchain, and its integration with the IoT. The study also analyses current blockchain platforms and applications, and provides an evaluation and comparison of the performance of different blockchain components in the IoT. Finally, this contribution aims to analyse the challenges and open issues for an integration between the IoT and Blockchain to ensure data reliability in IoT applications in the near future.

Therefore, the overall challenge of this thesis, to facilitate the adoption of the IoT, has been addressed through these aforementioned contributions. These contributions are not strictly related

with each other but together they contribute to addressing the overall challenge of this thesis as we will see below. First, the run-time and deployment system and the λ -CoAP architecture contributions have been developed as different iterations of the same project and are perfectly integrated. This can open up more avenues, such as applications that require large processing of data and hardware component management. For instance, an application use case can process the data received from the IoT and decide which physical components are not needed, and deactivate them thereby saving power in IoT devices. On the other hand, Appdaptivity was developed during a research stay with the IDLab research group of the University of Ghent. This project contributes to the overall challenge established and is part of the IDLab research group. Thus, this project has not initially been integrated with the previous contributions, however it could be achieved thanks to the component-based architecture detailed in Chapter 4. This integration should consider the distribution of the application logic between the entire λ -CoAP architecture, enabling the development of time-sensitive, portable, device-decoupled and compute-intensive applications, which would represent a great research challenge. Finally, the Blockchain and IoT integration contribution is an exhaustive study on the open issues and challenges in a cutting-edge and timely topic of research in order to be addressed and integrated with the rest of contributions in a near future.

1.3 Outline of this Thesis

The remaining chapters are structured as follows:

Chapter 2. State of the Art and Related Work.

In this chapter, related work with regard to the contributions of this thesis are presented. Firstly, an introduction to CoAP is given. Then, the related work is categorised into the four main contributions of this thesis: run-time deployment and management, device-decoupled applications and portability in the IoT, IoT and cloud computing integration and finally Blockchain and IoT integration.

Chapter 3. Run-time Deployment of Physical Resources in the Internet of Things.

This chapter presents the run-time and management system for incorporating new sensors or actuators in customised environments. This system facilitates the management of IoT devices, enabling libraries to be installed remotely. The evaluation presented in this chapter has been done in one of the main case studies in the IoT, the smart home.

Chapter 4. Appdaptivity: An Internet of Things Device-Decoupled System for Portable Applications in Changing Contexts.

Appdaptivity, a system that enables the development of portable, adapted to changing contexts, personalised and device-decoupled applications is presented in this chapter. The chapter includes some examples that show how to program these applications in Appdaptivity. The evaluation is based on a smart city use case, where 6LoWPAN devices along sensors and actuators are deployed. Lastly, the differences between CoAP, the CoAP++ Framework and Appdaptivity are analysed.

Chapter 5. An Internet of Things and Cloud Computing Integration.

This chapter begins with an analysis of the integration components in the integration of the IoT and cloud computing. This analysis establishes the guidelines for the definition of the integration. Then, the λ -CoAP architecture is presented as an integration of the IoT and cloud computing. This architecture enables the processing of large amounts of data with low latency. Finally, an evaluation of the architecture and challenges and open research issues is given.

Chapter 6. On Blockchain and its Integration with the Internet of Things

The current challenges and open issues in Blockchain, and its integration with the IoT are analysed in this chapter. It then continues with the comparison of current blockchain platforms and existing integrations. Lastly, an evaluation of blockchain components in IoT devices examines the potential of this technology in the IoT.

Chapter 7. Conclusions and Future Work.

Conclusions and future work with regard to the contributions of this thesis are presented in this chapter. The funding and the publications produced in the course of this thesis are detailed.

Appendix A. Resumen.

This appendix summarises this thesis in Spanish.

2

State of the Art and Related Work

This chapter presents the state of the art and related work with regard to the contributions of this thesis. The chapter is structured as follows. An introduction to CoAP, the backbone IoT protocol used in this thesis, is given in Section 2.1. Section 2.2 presents the state of the art and related work on run-time deployment and management. Section 2.3 surveys work on device-decoupled applications and portability in the IoT. Then, Section 2.4 discusses related work on the integration of the IoT and cloud computing. Lastly, integrations of IoT and Blockchain are presented in Section 2.5.

This chapter surveys the related work with regard to the four contributions and the backbone IoT protocol of this thesis. However, some related work to the contributions are extended in their respective chapters. For instance, several integrations of the IoT and cloud computing are presented in this chapter, whereas the analysis of integration components for an IoT and cloud computing integration are detailed in Chapter 5. Equally, integrations of the IoT and Blockchain are presented in Section 2.5, whereas the state of the art in Blockchain platforms for the IoT, and the challenges on Blockchain and its integration with the IoT are presented in Chapter 6.

This has resulted principally in two comprehensive blocks of related works with regard to the integration of the IoT with cloud computing and Blockchain, supported by two high impact publications, [2] and [13] respectively. The integration of the IoT and cloud computing, recently awarded with the Journal of Network and Computer Applications 2017 Best Survey Paper Award, was a cutting-edge contribution as was the contribution of Botta et al., [14], and they were published almost at the same time. Both contributions address integration, but with different approaches. The contribution of this thesis focuses on integration components, whereas Botta et al., centre more on the motivations toward the integration and applications. For an extensive survey on applications, protocols, and technology enablers for the IoT, please refer to [15]. The contribution of the integration between the IoT and Blockchain discusses, in more depth, in the challenges for Blockchain and its integration with the IoT, in addition to evaluating the use of Blockchain platforms in IoT devices. Related work on this topic such as [16] and [17] follow different approaches to address this integration, paying more attention to application use cases. Lastly, to solve the ambiguity between computing paradigms and find core-enablers and challenges in a cutting-edge paradigm, such as social dispersed computing, please refer to [11].

2.1 Constrained Application Protocol (CoAP)

Heterogeneity, limitations and vendor lock-in issues have been present for a while in resource-constrained devices. As mentioned above, the IETF has contributed with many standards to connect these devices to the Internet and reduce the gap between them and powerful ones. Notable, among these standards are the 6LoWPAN, RPL and the Constrained Application Protocol (CoAP).

CoAP provides a RESTful architecture similar to current web services, nevertheless it has been adapted to resource-constrained devices. The most notable feature with respect to current RESTful web services is the abolition of the high-consumption protocols involved, such as HTTP and TCP. Handshaking, packet reordering and header characteristics of these protocols suppose a high barrier to devices with limited capabilities and presence in constrained networks. CoAP has adopted UDP

as the transport layer, thereby reducing the overhead and requirements of TCP communications. The adoption of UDP brings IP multicast which can improve the control and reduce the number of communications in constrained networks. Due its UDP nature, CoAP provides two types of communications: confirmable and non-confirmable messages. Therefore, CoAP offers the strengths of confirmable messages in TCP while simultaneously providing lightweight communications for environments where confirmations are not necessary (e.g., stream data monitoring). To reduce the number of communications, confirmation responses can also be piggybacked. With a client-server model, Figure 2.1 shows two examples of CoAP communications. Concretely, a client sends a GET request with a non-confirmable message to a CoAP server, and the server returns the temperature status (Figure 2.1a); and the same communication with a confirmable message and a piggybacked response (Figure 2.1b).

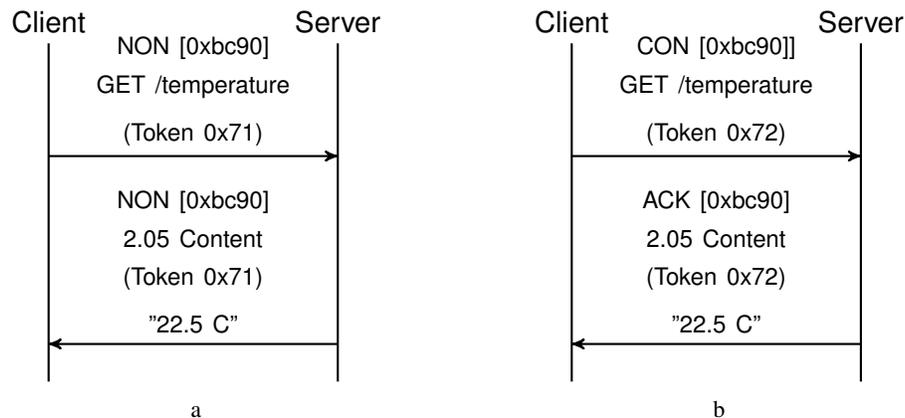


Figure 2.1: CoAP messages. **(a)** Non-confirmable CoAP GET request; **(b)** Request with piggybacked response.

The CoAP header is characterised by its short fixed length (only 4 bytes) that may be followed by a token, options and a payload. The token is used to match requests and responses. Messages in a CoAP communication possess the same token. Other options can be included in CoAP messages and include additional information of messages, such as the content format of responses and the URI (Uniform Resource Identifier) to identify a resource. CoAP options can be seen in a similar format to HTTP metadata. If there is any non-zero length payload, one-byte payload marker (0xFF) is added which indicates the start of the payload and the end of the options. A structure of the CoAP header and messages is presented in Figure 2.2. The rest of the fields are defined as follows:

- Version: unsigned integer which indicates the CoAP version.

- Type: indicates the type of the message: confirmable (0), non-confirmable (1), acknowledgement (2) and reset (3).
- Token Length (TKL): indicates the length of the token (0-8 bytes) if any.
- Code: indicates the message operation type (e.g, request, response and client/server error). The code is split into a 3-bit class which indicates the class of the operation (e.g, request and response) and a 5-bit detail which indicates extra information of the operation (e.g, GET request, success response).
- Message ID: 8 bit unsigned integer used to detect message duplication and match requests and acknowledgements.

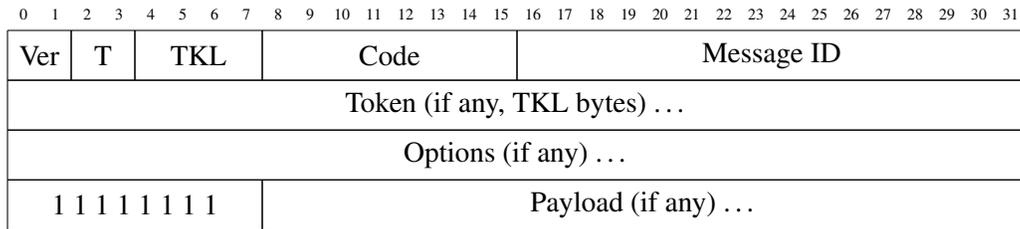


Figure 2.2: CoAP Message Format

Like RESTful web services, CoAP offers operations and objects through resources, which can be accessed with HTTP-style methods (GET, PUT, POST and DELETE). Accessing sensors and actuators can be defined in resources, thereby enabling an HTTP-style access to them. For instance, in Figure 2.1 a CoAP server contains a temperature sensor resource which is accessed through the ‘/temperature’ URI by CoAP clients. In monitoring systems, it is important to have an up-to-date representation of the interested resources. However, continuous resource polling can drastically affect the devices’ performance since communication is one of the most power-consuming tasks on these devices. CoAP supports proxying with caching capabilities, which can reduce the number of communications with final devices. Consider an HTTP-CoAP proxy which translates HTTP requests into CoAP ones. In the latter scenario, HTTP clients can access CoAP resources in the same way as the WWW (World Wide Web), enabling the known paradigm of the WoT (Web of Things). However, it does not make an efficient use of these resources. IoT devices have still limitations that they have to address, like sleeping cycles which complicate access by clients. Furthermore, most monitoring systems are not interested in receiving monitoring data, but they are interested in data changes. An extension of CoAP, known as resource observation [18] introduces a mechanism to optimise the interactions with CoAP devices. Resource observation enables a new asynchronous

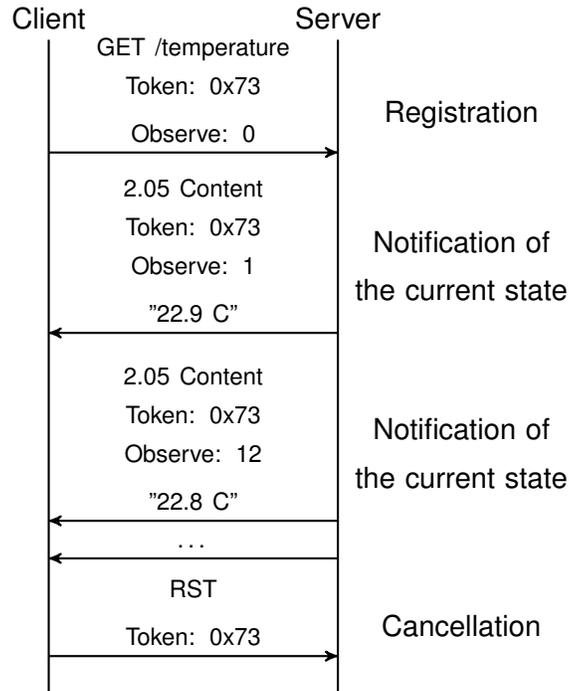


Figure 2.3: Observing a resource in CoAP

communication model in CoAP where clients can subscribe to resources and receive asynchronous notifications. The observe operation can be configured with a max age option to indicate an age up to resource statuses are acceptable. Resource notifications are also sent when its status changes. Therefore, the resource observation avoids continuous client polling in resources and final devices can send data when they are available combining it with other tasks like the sleeping mode. On the other hand, clients can receive asynchronous data updates without polling resources. An example of a observe communication is given in Figure 2.3. The communication starts with a GET request to the resource of interest by a CoAP client, a temperature sensor resource in this case, with the observe option established. Once the CoAP server has processed the request, it saves the client in the list of observers and sends observe updates until the client sends an RST message. Resource observation can also be optimised with conditional observations [19] where clients can establish a configurable criteria to receive resource notifications.

2.2 Run-time deployment and management

Traditionally, deploying and obtaining information from the physical world implies using highly engineered instruments and wired control protocols [20]. However, the newest challenges focus on

systems which support programming and manipulating the physical world so that devices can form themselves as general-purpose devices and support different application types.

The most recent proposals focus on integrating the physical world into the digital world and the Internet. One example is the SENSEI project [21]. The SENSEI project integrates WSNs (Wireless Sensor Network) into a business-driven architecture and offers applications and services over them. OpenIoT [22] is another project which offers a middleware for integrating and implementing IoT solutions with a range of functionalities. Although the integration of the IoT with the Internet is addressed in many approaches, including one of this thesis contributions (Chapter 3), the runtime managing and deploying of physical components in both is still necessary to control physical components.

Other proposals focus on deploying plug&play infrastructures [23]. Nevertheless, even though plug&play makes physical components instantaneously available, the components involved in the plug&play process correspond to autonomous devices that go beyond the sensors and actuators used in Chapter 3. In [24] another plug&play sensor infrastructure for deploying sensors is presented. The infrastructure can plug-in sensors and interpret their values through a sensor interface description. Sensors are subsequently available in the system when the drivers are defined. However, it focuses on the USB and IEEE 1451 standards which are usually found in computers instead of microcontrollers. The CoAP Middleware (CoM) proposed in this thesis (Chapters 3 and 5) is one example of a plug&play component. In [25] Yang et al. recognise the inefficiency of most plug&play protocols used, in terms of energy and memory usage and present μ PnP, a hardware and software solution for the automatic integration of peripherals in resource constrained devices. The hardware solution is based on the passive electrical characteristics of the components of additional hardware circuit added in the peripherals to uniquely identify them. The current prototype uses mini HDMI connectors to connect the peripherals. Once the peripherals have been plugged into the system, their drivers are automatically installed and downloaded when they are not locally available. Once each driver has been installed, it is loaded and the peripherals are ready to use. However, although μ PnP enables automatic peripheral identification and use, the solution requires additional hardware circuits and connectors, and the interconnection is only allowed with ADC, I2C, SPI and UART protocols. The solution presented in this thesis does not require additional hardware to install sensors and the management can be done remotely through a web on a smart-phone or tablet if the sensors are already connected to the microcontroller.

An extension the GSN middleware [26], a middleware which facilitates the deployment and programming of sensor networks, is proposed in [27]. GSN provides a connection with a microcontroller and sensors which is defined semantically through virtual sensors. Virtual sensors abstract

the implementation details from accessing sensors through the semantic definition of data stream. Once a virtual sensor has been defined, GSN looks to see whether there is one wrapper for each data stream defined in the wrapper repository. If the wrapper exists it is instantiated in the middleware and a new connection with the desired sensor is established. Although GSN allows semantic connection with new sensors without programming, the authors identify that this process requires the wrappers to be developed manually and there is no code sharing between wrappers. The authors have defined a new sensor device definition which generates and compiles new wrappers based on semantic definitions. Therefore, the programming and sharing code problems in wrappers apparently disappear. However, even though this solution enables sensor connection semantically and without programming, this solution is focused on the connection with sensors and microcontrollers which offer Application Programming Interfaces (API) or third-party libraries to access their sensor readings and does not provide the mechanisms to instantiate and manage new sensors and actuators.

Employing web services in embedded devices could constitute another solution for managing sensors and actuators in run-time, e.g., the Device Profile for Web Services (DPWS) [28] and RESTful web services [29]. Web services are currently one of the most open and extended ways of connecting things over the Internet. Nevertheless, the complexity due to the HTTP and TCP protocols can be too heavy for embedded devices. The protocol used in this thesis for interconnecting things, CoAP, can provide the benefits of the RESTful web services whilst simultaneously displaying a better performance than the protocols that use other web services [30]. The complexity of the web services can also be moved out to the Gateways [31], but it implies a dependency on Gateways. The Gateway involved in our solution (Chapter 3) facilitates the deployment and access to the underlying devices, since it translates protocols and connects to the cloud. Nevertheless, the underlying devices can be accessed directly without the Gateways through CoAP.

The run-time reconfiguration and upgrading of resource constrained devices is a hot topic addressed in many papers [32], [33]. Ruckebusch et al. present GITAR [32], a system that enables run-time upgrading of network stacks and applications in resource constrained devices. REMOWARE, a composed-based middleware for reconfiguring sensor networks dynamically, is also presented in [33]. Network and firmware updates are necessary requirements to maintain the IoT running over time and actuate over possible bugs or security breaches. In the scope of these approaches, updates can incorporate new libraries into the system, but there is still a lack of a component which can provide the mechanism to instantiate and manage libraries through well-defined interfaces, end users, even external applications as presented in this thesis.

The solution presented in Chapter 3 enables the management and deployment of physical components in resource-constrained devices. This solution leverages the CoAP capabilities for interact-

ing with the IoT, and permits the management and deployment of physical components in run-time with the use of CoAP. Therefore, the system can be comfortably installed in microcontrollers and can be integrated with the Internet. In fact, a Web UI (User Interface) has been developed to facilitate the management of the system by end users. Moreover, this solution does not require additional hardware components for its operation, which encourages its adoption.

2.3 Device-decoupled applications and portability in the Internet of Things

Programming wireless sensor networks has garnered a lot of attention due to its complexity, since it requires knowledge in multiple fields, such as distributed computing, embedded devices and wireless networks. Multiple abstractions have been released to reduce this gap, such as service-oriented, data-driven and group-based programming. The idea behind device-decoupled applications can be related to the macro-programming paradigm in wireless sensor networks [34, 35]. Macro-programming allows sensor networks to be programmed as a whole, rather than programming low-level software for individual nodes. This was a great advance in the development of sensor networks, and greatly reduced the programming efforts in large sensor networks. However, most approaches like EcoCast [36] do not focus on open standards, making their implantation in the continuous-growing IoT difficult. Our approach in Appdaptivity (Chapter 4) follows a similar approach to macro-programming, adapting it to current standards in the field, such as CoAP.

The dependency between devices and applications could be carried to its maximum by directly moving the application logic inside the sensor networks. This approach considerably reduces the latency and the number of packets, since communications are carried out inside the Low power and Lossy Networks (LLNs) and there is no need for communications with external components. In addition, the dependency on other factors such as routing and network reliability in cloud-based applications is reduced. This paradigm is known as in-network processing. Bindings and RESTlets [37] are extensions of CoAP designed by the IDLab research group, which introduce flexible bindings between sensors and actuators and application building blocks respectively. RESTlets configure the inputs, output and some processing tasks just using CoAP on the devices enabling in-network applications which use direct binding between sensors and actuators. T-Res [38] also proposes an extension of CoAP to enable in-network applications. T-Res improves the application logic, which enables scripts for defining custom algorithms to be programmed. However, it requires the use of an embedded Python virtual machine and stores URLs of input sources and destination outputs, which can affect the performance of resource-constrained devices. These solutions optimise the device

interactions and reduce the latency in its communications, nevertheless they do not leverage the changes on the environment to adapt the applications.

A Distributed Data Flow (DDF) programming model to develop IoT applications for fog computing is proposed in [39]. In data flow programming, the application logic is expressed through a directed graph of nodes. Nodes define a set of inputs and outputs and an independent function which does not affect to the rest of the nodes. Thus, nodes are highly portable and reusable for the creation of user applications. DDF addresses the interconnection of cloud, edges, IO and compute nodes which comprise fog computing applications. These interconnections enable different interactions such as things-to-things, fog-to-cloud and cloud-to-fog. Thing functionality is provided through specialised nodes by domain experts. Application developers only need to focus on data flow applications. Glue.things [40] presents an IoT mashup resulting from the COMPOSE European project. Glue.things brings three tools for the application development: the device manager, the composer and the deployment manager. Device manager provides a web-based application to connect IoT devices using different protocols (HTTP, STOMP, MQTT and CoAP). Once the IoT device has been connected to the system, it is available for the composer component to create application logic using a web-based data flow model. Lastly, the deployment manager controls and deploys the mashup applications resulting from the composer component. DDF and glue.things have similarities to our work in that they were built on top of Node-RED [41], a browser-based editor that allows visual data flow programming. However, both of them have only focused on the application development instead of providing a full solution with customisable user applications as Appdaptivity does. The most differentiated part of these approaches and others such as Apple Home, Google Home and BACnet, with respect to our work is the application's adaptability to the underlying physical infrastructure without having to connect these devices and recompile the system. The approach presented in Chapter 4 is intended to offer a novel framework for developing CoAP-based applications without reconfiguration. Appdaptivity enables the design of applications that are independent of the underlying physical devices and can be easily adjusted without recompiling. The IoT device has to be manually connected in glue.things and DDF and discovery are not supported, whereas Appdaptivity provides run-time discovery and update services that deploy portable user applications based on the IoT devices available at any given moment.

PatRICIA [42] is a novel programming model for the IoT which decouples physical devices from user applications. PatRICIA introduces two new abstractions: Intent and IntentScope, which represent desired tasks to be performed in a physical environment and logical group of physical devices respectively. Intents are abstract representations of tasks and are applied on IntentScopes. Device controlling and monitoring are defined by domain experts using tasks through device services.

On the other hand, application developers define device-decoupled applications using Intents which are dynamically transformed into tasks based on the information contained. Therefore, application developers define device-decoupled applications which use device-coupled applications written by domain expert users. Our solution differs from PatRICIA in the programming model. Instead of having different roles of development which decouple applications from devices only once, Appdaptivity has just one programming model which enables device-decoupled programming with visual components which leads to better results [43, 44]. Appdaptivity is based upon a standard and promising IoT protocol, CoAP, with more than 30 open-source implementations for IoT devices [45]. This increases the adoption of Appdaptivity as an application development system in the IoT unlike PatRICIA. Moreover, it has not been demonstrated how portable and personalised applications in PatRICIA could be achieved. Lastly, the programming model for device-decoupled applications in PatRICIA requires a familiarity with Intents programming, which increases the development efforts in comparison to visual component programming.

A macro-programming framework for wireless sensor networks is also proposed in PyoT [46]. PyoT hides the complexity of the network by providing the available resources as a set of Python objects for the application developers. A Web UI has been incorporated into the system where users can carry out basic operations such as sensor monitoring, actuation control and resource listing. Therefore, application logic can be defined in two ways: a shell interface where application developers use the Python API to interact with the CoAP resources, and the Web UI for basic operations. PyoT shares some similarities with respect to Appdaptivity (Chapter 4). On the one hand, PyoT uses CoAP as application protocol and enables automatic discovery of available resources and event actions to be performed when they are detected in an abstract way for end users. On the other hand, PyoT provides a high-level abstraction framework for developing IoT applications which reduces the aforementioned difficulties in programming such devices. Lastly, although more limited, a Web UI enables the application development for non-expert users. As has been pointed out in future work, PyoT does not support 6LoWPAN nor CoAP group abstraction, unlike Appdaptivity, thereby it can decrease both the network and device performance. Finally but not least, the CoAP implementation does not provide support for Datagram Transport Layer Security (DTLS) and the solution could be vulnerable in real IoT deployments such as buildings with access control management.

The importance of the separation between the application logic from device firmware is addressed in [47]. This approach also uses CoAP, providing thin servers— devices as a role of CoAP servers without any application logic. Thin servers are provided with embedded metadata (e.g., geographical information, name, brand) to enable a user-friendly and efficient look-up. The programming model is similar to Web 2.0 mashups, enabling the reuse of deployed services and ap-

plications on the cloud. Actinium [48] provides a run-time container for dynamic management of application scripts. The resulting applications are modelled as resources themselves, so that they can be combined with other apps resulting in multi-layer applications, CoAP servers or even external applications. Although these approaches abstract the application logic from final devices and make use of standard protocols, they are still not completely agnostic from end devices (they use device services or resources instead), cannot be portable and are not provided with the run-time updates necessary to keep applications up-to-date with changes in the underlying physical infrastructure.

In addition of CoAP, many other application protocols have been released during the last years. MQTT (Message Queue Telemetry Transport) [49] is probably the IoT application protocol which has acquired an attention similar to CoAP. MQTT is a publish/subscribe protocol designed to interconnect things. The comparison between CoAP and MQTT has been deeply studied in the literature [50, 51]. Although most studies conclude that the use of each protocol depends on the working scenario and the bitrate, CoAP incorporates RESTful web services that follow the current trends in the Web. It therefore facilitates the integration of the IoT with the Web (WoT), and its observe operation also enables a publish/subscribe model which can be configurable based on the consumer's needs. IoTivity [52] is a software framework for creating IoT applications that implements Open Interconnect Consortium (OIC) standards. OIC was developed by leading technology companies including Samsung, Cisco and Intel to define standards for ensuring the interoperability of the IoT. AllJoyn, another framework for the IoT was recently integrated with IoTivity. IoTivity is also based on CoAP and provides a connectivity abstraction for other non-IP protocols such as Bluetooth, Z-Wave and ZigBee. That connectivity abstraction would improve the interconnection of proprietary IoT devices. Multiple services allow the configuration of devices, accessing them and creating groups of devices like Appdaptivity. Nevertheless, the application logic is couple with the physical infrastructure. OM2M [53] is an Eclipse project that provides a platform for developing services discovering the underlying network. This provides a RESTful service capability layer which offers an abstract layer for handling REST requests. Applications can be created using REST (REpresentational State Transfer) requests that could complicate the application development and a binding protocol for resource-constrained devices like CoAP are not yet available. Kura [54] is another Eclipse project that provides a platform for building IoT applications in gateways. Kura provides a wide range of APIs (including hardware access with protocols such as I2C, USB and GPIOs), which enables the deployment of multiple applications using OSGi, the dynamic component system for Java systems. Kura focuses on gateway IoT applications, therefore it can be part of the underlying physical infrastructure through CoAP and its configurable services can be used for improving the management of the IoT.

Appdaptivity, presented in Chapter 4, is a framework that enables the development of device-decoupled, portable, personalised and adapted to changing contexts in IoT applications. Instead of having multiple programming models, Appdaptivity provides a unified programming model to visually and intuitively develop IoT applications. Changes in the underlying physical infrastructure, such as device dis/connection, are managed by Appdaptivity to adapt the developed applications. Therefore, application developers just need to focus on the application logic of their applications. In addition, Appdaptivity adopts current standards of the IoT, such as CoAP, thus facilitating the adoption of the system and its applicability to resource-constrained devices.

2.4 Internet of Things and Cloud Computing Integration

In this section, different existing proposals for cloud computing and IoT integration are summarised. The proposals cover research projects, enterprise products and open-source projects in multiple areas, so they form a multidisciplinary set of existing solutions in this field. Table 2.1 shows a comparison of the analysed proposals.

OpenIoT [22] is an open-source middleware, co-funded by the European Union's Seventh Framework Programme, for getting information about sensors, actuators and smart devices, offering utility-based IoT services in a cloud platform. OpenIoT leverages the state of the art on middleware frameworks of RFID/WSN and the IoT like XGSN [55] and AspireRFID [56]. Virtual sensors are also promoted by OpenIoT, enabling the concept of Sensing-as-a-Service. The main inclusion is the semantics structure; using ontologies, enabling semantics interaction and interoperability between external systems and offering Open Linked Data interfaces. OpenIoT has also designed a set of applications over the cloud platform to enable on-the-fly specification of service requests to the OpenIoT platform, data visualisation, and configuration and monitoring components over the sensors and OpenIoT services.

Table 2.1: Integration proposals comparison

Platform	REST API	Management Web UI	Connection with devices	Online for use	Available code	Security	License
OpenIoT	Yes	Yes	X-GSN	No	Yes	OAuth 2.0 authentication, permissions and roles.	GNU General Public License version 3
SENSEI	Yes	Yes	CoAP, Sockets, USB and REP instantiation	No	No	Token authentication, permissions, roles	N/A
Nimbits	Yes	Yes	RESTful API	Yes	Partially	Authentication with token and user/password	Apache License Version 2.0 (portions)
Xively	Yes	Yes	RESTful API, Sockets, Web-Sockets and MQTT	Yes	No	Authentication and permissions with OAuth 2.0 and API keys. User/password authentication	End User License Agreement
Paraimpu	Yes	Yes	RESTful API	Yes	No	Token authentication	End User License Agreement
Particle	Yes	Yes	CoAP	Yes	Yes	Authentication with user/-password and OAuth 2.0	Several open-source licenses
Thinking Things	Yes	Yes	N/A	Yes	No	Authentication with token and user/password	End User License Agreement
Sensor Cloud	Yes	Yes	RESTful API	Yes	No	All transactions occur over TLS	End User License Agreement
CloudPLugs	Yes	Yes	MQTT, RESTful API, SmartPlug instantiation and WebSockets	Yes	No	SSL for communications and AES and RSA to protect user sources	End User License Agreement
Stack4Things	Provided by OpenStack	Yes	AMQP and CoAP	No	No	N/A	N/A

The SENSEI project [57], [21] is also an Integrated Project in the EU's Seventh Framework Programme, which aims to integrate heterogeneous WSN (Wireless Sensor and Actor Networks) in an open business-drive architecture in order to offer services and applications on them. SENSEI's result is a secure marketplace where users can manage and access information about WSN resources. The Resource is the main concept in the architecture, which abstracts sensors, actuators, processors and software components through a semantic specification, at the same time that offers a set of services. SENSEI also offers an Open Service Interface and a management UI designed to enable machine processing which facilitates the dynamic composition, discovery and instantiation of new services. For communication and management with end points, SENSEI makes use of the Resource End Point (REP), which makes resources on embedded devices or sensors available to the SENSEI system through RESTful interfaces. REPs can be directly deployed on devices through TinyOS, Contiky, and Android implementations or by REP gateways which act as a resource proxy between the underlying WSN and the SENSEI systems.

Nimbits [58] is an open-source platform for connecting things to the cloud, and one another. Nimbits could be downloaded and installed privately in addition to using a public cloud Nimbits deployed in Google App Engine. Currently, Nimbits is deprecated. Nimbits is composed of two main components: a web server which records and processes geo and time stamped data and executes user rules on the data such as push notification, emails and XMPP messages; and a Java library for developing and connecting new applications on the platform. Moreover, Nimbits contains a library for Arduino support and an Android App to manage and visualise all connected data prints. The data is sent by clients using the JSON (JavaScript Object Notation) format. Users can configure data points to behave in many cascading ways when new data is recorded, generating different triggers and alerts in each situation. Several parts of Nimbits were provided with an open-source license, but the core components do not contain an open-source license, though the latter could be downloaded for free.

Xively [59] is an IoT platform as a Service for developing applications over connected things. Xively offers different communication methods for connecting things, such as RESTful, HTTP, Sockets and MQTT; different data formats such as JSON, XML and CSV, and official libraries for dozens of languages and platforms like Arduino, Android, Java and C. The Xively API has been built to read and write data and manage products and devices. Xively has been designed to support a development process in three stages: development for testing devices and applications, deployment for turning prototypes into products, and management for batching products and support real-time devices. This can be done by the Web UI offered by Xively. Furthermore, the platform contains end-to-end security for protecting communication channels and for fine-grained permissions. Xively has

a proprietary license, allowing users and systems to connect with its platform.

Paraimpu [60] introduces a novel paradigm to share objects, data and functionalities with other people in order to attain a participative and collaborative use of a friend's things. Paraimpu is a web platform which allows the adding, use, inter-connection and sharing of real smart objects and virtual things like services on the Web and social networks. Users can define connections between a sensor and an actuator through a sequence of rules and actions, and manage the thing's privacy. As a result, a user can publish on social media, like Facebook, an information message when a sensor's temperature exceeds a certain threshold. The platform promotes the concept of the WoT, where multiple connected smart objects communicate using Web protocols. Currently, Paraimpu has a proprietary license and only allows things to be used or connected to its platform.

An integration with their own products has been released by Particle [61]. Particle offers a set of development kits at low prices—starting from 19\$ a microcontroller with Wi-Fi support—with a free hosted cloud platform for each device. The Particle team has obviously thought of the importance of IoT software and they have designed a set of software tools for developing, managing and monitoring the Particle devices. The software tools include a mobile app for remote control and monitoring, Web and local IDEs, a Particle CLI and programming support for Node.js, Arduino-like development and a REST API. However, the main advantage of Particle is that all their software contains an open-source license and all are published on GitHub (Particle GitHub: <https://github.com/particle-iot>). Moreover, the solutions have also been integrated with several external systems such as a smart bed and a water monitoring system. The development kits have Wi-Fi and 2G/3G support, so together with their low price and the hosted cloud, they form a cheap choice to start with IoT and cloud computing.

As is Particle, Thinking Things [62] is a project founded by Telefonica with their own devices, which aims to connect IoT devices and cloud technologies. Unlike Particle, Thinking Things has presented a set of modular development kits with integrated sensors—air temperature, air humidity and ambient light—making up a stack of connected blocks that you can put together like Lego pieces. Furthermore, a REST API and mechanisms to monitor and define action rules form part of the platform. Thinking Things also offers devices with Arduino-compatibility, with free cloud connection and utilisation. A collaboration with a pizza company has concluded with the development of an embedded button to order pizzas in real-time. However, Thinking Things development kits are more expensive than Particle kits, they do not contain as many software tools as Particle and they only have connectivity support for GSM communications.

SensorCloud [63] leverages the cloud computing technologies to provide a data storage, visualisation and management platform. For the IoT interconnection, the company MicroStrain offers a

specific gateway which collects data from its sensors and pushes it to SensorCloud. Moreover, data from other IoT devices can be published on SensorCloud through the RESTful API offered. SensorCloud also incorporates the MathEngine—a set of software tools to process, analyse and monitor sensor data—, a mechanism to upload you own scripts to process data, and an alert engine (as other integration platforms). Lastly, SensorCloud offers multiple plans to use its platform, from free ones with 25.000 transactions per month.

The concept of a secure and robust software agent is promoted by CloudPlugs [64]. The agent, known as SmartPlug, connects with the CloudPlugs IoT platform in addition to having local intelligence for communication and to control other devices. CloudPlugs also has a pay-as-you-grow IoT platform with which you can manage and deploy the underlying IoT deployment and access your IoT data through a REST API. SmartPlug has also been designed to deploy applications over a Node.js server, offering communication support with local sensors through several local interfaces like ZigBee and Bluetooth and multi-operating system support. Final devices can connect directly to the CloudPlugs platform through several communication protocols and all things can communicate with each other independently from the protocol through a smart message bus. CloudPlugs is the only platform that besides the cloud computing integration offers mechanisms to support local intelligence. Although, CloudPlugs has a pay-as-you-grow platform, it also offers a free account to test the platform.

Stack4Things [65] proposes an extension of the OpenStack platform in order to enable a cloud-oriented infrastructure for managing the IoT. Stack4Things relies on OpenStack as a cloud platform as well as virtual infrastructure manager to provide higher level services applied to the IoT. The s4tProbe components constitute the end components in the system, which are deployed in Arduino YUN boards. The s4tProbe components connect the IoT with the OpenStack platform providing a pull-based design through the AQMP protocol and a push based design through CoAP. The OpenStack platform has been extended to support new UI functionalities, integrate s4tProbe components and provide complex event processing.

The comparison of integration proposals in Table 2.1 shows that all the surveyed proposals have REST API support and a management Web UI. The connection with devices is established in different ways. On the one hand Nimbits, Paraimpu and Sensor Cloud just rely on a RESTful API whereas Particle and Stack4Things rely on the lightweight CoAP protocol. Stack4Things also relies on AQMP for a pull-based design. OpenIoT, SENSEI and CloudPlugs rely on, respectively, middleware, gateways and agents to abstract the underlying IoT deployment. In the case of SENSEI and CloudPlugs, they also offer support for other protocols. Lastly, Xively allows communications with several protocols and Thinking Things abstracts the communication through its own devices.

Most integration proposals are available online for use and deployment of an IoT infrastructure except for SENSEI, Nimbits, Stack4Things and OpenIoT, but the latter has the source code available online. In the case of Thinking Things and Particle, it is free to use the cloud, but they require the use of their own devices for this. Moreover, Particle goes beyond and apart from the free use of cloud, all source code of the platform's components contain open-source licenses and are available online. Stack4Things leverages the cloud features of a well-established solution as OpenStack to build the system. Nimbits also contains an open-source license for its source, but in this case only for some portions. For security aspects, most components make use of tokens in order to allow a secure form of authentication, identification and permissions over a set of users or devices. On the other hand, SensorCloud and CloudPlugs protect their communication through secure protocols, but they do not make use of tokens as it can be overly weighty for embedded devices.

The λ -CoAP architecture described in Chapter 5 also represents an integration between the IoT and cloud computing. However, this architecture has been designed to meet the requirements of time-sensitive use cases, since it focuses on applications that require processing large amounts of data with low latency. This has been achieved thanks to the adoption of the Lambda Architecture to reduce the processing time in the cloud side, and an edge layer to reduce both the latency and bandwidth in IoT-cloud communications. The IoT has been integrated with the de facto IoT protocol, CoAP, enabling its use in resource-constrained devices.

2.5 Blockchain and Internet of Things Integration

In finances, there has been a notable emergence of alternative cryptocurrencies (altcoins, 1486 according to coinmarketcap) that have built a new market and have enabled new forms of payment and investment. Examples of them are Ripple [66], Litecoin [67], Nxt [68], Peercoin [69], Bitshares [70], Dogecoin [71], Namecoin [72], Dash[73], and Monero [74]. This new market has also led to the appearance of new applications for payment, exchange and trading infrastructures for these new currencies.

Beyond finances, the distributed and trustless ledger of blockchain has been identified as an ideal solution for traceability systems. What were economic transactions in Bitcoin, have become changes of the plotted objects. In this way the chain stores the objects and all their changes, allowing a complete, open and reliable traceability, only possible through this technology. There are some ongoing projects from prominent companies such as IBM, Unilever, Walmart and Nestle collaborating for food traceability [75] or Renault to track vehicle maintenance history [76].

Identity verification has also become a popular application of the technology, blockchain pro-

vides an open trusted distributed ledger that can be used to store identities. This makes the global management of identities possible. In the field of e-government many countries have proposed the use of blockchain technologies for instance: for passports in Dubai [77], e-identity in Estonia [78], Illinois to digitise birth certificates [79] and for land registration in India [80].

Additionally, the integration of smart contracts opens up a world of possibilities for many industries: energy, insurance, mortgages, music royalty payments, real estate, gambling, betting, etc. Cloud storage, education, or e-health are other areas in which blockchain applications have been proposed. Table 2.2 lists some of these applications in different areas.

IoT blockchain applications

Although the use of blockchain in the IoT is relatively recent, there are already a large number of proposals where this technology is used in different ways to improve current IoT technology. A summary of some of these proposals is shown in Table 2.3.

In fact, IoT can join forces with blockchain in many scenarios. Almost all traceability applications can benefit from the inclusion of IoT devices, for instance, sensors that are attached to the product to be traced can provide information during its distribution. Similarly, many applications that digitise the world by providing sensed data can be enriched with blockchain technology. It can help to increase data legitimacy or reliability and moreover provide distributed identity, authentication and authorisation mechanisms without the need for central authorities.

Blockchain could be a powerful candidate to make the smart city concept a reality. The concept of the smart city is based on smart IoT devices that can work autonomously. Blockchain can increase the autonomy of devices since it eases interaction and coordination by providing a distributed open ledger where devices can query trusted information with reliability. Moreover, the autonomy that blockchain enables, favours the creation of new IoT marketplaces. [108, 100, 109, 110].

As Table 2.3 shows, Ethereum is the most popular platform for IoT blockchain applications. Ethereum [111] provides more features than Bitcoin, and the inclusion of smart contracts greatly expands the possible applications.

In LO3 Energy [108] an energy microgrid that uses blockchain has been demonstrated in Brooklyn (USA), southern Germany and South Australia. Microgrids are localised groupings of electricity generation, energy storage, and electrical loads. The project builds a community energy marketplace that creates a decentralised P2P energy network that is coordinated with the broader power grid. It is the first ever energy-blockchain-platform, that allows devices at the grid edge to securely and directly transact for energy sales among microgrid participants. A hybrid device measures a building's energy production in use and sends data to the network.

Table 2.2: Blockchain applications

Application	Classification
Ripple [66]	Cryptocurrency
Litecoin [67]	
Nxt [68]	
Peercoin [69]	
Dogecoin [71]	
Namecoin [72]	
Dash [73]	
Monero [74]	
BitPay [81]	New payment infrastructures
Abra [82]	
BitNation [83]	Identity verification
Onename [84]	
Keybase [85]	
ShoCard [86]	
Passport management [77]	e-government
e-identity [78]	
Birth certificates [79]	
Land registration [80]	
Follow my vote [87]	
Tierion [88]	Verification of ownership or provenance
Proof of Existence [89]	
Factom [90]	
Everledger [91]	
MIT's digital diploma [92]	Product traceability
Provenance.org [93]	
SkuChain [94]	
IBM Food traceability [75]	
Renault vehicle maintenance history tracking [76]	e-health
Robomed [95]	
Medrec [96]	Energy, insurance and mortgages
Synechron [97]	
Ubitquity [98]	Real estates
Atlant [99]	
Slock.it [100]	Renting, sharing and selling
DAO.Casino [101]	Gambling and betting
Peerplays [102]	
Wagerr [103]	
Storj [104]	Cloud storage
Sony education history [105]	Education
Ujo [106]	Music royalty payments
Resonate[107]	

Table 2.3: IoT-Blockchain applications

Application	Classification	Platform
LO3 Energy [108]	Energy microgrid	Ethereum
ADEPT [112]	Smart contracts involving IoT devices	
Slock.it [100]	Renting/Selling/Sharing smart objects	
Aigang [109]	Insurance network for IoT assets	
MyBit [110]	Investment in IoT devices	
AeroToken [113]	Sharing airspace market for drone navigation	
Chain of things [114]	Identity, security and interoperability	
Chronicled [115]	Identity, data provenance and automation	Multiplatform
Modum [116]	Data integrity for the supply chain	
Riddle and Code [117]	Sharing and machine economy	
Blockchain of things [118]	Secure connectivity between IoT devices	

The project for Autonomous Decentralised Peer-to-Peer Telemetry (ADEPT) led by IBM and Samsung [112] aims to promote device autonomy, and to this end they use blockchain technology to ensure code execution on edge devices. ADEPT uses three protocols: Telehash, Bittorrent and Ethereum, for messaging, file sharing and blockchain, respectively. Blockchain technology provides authentication, engagement, contracts and checklists. Their proof of concept consists in smart washing machines that use smart contracts to buy detergent supplies from retailers.

The blockchain framework proposed by Slock.it [100] aims to address security, identity, coordination and privacy over billions of IoT devices. The objective is to build a sharing economy where each IoT asset can be rented securely and quickly without the need for any authority. They are working on a variety of projects, for instance a marketplace to provide a charging infrastructure for electric vehicles, called Blockcharge. This solution uses a smartplug, a mobile application to activate the plug and control the charge, and the blockchain to pay for the services. They are also working on a smart lock to automate apartment renting.

Aigang [109] is an autonomous insurance network for IoT assets. Aigang has deployed smart contracts over the Ethereum test-bed that issue policies, conduct risk assessment and process claims automatically. They have also created a custom virtual currency (AIX). They offer investment opportunities in different products with different risk levels and potential gains. Smart contracts connect intelligent devices with insurance policies. Through these contracts the devices can order maintenance and insurance payment can be automated. With the inclusion of Oracles to report events, claim handling can be automatically handled.

MyBit [110] plans to build an ecosystem of services where IoT assets (from drones to cars) are

owned by a group of people, and the revenues are shared. A new financing model and investment opportunity open to anyone. Ethereum smart contracts are used to automate processes: when the IoT devices generate revenue the investors automatically receive a share of the profits proportionate to their ownership stake. A central smart contract is responsible for the control, maintenance and updating of the platform. The platform defines different asset types, and IoT devices are linked to assets. Once installed they send and request information through an API. Oracles are used to connect devices to the network.

AeroToken [113] aims to create a real-time navigation and property access authorisation system for low-altitude commercial drone services. They provide a solution to drone navigation by voluntarily sharing airspace over properties. This helps to solve the problem of drone operation permission, building a new sharing marketplace. Property owners offer their airspace using smart contracts in the blockchain, and drone service providers pay for a temporary access. The application has been developed using Ethereum smart contracts.

The Chain of Things [114] is a blockchain-enabling IoT research lab that proposes Maru, an integrated blockchain and IoT hardware solution. Blockchain provides devices with universal identity from birth, security and interoperability. Three case studies are presented: Chain of Security, focused on providing security to IoT through blockchain; Chain of Solar, that aims to connect solar panels to the blockchain to store produced energy for a variety of applications; and Chain of Shipping, which plans to improve security in the shipping and logistics industry. Data logging devices are used to send data to the network. The proof of concept has been developed over the Ethereum network.

Chronicle [115] has developed several IoT products provided with cryptographic capabilities with the objective of creating the world's most trusted IoT and supply chain ecosystems. Chronicle platform includes a blockchain synchronisation server capable of synchronising with multiple blockchain systems such as Quorum, Ethereum and Hyperledger. Registration and verification of device identity is performed through smart contracts.

The Modum [116] blockchain solution aims to provide data integrity for physical products, focused on improving supply chain processes. The Modum sensors record environmental conditions during shipments. Modum has been designed to work with different platforms. A solution for the distribution of medical products has been developed using Ethereum blockchain. Ethereum smart contracts are used to verify sensor data each time goods change ownership, the contract validates that the transaction meets the customer's standards. The solution integrates sensor tags that collect measurements during transit, a mobile application, used to connect and activate sensors and change ownership, and a dashboard to analyse sensor collected data after shipment reception.

Twin of Things is a solution for securing the ownership and provenance of everyday objects [117] developed by Riddle and Code. The solution combines blockchain and cryptography to generate a hardware-based digital identity for all connected physical objects. Communication and transactions between devices is autonomously and securely performed thanks to blockchain. A highly secure crypto chip enables each device to become a blockchain node. The chip is produced in the form of an adhesive non-removable NFC tag and an Android application is used to carry out a blockchain transaction to register the unique, tamper-proof identity of the chip. Once validated in the network, it can interact with other devices. The solution has been designed to be blockchain agnostic, it can currently work with Ethereum, Bitcoin and BigchainDB blockchains.

Lastly, the Blockchain of Things [118] provides a secure open communication platform for industrial IoT integration. They propose Catenis, a web service layer for rapid Bitcoin blockchain integration, with end-to-end encryption. It can also be adapted to Ethereum, Hyperledger and other blockchains. In Catenis, each IoT device is represented as a virtual device in a Catenis Hub and Gateways (with increased security). Each virtual device will manage a host of Catenis services for the IoT device.

The contribution presented in Chapter 6 does not, in itself, represent an integration between the IoT and Blockchain, rather it analyses the challenges and open issues to overcome that integration. In addition, several blockchain platforms are analysed and an evaluation of blockchain nodes in IoT devices is undertaken to validate the feasibility of Blockchain in the IoT. Therefore, this contribution has its as goal to define the guidelines for future IoT and Blockchain integrations.

3

Run-time Deployment and Management of CoAP Physical Resources for the Internet of Things

In this chapter, the first contribution of this thesis is presented: a run-time deployment and management system for physical resources. This contribution will enable the deployment and management of physical sensors and actuators without having to access and program the micro-controller boards. First, the problem statement and research goals are defined in 3.1. The system and all its components are presented in Section 3.2. Then, Section 3.3 shows how the system has been implemented. Finally, the evaluation of the system is presented in Section 3.4.

3.1 Problem Statement and Research Goals

The continuous growth of the IoT in recent years has meant it is increasingly more present, as IoT scenarios such as smart homes and smart cities become part of our everyday lives. This growth has, in turn, affected the number of IoT solutions, devices and applications available to end users. As mentioned, standardisation is key in the IoT, where protocols such as 6LoWPAN, RPL and CoAP bring the IoT closer to end users.

Resources in CoAP represent an operation on a constrained web service. Resource operations can contain both sensors (e.g. a temperature sensor) and actuators (e.g. a light actuator). Normally, resources are defined in compile time which does not permit managing the resources in run-time. Although CoAP defines the guidelines for creating resources in run-time through a CoAP request, it does not establish the rules for managing and deploying resources with physical components such as sensors and actuators.

The goal of this contribution is to leverage CoAP's capabilities for creating and serving resources, and provide mechanisms for controlling the resources with physical components in run-time. The latter does not require programming or installing external components, so end users could incorporate new sensors or actuators in their installed microcontroller without having to access and program the microcontroller board.

3.2 Run-time Deployment and Management of CoAP Resources

The overall architecture in the run-time deployment and management system comprises three differentiated components. On the one hand, the Web UI provides a user-friendly and accessible way to manage the resources belonging to the associated CoMs (CoAP Middleware). Through the Web UI users can either create a new resource on a CoM or edit and remove previously created resources. On the other hand, a Smart Gateway provides a proxy for translating HTTP to CoAP which is not only used by the Web UI to control the CoMs but could also be used for external HTTP clients through its REST interface. The Smart Gateway also comprises a Database/Cache which stores information about the resources of the connected middlewares and their data in order to protect them from high access rates. Lastly, the CoM offers a CoAP server and a set of resources which can be managed by the Smart Gateway and the Web UI.

Figure 3.1 shows an overview of the architecture with the aforementioned components and the data flow between them. The data flow starts with an HTTP request for an operation to interact

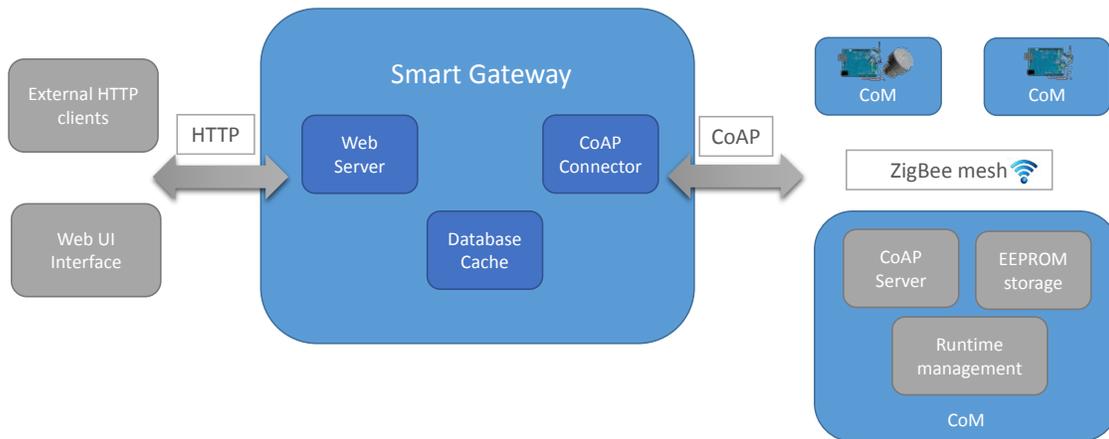


Figure 3.1: Run-time deployment and management of CoAP resources architecture overview

with a specific resource from the Web UI or an external HTTP client. Then, the Smart Gateway translates the HTTP request to CoAP and transmits it to the objective CoM. Finally, the Smart Gateway responds through HTTP once the CoM responds with the operation's result.

3.2.1 CoAP Middleware

The CoM is one of the main components in the system, so it serves the resources registered to it and enables the management and deployment of these resources. The CoM is allocated in the microcontrollers deployed. Therefore, the CoM provides the way for end users to install and access resources.

Connecting sensors or actuators in a microcontroller involves connecting them in some connector/s or pin/s. That is the main idea with which we have designed this approach. Users can connect a sensor or an actuator to connectors in a microcontroller. Once the sensor or the actuator has been correctly connected to the microcontroller, users can register the installed resource and its connectors in order to establish the resource available for interacting with it in the CoM. Therefore, users can easily create a new resource, connecting it to the microcontroller and registering it on the Web UI.

Although some sensors and actuators do not need external libraries to interact with them, such as some analogical sensors, many others do require them. The available libraries for sensors and actuators in microcontrollers could require a large amount of storage. Hence microcontrollers with storage limitations are far from being supported. To alleviate this, we propose the concept of *objective middleware*. An objective middleware is a middleware that supports a set of suitable sensors

or actuators for a certain environment. For instance, an objective middleware designed for smart cities with support for environmental sensors (CO₂, humidity and temperature). Objective middlewares will form as many groups as there are situations and possible scenarios constituting the IoT, since the system architecture is designed to be easily increased. Furthermore, generic libraries with support for well-known protocols such as I²C, SPI or UART can also be created in order to reduce the library size and enable different components to be installed with the same library.

The CoM has been designed to support the creation and association of new sensors and actuators without modifying the source code, so it avoids the human errors that are produced when modifying code and provides an easy way to define new components in the system. To define new components, a new class should extend and implement the virtual methods defined in the CoapSensor class. The CoapSensor class provides the mechanisms to access and manage the sensors and actuators defined for the CoM. A new sensor or actuator can be defined, implementing the virtual methods of the CoAPSensor class shown in Listing 3.1.

```
Class CoapSensor

//Initialise the sensor/actuator
Public Subprogram init(name, connectors)
...
End Subprogram

//Set the value input_data received as argument
Public Subprogram set_value(input_data , input_data_len , output_data ,
    output_data_len);
...
End Subprogram

// Get the current value of the sensor
Public Subprogram get_value(output_data , output_data_len)
...
End Subprogram

// Uncouple the sensor/actuator
Public Subprogram disable()
...
End Subprogram
End Class
```

Listing 3.1: CoapSensor class pseudocode

The `init` method in the `CoapSensor` class is responsible for initialising the necessary components to deploy the desired component based on the connectors' information provided as argument. The `get_value` and `set_value` methods enable the query or the actuation in the sensor or the actuator respectively. The CoM detects the nature of the component implemented (sensor or actuator) based on the methods implemented. Hence in the case of defining a new sensor, it is only necessary to implement the `get_value` method, and the `set_value` method in the case of a new actuator. Lastly, the `disable` method uncouples the components connected.

New libraries can be created dynamically in the CoM through CoAP requests. When a CoM receives a CoAP POST request with a valid payload for creating libraries, the CoM gets the library type and finds the target library by means of the Registry Software Design Pattern [119]. The Registry Pattern allows the CoM to get references and create classes dynamically through the class name, so it avoids modifying the solution's source code and enables the creation of new classes just by including them in the project solution. Once the libraries have been instantiated in the CoM, their references are saved in the memory and can be updated or removed through CoAP PUT and DELETE requests respectively. If something fails during an operation (e.g modules activated wrongly), the libraries are not instantiated and a CoAP error response is sent to inform the users of the operation's result. Figure 3.2 shows the behaviour of each CoM when it receives a CoAP request. In order to reduce the flow's complexity, the error cases that can be thrown after each operation have been excluded.

Each CoM has, by default, one resource, called `lib`, which returns the supported sensors and actuators which can be installed in it. The resource `lib` response replies with the following Core Link Format [120] (Listing 3.2).

```
RES: 2.05 Content
</lib >;a="0 1 3 ..."
```

Listing 3.2: Lib response example

A new link-extension parameter, a , has been added to the Core Link Format to enable the supported libraries in each library response. The `lib` response includes the URI for accessing the resource and the list of available id types separated by spaces in the a parameter. Moreover, the Smart Gateway will add information such as the name and the number of connectors to each supported library when the response is received. The latter also prevents the overload of the CoMs since they only need to send the id type and the rest of the information is completed by the Smart Gateway.

The CoAP resource discovery has also been modified to incorporate additional information from the CoAP resources created, such as the resource type deployed and the connectors used in the installation. A new link-extension parameter p is in charge of indicating the connectors

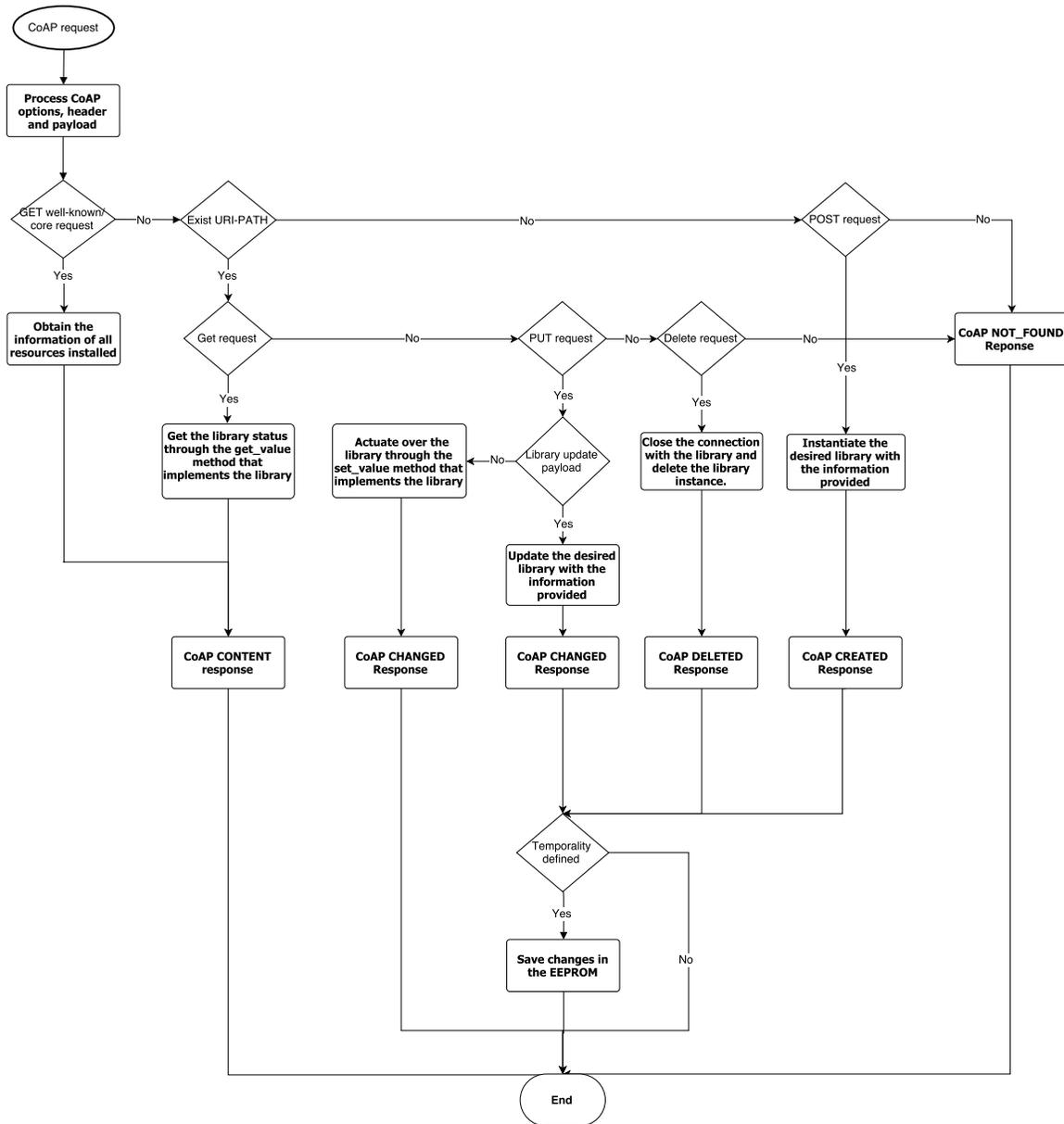


Figure 3.2: Flow chart with the CoM behaviour on CoAP request

used, separated by spaces, and the resource type *rt* attribute has been reused to indicate the type of resource deployed. The *p* parameter is also included in the resource discovery process to inform users in which connectors the sensors are deployed. Hence a possible response from a CoM with two resources could be the one in Listing 3.3.

Response in Listing 3.3 shows the resource discovery of a CoM with two resources: gas and light. The gas resource has been installed in the connectors 5, 6 and 1 with an ID type 3 (CO

gas sensor), and it is a sensor resource (core.s, sensor in Core Link format). On the other hand, the resource light is an actuator (core.a, actuator in Core Link format) and has been deployed in connector 12 with the ID type 2 (light actuator).

```
RES: 2.05 Content
</gas>;rt="3";if="core.s";p="5 6 1",
</light>;rt="2";if="core.a";p="12"
```

Listing 3.3: Resource discovery response example

When resources are created in the architecture, they are available for interaction, but when microcontrollers are reloaded or stopped for some reason (power, error, damage, and so on) the changes may be lost. For this reason, the resource persistence also has to be taken into account. We have therefore used the EEPROM storage for the majority of the microcontrollers to store the information of the resources. Therefore, when CoM starts, first of all it checks whether or not there are indeed resources to be restored in the EEPROM storage. In the case that there are resources in the EEPROM, they are restored, based on all the information provided during their creation (name, resource type and connectors). This avoids the changes being lost when the CoM deviates from its normal behaviour, which in turn provides resource persistence.

3.2.2 Smart Gateway

The Smart Gateway provides a cross-proxy to convert HTTP requests into CoAP [121], [122]. This avoids having to know the format or additional information from the CoAP request. As all the information is gathered in an URI, it sends the format of the CoAP requests which are hidden from the end users, and enables an accessible way to communicate with them through well-defined URIs.

In order to carry out tests or simply an operation which is not required in a reload of the CoM, a new concept of *temporality* is introduced into the system. Temporality implies that all changes made in one CoM (creations, updates, deletions) can optionally persist in the CoM depending on the user's needs. Accordingly, users can create, update or remove a resource in a CoM and decide whether or not to apply these changes throughout its useful life. This also protects the storage in the case of multiple uses, like, for example, testing, because usually storage has a limited number of writings. It has been implemented with a new link-extension parameter *t* which indicates with the value 1 that the operation is temporal, and 0 if not.

All HTTP operations in the proxy follow a well-defined interface shown in Table 3.1. The API is used by the Web UI to manage the CoAP resources in the CoMs but it could also be integrated into external applications.

Table 3.1: Smart Gateway API

URI	HTTP Method	Description
/nodes	GET	Obtain the CoM list from the Smart Gateway
/CoM/.well-known/core	GET	Obtain the CoM resource list
/CoM/lib	GET	Obtain the available libraries from the CoM
/CoM/res	GET	Obtain the data of the resource <i>res</i> in the CoM
/CoM/res/val	PUT	Actuate over <i>res</i> with the value <i>val</i>
/CoM/res/typ? p=[v1;v2;..]&t=[01]	POST	Create a resource <i>res</i> with type <i>typ</i> , connections contains in <i>p</i> and temporality <i>t</i> defined
/CoM/res/new_res/ typ?p=[v1;v2;..]&t=[01]	PUT	Update the resource <i>res</i> with a type <i>typ</i> , a new name in <i>new_res</i> , connections contains in <i>p</i> and temporality <i>t</i> defined
/CoM/res?t=[01]	DELETE	Remove the resource <i>res</i> with the temporality <i>t</i> defined

As can be seen in Table 3.1, the Smart Gateway enables a set of operations for interacting with resources as well as managing and deploying them. All operations can be done through a URI, so it is not necessary to learn any format or include any payload more than the URI itself. Some operations for managing the resource observations are also included. To ensure the matching between requests and responses, each request sent by the Smart Gateway includes a unique token which is validated with the token of the corresponding response received.

```
HTTP PUT /temp/new_uri/0?p=[1;4]&t=1
```

```
CoAP PUT /temp
</new_uri>rt="0";p="1 4";t="1"
```

Listing 3.4: Translation between HTTP and CoAP

Creating and updating resources requires some information in the CoAP payload. That information includes the connectors' information *p*, the resource type *rt*, its new resource URI and its temporality *t*. DELETE requests only require the temporal parameter to remove the resource. The

translation in Listing 3.4 shows an example of the translation done by the Smart Gateway to temporarily update a resource.

Lastly, a Database/Cache in the Smart Gateway stores information on the CoM resources as well as data from its own in some cases. Periodically, with a configurable timeout, the CoMs send a request to update its registration. The latter also maintains the contact between the Smart Gateway and the CoMs and can detect possible service interruptions. CoMs are registered with the Smart Gateways when they start up, through a registration process. When data from an observable resource is received in the Smart Gateway, it stores the data used in the request for this resource. As a result, the number of requests to the CoMs is reduced and this prevents overload and reduces its power consumption.

3.2.3 Web UI

The Web UI enables registered users to straightforwardly access the CoMs and CoAP resources involved. The Web UI also avoids having to use HTTP clients and learn the URI formats for managing and actuating over CoAP resources. Moreover, the responsive design allows a proper visualisation on many devices, thus the Web UI can be accessed on smart-phones, PCs and tablets.

The Web UI is organised on several levels: the associated CoMs, the resources of the CoMs, and the interaction with their resources. Once the registered users have accessed the system using their credentials, a form with the associated CoMs in the Smart Gateway is provided. Users can visualise the resources belonging to each CoM by selecting the option of view resources in each CoM. If the option is selected, a form, as shown in Figure 3.3a, with the list of resources is provided. This list shows the resources that belong to the selected CoM with its corresponding information such as its resource type, connectors used and four types of operations—create a new resource, interact, update or delete one of them—which are available for each of the resources. The interaction with the resource can differ, depending on the resource type. In the case of resources which contain a sensor, the communication will be established through GET requests, both for HTTP and CoAP. When a user decides to interact with a sensor resource, a GET request is sent to the Smart Gateway and a pop-up window shows the data response received. In the case of resources associated with actuators, the communication is done through PUT requests. The pop-up window in a PUT request (Figure 3.3b) first enables the insertion of a value to control the state of the resource, e.g., a behaviour in one actuator resource could be to turn it on with a 1 value, turn it off with a 0 value and create special behaviours with other values. The behaviours will depend on the possibilities of each resource type and the possible values for interacting with them will be indicated to the end users. When the value is inserted and the user confirms the operation, the PUT request is sent to the Smart Gateway and the

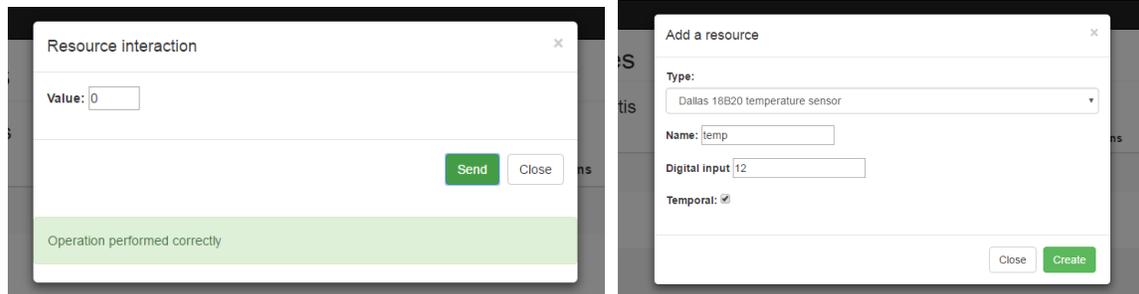
CoAP Resources

CoAP resource list of

Resource Name	Resource Type	Library Type	Connections	Operations
temp	sensor	Dallas 18B20 temperature sensor	[37]	  
light	actuator	Digital LED actuator	[7]	  

[Add resource](#)

a Resource list of a CoAP Middleware



b Pop-up window for actuation with a actuator resource

c Pop-up window for adding a resource

Figure 3.3: Screenshots of the Web UI

pop-up window shows the result of the operation when the response is received. These behaviours follow the good practices of RESTful web services

Creating resources, Figure 3.3c, enables the creation of new resources from the resource types available in the CoM selected. In the resource updates, users can update the access name and connectors of a resource, choosing the temporality of the operation. Equally, users can remove each resource, establishing its temporality.

When an end user decides to create a new resource in the form of list of CoM resources (Figure 3.3a), he/she only has to press the new resource button to create it. As the CoMs can have different objective middleware depending on the scenario, the first step is for the Web UI to send a request to the CoM to obtain its supported libraries. When the Web UI obtains the information requested, a pop-up window to create the new resource is displayed (Figure 3.3c). In the creating resource window, the user can select the resource type, introduce its access name and connector information and decide whether or not the changes will be temporal. When the user introduces the information and presses the confirm button, the information is sent to the corresponding Smart Gateway. Then,

the Smart Gateway translates the request to the CoAP format defined and responds to the Web UI when it receives the response from the CoM. If the resource has been correctly created, the user will see the resource in the resource list form and interaction with it is allowed from this moment onwards.

3.3 Implementation

The CoM is based on the implementation of the CoAP library for Arduino with XBee radio support in [123]. This implementation follows the 8th version of CoAP and supports the observe option and provides a well-defined abstraction of the CoAP protocol. The CoM has introduced changes in the implementation provided, such as the capacity to manage and deploy resources in run-time in addition to ZigBee communication support and the EEPROM management. As stated, an objective middleware has been defined to control the supported libraries, and the limited memory of the IoT devices. Therefore, different objective middlewares will contain different sets of supported libraries. These libraries are included at compile-time in the CoM, as shown in Listing 3.1, and can be queried and instantiated at run-time by end users through CoAP, the Smart Gateway and the Web UI. When an end user checks the available libraries in a CoM, a CoAP GET request is sent to the CoM to get the list of available libraries installed to be instantiated. All the operations with regard to this system can be saved. In that case, when the CoM is restarted for any reason, all the saved operations are restored when the CoM is once again available.

The Smart Gateway is responsible for translating HTTP requests into CoAP, and vice versa. The Smart Gateway receives HTTP requests through an REST API defined with the Python open source Wheezy Web server. Wheezy Web is a highly concurrent, lightweight and high performance Web framework that enables a REST API to access the operations in each Smart Gateway. The CoAP support in the Smart Gateway has been developed from scratch with ZigBee communication. The information contained in the Database/Cache is stored in the memory with access control.

Lastly, the Web UI has been designed in two different components through the widely-used back-end/front-end paradigm. The open source Python Web framework Django has been chosen to allocate the back-end. Django allows Web applications or REST APIs to be created with very little code and very quickly. The open source JavaScript frameworks AngularJS and Bootstrap have been utilised for developing the front-end. Whilst AngularJS enables the creation of dynamic Web applications and the back-end interaction, Bootstrap contains a large set of components and utilities for creating responsive Web applications with very little effort. The Web UI enables the management and visualisation of the system through a user-friendly and accessible interface. Some

information on the Web UI, like the registered users, is contained in an SQLite database.

3.4 Evaluation

3.4.1 Smart Home case study

The evaluation has been done in one of the main case studies in the IoT, the Smart Home [124]. The system was evaluated in a room where three CoMs connected through batteries were deployed in order to enable the deployment of three different components (a humidity sensor, a temperature sensor and a light actuator) in each one of them. In the case study, the behaviour of the aforementioned components has also been tested with the three operations to manage sensors and actuators in run-time: creation, updating and elimination. Therefore, the Smart Home case study comprises a real IoT system with two IoT sensors (temperature and humidity) and one IoT light actuator.

The Smart Gateway was deployed in a laptop in the same room with a ZigBee connection. Once each CoM had been connected to a battery, its corresponding sensor or actuator was attached to some free connectors in the microcontroller and registered in the CoM as indicated above in Figure 3.3c. Once each sensor and actuator had been attached in the CoM and registered in the system, the evaluation was done through an HTTP client which sent requests to the Smart Gateway. In the evaluation, the HTTP-client carries out the same operations as those in the Web UI since it also sends requests to the same API in the Smart Gateways. But it was chosen so as to carry out multiple operations in a batch and test the system with a higher stress. Moreover, the Web UI is not the only client that can manage the system, since any other system can be integrated into the proposed system through the Smart Gateways API, so the evaluation also tests the system as it integrates with external applications. Figure 3.4 shows the case study proposed in the system evaluation.

We have used an Arduino Mega 2560 as the microcontroller for deploying the CoMs. Arduino is part of the open hardware microcontroller board family which is present in a large number of IoT projects due to its reduced price and its large community. A lot of IoT microcontrollers offer support for Arduino such as Libelium [125] and Particle [61], so the CoM could also be tested in other microcontrollers other than Arduino. Moreover, as the implementation is done in C/C++, it can be easily extrapolated to another platform. Four XBee Pro S2B modules configured with API mode with escaping and a baud rate of 19200bps have also been used to create a ZigBee mesh (one coordinator connected to the Smart Gateway and three end devices integrated in Arduino Mega). The Smart Gateway, the Web UI and the HTTP client have been installed on a Windows 7 OS PC with 8GB of RAM. The following subsections present the memory footprint, data transmission, communication delay and power consumption evaluations performed.



Figure 3.4: Smart Home case study

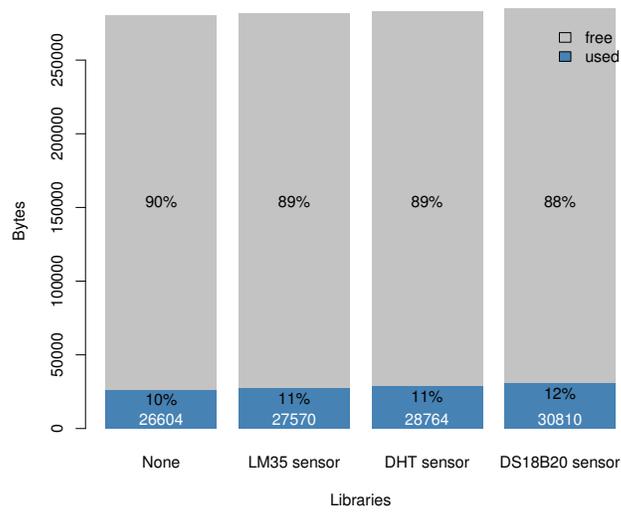
3.4.2 Memory Footprint

The memory footprint is one of the most critical parts of resource constrained devices, since their available memory is normally highly limited. In order to evaluate the memory consumption when new supported libraries are added to the CoM, an evaluation with three libraries of different sizes has been done. These libraries include an LM35 temperature sensor which does not require external libraries to work, a DHT temperature and humidity sensor with medium-sized external libraries and a DS18B20 temperature sensor which requires large-sized external libraries.

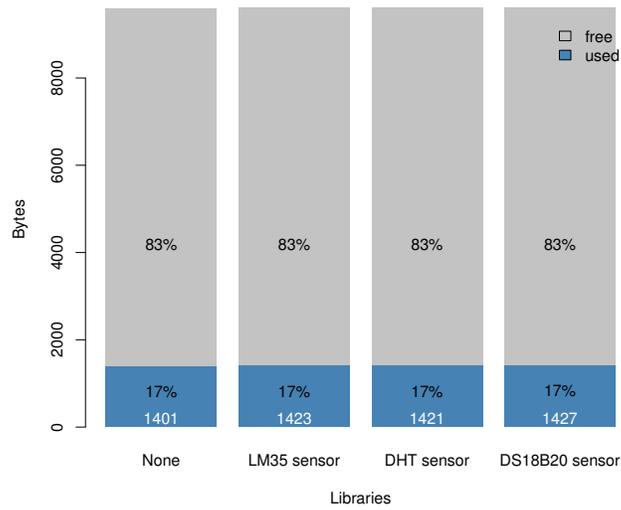
Figure 3.5a and Figure 3.5b show, respectively, the flash and SRAM footprints of the uploaded sources to the microcontroller by the Arduino IDE in the aforementioned scenarios. The evaluation demonstrates that the flash footprint in Arduino Mega 2560 varies from 1 to 2% with each library, depending on its requirements. The SRAM footprint barely changes with each new supported library. Therefore, the CoM runs comfortably on the Arduino Mega platform and enables support for around 45 to 90 types of libraries approximately.

3.4.3 Data Transmission

The data transmission is another crucial requirement in resource constrained devices, since the transferred data is restricted most of the time and large data transmissions require high-consumption.



a Flash-consumption



b SRAM consumption

Figure 3.5: Memory footprint with different library types

In fact, some configurations with ZigBee, with security and in API enabled mode can restrict data transmission by as much as 62 bytes. Moreover, the maximum transmission unit in IEEE 802.15.4 on which are based protocols such as 6LoWPAN and ZigBee, is established at 127 bytes.

Each data packet exchanged in the run-time CoAP resources control system is composed of the following components: CoAP header, token, URI-Path, and payload. The CoAP Header, as mentioned, is fixed at 4 bytes. Although the token size can vary between 0 to 8 bytes, in the current CoAP version, our implementation uses a fixed token of 2 bytes. One byte to indicate the token option and another to generate 255 different tokens which is sufficient for the proposed scenarios. The URI-Path depends on the URI provided in the Web UI by the users, however it is recommended that short URIs like *'/sen'* with 4 bytes are used. The URI-Path is included as an option in the CoAP packet, so a minimum of 1 byte is also added. Lastly, the payload size changes with the number of connectors used, the URI-Path and the operation being done (some operations do not require all the parameters). A payload of a creation operation with a URI-Path of 4 bytes, one connector with two bytes and temporally defined (`</send>rt="0";p="12";t="1"`) is formed by 24 bytes. Therefore, the packet size of one CoAP operation can be composed as $4 + 2 + 5 + 24 = 35$ bytes.

In general terms, the exchanged packet size in the operations offered by the system is fitted into only one data packet and only sent once. Therefore, it can be sent in the most resource constrained networks and it also avoids other mechanisms such as fragmentation, which overloads the network.

3.4.4 Power Consumption

The power consumption has been measured with a high accuracy digital multimeter which measures the circulating draw provided by the external batteries to the CoMs. In the previous tests, an HTTP client executes a testbed with a set of creations, updates and deletions with different resource types and behaviours. The power consumption of the Arduino Mega was measured, running the CoM without any operations and with an empty loop in order to contrast the results provided by the microcontroller consumption. The multimeter has a USB interface to upload data to the PC. Figure 3.6 shows the power consumption results.

The evaluation in Figure 3.6 shows an average consumption of the CoM of 103mA. That power consumption can be considered as high, because a normal battery of 2200mAh will be exhausted in approximately $2200/103 = 21.35$ hours. This is because the average consumption in Arduino Mega with an empty loop is 70mA which is more than two thirds of the total consumption of the CoM running. Note that this is the raw power consumption in Arduino Mega without taking into account any power management libraries. The idle current of Zigbee Pro S2B of 15mA is also included. Due to the high power consumption in Arduino Mega, other low-power microcontrollers like Moteino Mega will be considered in future tests.

The average operation consumption does not vary to any great extent with respect to the CoM consumption. Some outliers of between 114 and 116mA in all cases were found, which could be

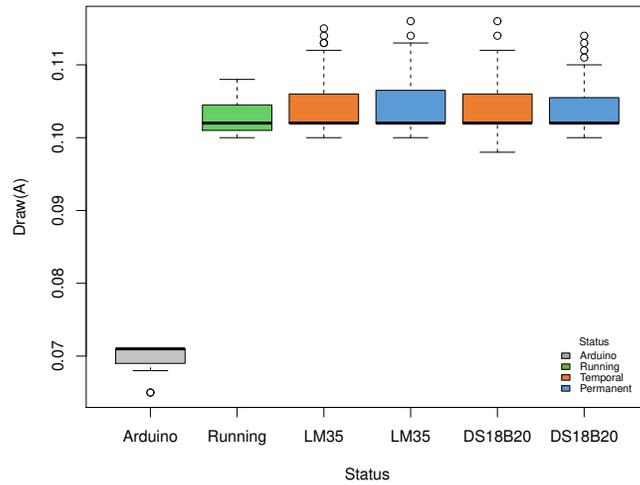


Figure 3.6: Power consumption with different behaviours

due to high communication rates, but the overall consumption is constant, close to 104mA in all scenarios. Therefore, the run-time CoAP resources control system does not cause a high-power consumption in the management operations with Arduino Mega.

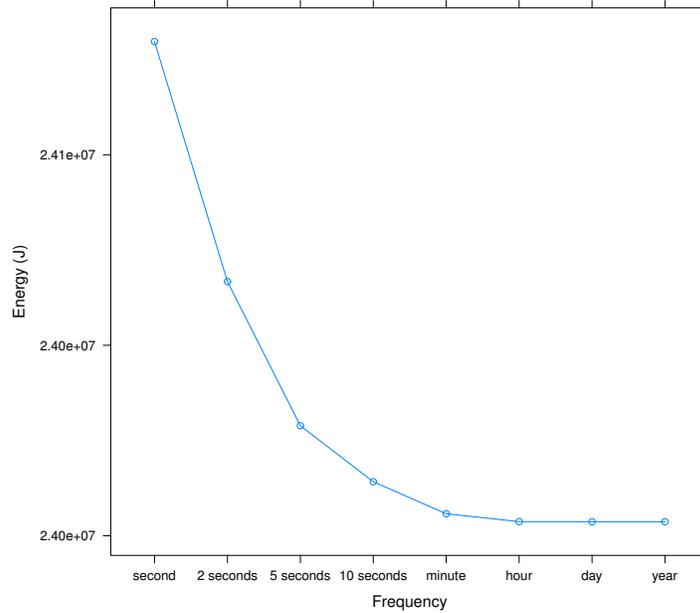


Figure 3.7: Energy simulation with several operation rates over a year

The total energy consumed by the microcontroller can be estimated using the following formula [126]:

$$E = \sum_{j \in \mathcal{M}} i_j \cdot v \cdot \Delta t_j \quad (3.1)$$

where \mathcal{M} is the set of operation modes of the microcontroller (active, idle, RX and TX); i_j is the current in state j ; v is the nominal voltage of the battery; and Δt_j is the time the microcontroller spent in operation mode j . Specifically, the XBee-PRO S2B radio transceiver used in the case study consumes an average of 54.5mA in read mode (RX), a peak 212.5mA in writing mode (TX), and 15mA in active mode. The consumption of the CoM in active mode is 103mA, including the ZigBee active mode based on our evaluation. Lastly, the battery voltage is 7.4V.

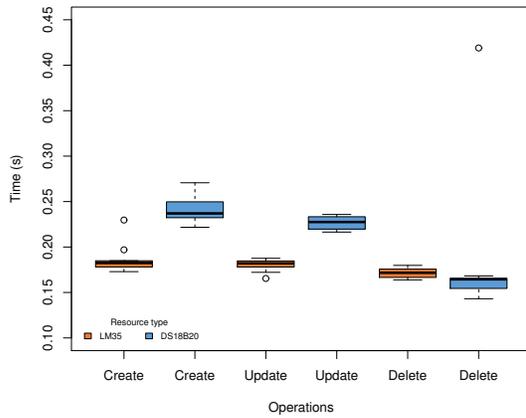
In order to estimate the energy consumption based on the number of requests over time, we make the following assumptions:

- baud rate obtained in evaluations of 15389.78bps.
- ZigBee API mode with escaping and no packet loss.
- CoM is always running, idle mode=0.
- other requests are ignored.
- 31 and 6 bytes for the CoAP requests and responses to the CoM respectively.
- 1 year simulation.

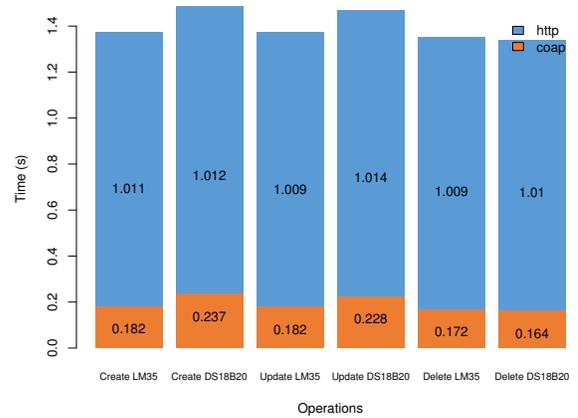
Figure 3.7 presents a simulation of the energy consumption with different operation rates over a year. Although, in order to work for long periods microcontrollers have to sleep to reduce the power consumption, the results show an exponential energy decrement with respect to the operation rate. However, the energy consumption remains high due to the microcontroller's consumption.

3.4.5 Communication Delay

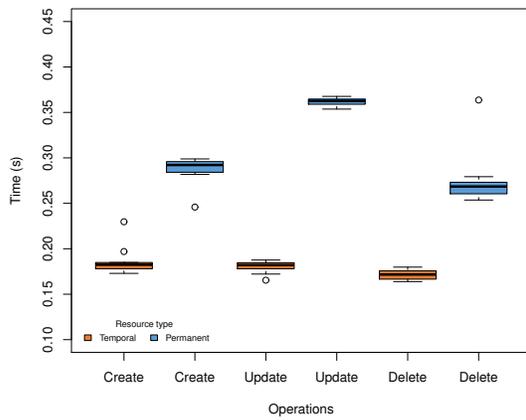
In order to measure the communication delay with the CoMs, a testbed with different types of physical sensors and behaviours was also designed. In each operation, the HTTP and CoAP response time of the creation, update and elimination operations from an HTTP client to the Smart Gateway was measured 10 times. The results are represented as box plot graphs in Figure 3.8 which shows the median, outliers and quartiles of each operation. LM35 and DS18B20 temperature sensors



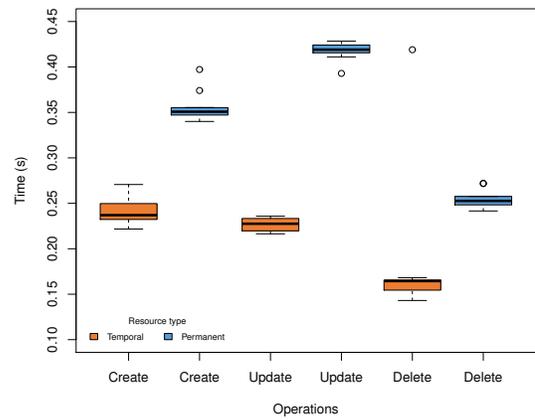
a CoAP response time with different operations



b Median response time with different operations



c CoAP response time in LM35 with different temporally



d CoAP response time in DS18B20 with different temporally

Figure 3.8: Communication delay measures

were chosen to evaluate the response time with sensors with different requirements. The temporal behaviour was also evaluated for each sensor type.

Figure 3.8a shows the CoAP response time in the management operations in the LM35 and DS18B20 sensors, respectively. The creation and update operations differ from each other by between 4 and 5 centiseconds. This is due to the delay in the instantiation of the libraries needed to register the DS18B20 sensor. Therefore, different sizes in the management operations only imply the consumption of some centiseconds. However, the deletion operation does not require the instantiation of new libraries, so the delay is smaller. In fact, the results show a slightly better result in the DS18B20 deletion.

In Figure 3.8b the median response time of HTTP and CoAP in each operation is shown. The main purpose of HTTP is to translate the HTTP request into CoAP, so resource types and behaviours do not affect it. This is the reason why the HTTP time only changes in milliseconds between operations and its time is not representative beyond measuring the overall response time in each operation. Nevertheless, the results reflect that the management operations can be done through the Web UI or an HTTP client in little more than 1 second in the deployment scenario, irrespective of the sensor type used.

As can be seen in Figure 3.8c and Figure 3.8d, the temporality affects the CoAP response time, to a greater extent, in both types of sensors. This is because when temporality is defined changes are persisted in the EEPROM memory and the storage delay is also present. Thus, the temporality not only protects the EEPROM memory but it also reduces the response time in the management operations.

3.4.6 Comparison with other proposals

As mentioned in Section 2.2 of the Chapter 2, most current systems deploy sensors and actuators in compile-time, or else, provide mechanisms to deploy them in run-time using standards suitable for computers such as USB or IEEE 1451 instead of microcontrollers. The latter entails a difficult task to achieve a comparison with other systems in the same scenarios due to the anatomy of the plug&play components, which are not suitable for resource constrained devices, and vice versa. Another solution could consist in modifying the compile-time deployment systems in order to provide the run-time deployment functionality, however this goes beyond the scope of this contribution.

To the best of our knowledge, only μ PnP provides a system that allows the deployment of sensors and actuators in run-time for resource constrained devices. However, μ PnP requires specific software and hardware components to compose the system architecture and they are not accessible to make a system comparison. Based on the evaluation provided in the μ PnP paper [25], the system

proposed in this Chapter provides a lower RAM and a higher Flash consumption. In the Smart Home case study the Flash consumption is not a drawback since there is about 90% free space with some libraries. The data transmission time shows an average time with the proposed scenario similar to our average time with the sensors and actuators tested and the temporality defined. The HTTP time is not included in the last comparison, but it also enables the system to be managed with a large variety of clients, as for example the Web UI. Lastly, in terms of energy consumption the proposed system provides a higher energy consumption, however the proposed system has been tested in a real scenario whilst the μ PnP presents a simulation with ideal behaviours.

4

Appdaptivity: An Internet of Things Device-Decoupled System for Portable Applications in Changing Contexts

This chapter presents the second contribution of this thesis, Appdaptivity, a device-decoupled system for the development of portable, personalised and adapted to changing contexts IoT applications. This contribution was developed during a research stay in the IDLab research group of the University of Ghent, Belgium. With Appdaptivity, application developers just need to focus on the application logic with personalised behaviours, rather than the low-level details of the IoT, which are abstracted the by system. Moreover, these applications can be intuitively created and are portable and adapted to changing contexts. The chapter begins with the problem statement and research goals in Section 4.1. Appdaptivity, its requirements (4.2.1), the approach and design (4.2.2), its ways of deployment (4.2.3), and the components involved are presented in Section 4.2. The evaluation and the validation of Appdaptivity in a smart city use case, and the experiences of Appdaptivity in HomeLab, a smart home at Ghent University, are given in Section 4.4. Lastly, the chapter concludes with a discussion about the differences between Appdaptivity and some technologies used: CoAP and the CoAP++ framework.

4.1 Problem Statement and Research Goals.

The IoT is continuously expanding in an unprecedented way, changing the way we interact with each other and with the environment. Many companies have adopted this epoch-defining revolution to improve their business processes and manufacturing chains with the capabilities offered by the IoT. This revolution has also reached many other environments such as housing and buildings, healthcare, cities and farming, to name but a few [2, 127, 128]. However, most have resulted in vertical silo applications with established and predefined physical IoT devices. This makes the inclusion of new components at run-time difficult and goes against one of the IoT philosophies of having ready-to-use devices with minimal configuration. Although some systems provide discovery and joining mechanisms, forming ready-to-use devices, applications are still dependent on the physical devices and reconfiguration is usually required upon the inclusion or disruption of devices. Application logic should be abstracted from the underlying physical infrastructure adapted to changing contexts, automatically adapting the application logic to the underlying infrastructure changes.

Moreover, IoT devices are becoming more personalised. The advances in the IoT have contributed to the release of a wide range of devices for multiple areas. These devices apart from being shared between the community (e.g., a city CO₂ sensor), also belong to people (e.g., a door locker). The development of IoT applications should take this into account as part of their design, but without deviating from their main goal: the application logic. Otherwise, the development for highly personalised applications would represent a high effort.

In large IoT deployments, it may be a challenge to define the application logic. If the IoT deployment comprises multiple infrastructures, the application logic is normally installed and configured in each one of those scenarios, again making the expansion of the IoT difficult. If instead of having to configure each environment, these environments could be dynamically configured and be part of the IoT applications, the IoT applications would not have to address the challenges of having multiple infrastructures. This would enable the portability of applications for heterogeneous environments.

The application development itself can also represent an obstacle for the development of IoT applications. The IoT has been taken up by multiple kinds of users, both with and without expertise. This means that the spread of the IoT would be limited if the application development required the use of complex abstractions. To reach a wide variety of end users, these applications should be intuitively created, i.e., the development should be easy to do and not represent an obstacle for end users.

Finally, as it is known that IoT devices are subject to a set of limitations. The work should focus on devices with low capabilities, like Class 1 devices (≈ 10 KB RAM and ≈ 100 KB ROM) that can be considerably reduced with networking stack libraries and the application logic. Therefore, the solution should affect these resource-constrained devices as little as possible, otherwise it cannot be applied.

Our research aims to answer the following question based on the problems identified in the development of IoT applications:

- How can we intuitively define portable, adaptable, personalised and device-decoupled IoT applications?

4.2 Appdaptivity System

4.2.1 Requirements

Requirements establish the guidelines for the solution design and should be kept for the solution's life cycle. Based on the previous research questions and problem statement, the following requirements have been identified:

- Application development completely agnostic from physical devices. Application developers should focus on the application logic instead of considering the final devices that will be part of the system. Final device inclusion and subscription should adapt the logic of the targeted applications.
- Intuitive application development. The IoT is penetrating the consumer market with a large variety of solutions for multiple areas. A minimal device configuration and an intuitive application development are both key to spreading the solution.
- Portable application development. End users can be part of extremely dissimilar environments such as an entire building, a single smart-home and an embedded device. The solution should be able to be dynamically part of these environments with minimal configuration.
- Personalised applications. In some IoT deployments, users have rights to certain devices and cannot interact with any other (e.g., restricted areas). The system should be able to restrict the access to the underlying infrastructure when it is required.
- Adoption of current standards. The IoT requires the use of open standards, otherwise the solution will be taken into a vertical silo, increasing the IoT heterogeneity.
- Embedded solution to reach resource-constrained devices. IoT devices have serious limitations in term of processing, storage and power. The proposed solution should take into account these limitations and not affect their performance.

4.2.2 Approach and Design

As targeted in the aforementioned requirements, the application development should focus on the application logic instead of designing applications for specific devices. Appdaptivity enables the development of IoT applications independently of the physical infrastructure. Through a discovery process, the underlying physical infrastructure is part of the system and the application logic when it is available. Discovery can also be done in multiples scenarios thanks to the involvement of end users. In this case, discovery is dynamically done as a background process and the underlying physical infrastructure can be discovered in multiple contexts. Application logic only has to be done once and this discovery design enables the portability of these applications to multiple heterogeneous environments. Application portability is one of the key features of Appdaptivity.

An abstraction has been adopted to categorise the underlying physical infrastructure and the context changes for end users: the location. The location refers to a logical abstraction where a physical infrastructure

has been deployed, e.g., a room. Location enables the modelling of the real work as it has been conceived. All context changes that have happened in locations will be reflected in the application logic defined. This requires that the physical infrastructure has to support the location on their devices in order to correlate them with logical locations. Enabling the location on constrained devices can incur a loss of performance. Here, previous work has been exploited to force/encourage users to configure the location through a virtual resource [129], which does not impact on the resource-constrained devices' performance. Personalised applications are developed, exploiting previous work on managing access in resource-constrained environments [130], so that only authorised people can access devices. In the discovery process the underlying infrastructure is discovered which a target user has rights to, and the application logic will be fed with these updates. If a certain logic depends on a situation (e.g., temperature values), the application logic will not be activated until all the corresponding components have been received. Therefore, the application development can be globally done for all the personalised applications, since the application logic will be activated depending on the rights of each end user.

Appdaptivity has adopted a data flow programming model [39] to intuitively define user applications. In data flow programming, users define applications through a directed graph of nodes, modelling a flow of actions from which data should be taken. Nodes have a defined function which does not depend on other nodes, therefore they form highly portable and reusable components for the creation of applications. Moreover, Appdaptivity provides a large set of nodes to define user applications that can be created without having to program any lines of code. This mechanism is so non-technical users find it easier to use the system, which in turn facilitates the expansion of the proposed solution. Other frameworks provide some of the capabilities to develop device-decoupled, adapted changing contexts and personalised application. To the best of our knowledge this approach is the first to provide a framework for intuitively creating IoT applications that can be portable, personalised and adapted to changing contexts.

Appdaptivity comprises the Portability Core (PoCo), which is the component responsible for enabling the data flow programming in an accessible and transparent Web UI. The PoCo enables location-based application development where users can define their applications. The resulting applications can be dynamically portable to other environments. An adaption for diverse deployment-scenarios has been taken into account and the PoCo offers a variety of deployments for different use cases. The application logic comprises different behaviours in data flow programming, whereas user applications are a set of clients that receive and send data to the PoCo with the information originating from the system and the required information to activate the flows. Behaviours comprise the application logic of the resulting applications and will always be up-to-date with the underlying contexts. User applications are typically smartphone applications which display an up-to-date representation of the behaviours of the underlying physical infrastructure provided by the PoCo to end users.

IoT devices are another key part of the architecture, and the most restricted one, since it contains resource-constrained devices. For this reason, Appdaptivity has adopted CoAP as the application protocol to interact with final devices. In the scope of this contribution, there will be sensors to be monitored and actuators to be interacted with, therefore a lightweight standard like CoAP with accessible resources represents a

suitable protocol. All the wireless sensor networks with CoAP support that can be part of Appdaptivity will henceforth be referred to as the CoAP network. The CoAP network provides the sensors and actuators and device information which are accessible using CoAP to create IoT applications. The PoCo is responsible for interacting with the CoAP networks and enables the definition of data flows with the obtained CoAP resources that will form different behaviours in the user applications.

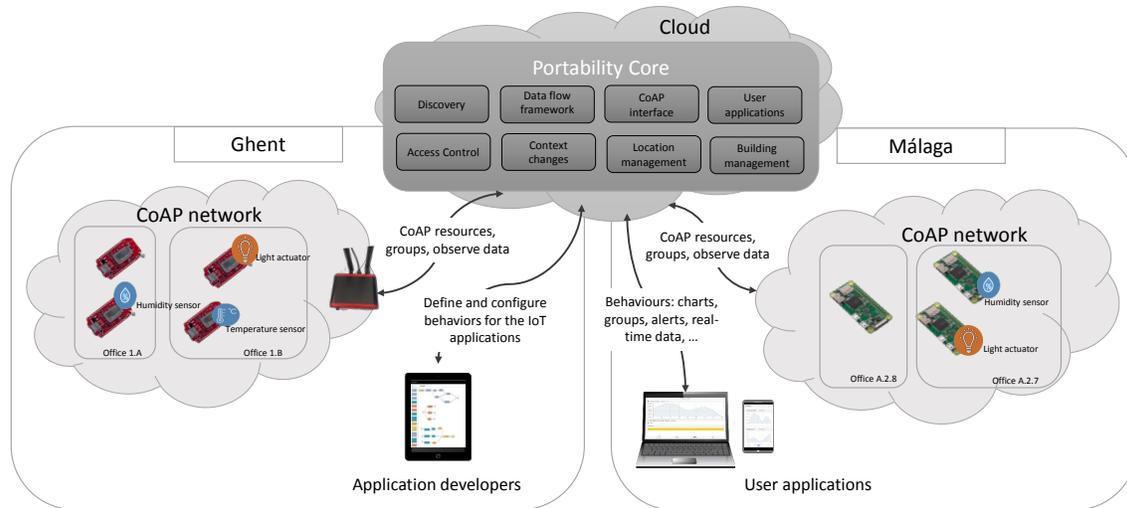


Figure 4.1: Appdaptivity architecture.

An overview of the Appdaptivity architecture is presented in Figure 4.1. In this case, the PoCo adopts a cloud deployment, however it can be deployed in other scenarios as we will see later. In real worldwide IoT deployments, CoAP networks along with their sensors and actuators are deployed in different physical locations, for example different cities, such as Malaga and Ghent. Appdaptivity handles the process of portability of the IoT applications to these environments while at the same time it allows the definition of custom behaviours with a unified and intuitive interface. Discovery is responsible for discovering the CoAP networks in different environments where the portability will be carried out. The behaviours comprise the functionalities that can be defined in the PoCo through its data flow framework and define the actions to be taken with the discovered CoAP networks. Behaviours are executed in the PoCo, however some tasks such as a device interaction can be carried out directly by user applications once the corresponding behaviour has been received. Examples of these behaviours are: the creation of charts with the average value of temperature sensors, alert monitoring based on the sensor data and CoAP groups, for interacting with them with just a unique CoAP request. In the last example, end users could directly interact with the group using CoAP. The CoAP interface in PoCo manages all the CoAP interactions in the system such as a group creation and an observation.

4.2.3 System Deployments

Depending on the use case and the scope of the application, IoT applications can vary from local use up to worldwide deployments. The portability and access of the resulting applications are different in each use case. The system architecture has been designed to adapt itself to a variety of use cases and enables three ways of deployment (Figure 4.2):

- cloud (3): a cloud deployment of the PoCo enables the portability of applications and the access from anywhere with an Internet connection, e.g., access control in an international company. In this case, the discovery of the local CoAP networks by necessity has to be performed by the user applications since the PoCo could be deployed in a different network than CoAP devices and the IP multicast for discovering the CoAP networks has to be performed in the local network. Resource Directory (RD) [131] and CoAP devices will have to be located in the local network and they will have to be IP-accessible in order to enable their discovery and interaction respectively. The RD is an entity launched by the CoRE group which holds information on other servers such as list of resources and their characteristics. RDs are mainly used to indirectly discover nodes when it is not possible or practical due to the given network's specific characteristics, in this case RDs provide the resources and end points contained in the CoAP network to the PoCo.
- local (2): in this case, user applications and the PoCo have been deployed in different devices. The PoCo will run on a dedicated server, but in the same network as the user applications, e.g., controlling the sensors and actuators of a building on behalf of the employers. The application portability can be done with the available CoAP networks in the local network. Discovering of RDs can be done by both user applications and the PoCo.
- embedded (1): user applications and the PoCo have been deployed in the same device. This enables a portable and embedded solution that can be applied to controlled areas, e.g., smart home controlled by the owners. CoAP networks are discovered by the PoCo. In this case, the portability is done in situ with the discovered CoAP networks.

4.2.4 CoAP Network

The CoAP network comprises a set of CoAP servers deployed in known locations that contains a set of resources with the actuators and sensors available for IoT applications. A CoAP server is an IoT device running a CoAP server that collects data and performs actuation in its resources. These CoAP servers have to be previously registered with the RD which enables the observation of the CoAP servers resources and end points by the PoCo. Apart from the sensors and actuators, CoAP servers should define the physical location to enable the user interaction in different locations. CoAP servers could require an extra resource for the location, and this can affect the performance of constrained devices. However, the location can be established through sensor virtualisation [129, 132], through which resources are defined in the gateways and it does not affect the performance of the resource constrained CoAP servers.

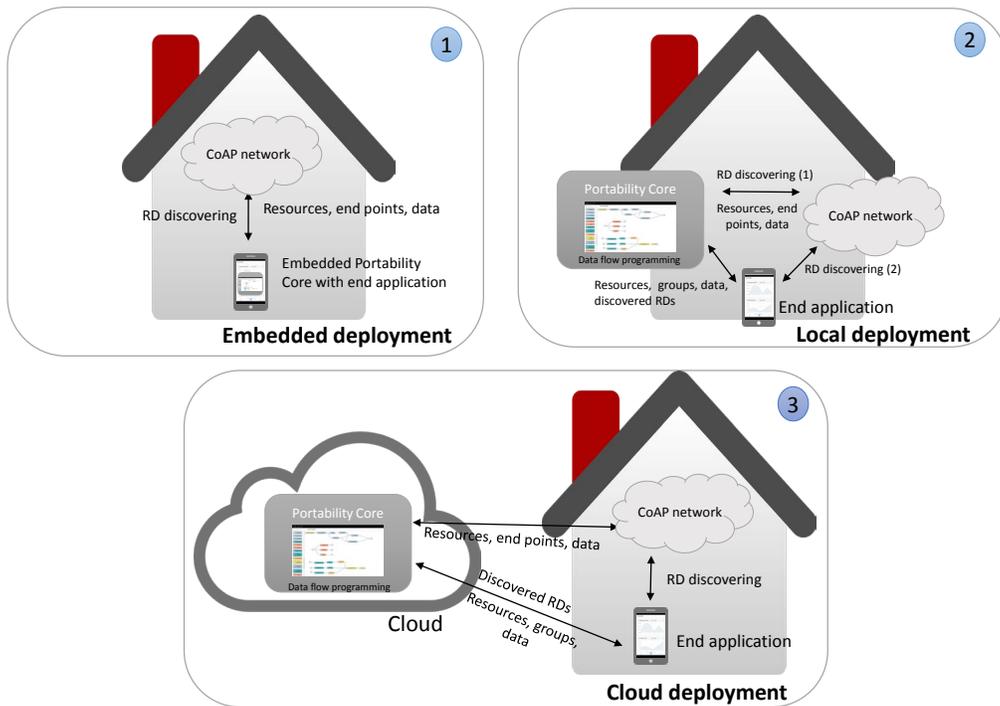


Figure 4.2: Possible deployment configurations in Appdaptivity.

Once a CoAP server has been started, it can discover the RDs in the network. In the case of an RD discovery, CoAP servers can register its resources in the RD through a POST request as defined in the standard. In the PoCo, an observation to the resources and end points of each discovered CoAP network is sent to maintain an up-to-date status of the CoAP networks. Therefore, the system is self-adaptive to network changes and user applications always contain up-to-date behaviours based on the current CoAP networks.

4.2.5 Portability Core (PoCo)

IoT applications are designed using the visual data flow framework provided by PoCo. These applications define the behaviours that will be rendered in the user applications without any configuration by end users, allowing the portability of them to multiple environments. The application developers only need to focus on high-level functionalities, rather than the implementation of low-level details of the underlying physical infrastructure. The service continuity can be a key challenge in applications with large IoT deployments since they can involve multiple heterogeneous networks. In that sense, the PoCo provides the mechanisms to maintain the service continuity in IoT applications independently the underlying physical networks used by the applications. Application can be portable to multiple environments and networks, thus applications are independent of the underlying physical devices and can adapt to the changing context without recompiling, keeping the service continuity of IoT applications in multiple scenarios.

The discovery process used in Appdaptivity follows the OMA LightweightM2M (OMA LWM2M) stan-

standard [133], which enables the association of sensors and actuators with an open specification, i.e., resources are identified and organised based on the rules established in the standard. The use of standards allows a unified identification of resources. Nevertheless, apart from the OMA LWM2M standard, other standards like the IPSO Application Framework can also be added and used in Appdaptivity. In case of the adoption of another standard, the specification of the location resource should be configured in Appdaptivity according to the new standard. Moreover, user applications will have to integrate the standard in their UIs in order to know the type of behaviours received and display them accordingly. Moreover, CoAP discovery can be enhanced for supporting semantic matchmaking via inference services and logic languages like the SWoT framework [134].

The work flows that can be configured in Appdaptivity are not only restricted to location-based flows, they can also dispatch the resources belonging to the discovered RDs. Appdaptivity enables two main flows: the location and the building flows. Both flows take as input the resources discovered and the locations of each end points. The location flow is responsible for enabling the behaviours defined in one physical location no matter what RDs have been found; whilst the building flow enables the behaviours defined with all the locations in the RD received, e.g., an entire building. End users can establish their location, or just get their available behaviours in the building based on the last RD received. In order to reduce the number of communications with the RDs and CoAP servers, flows are only activated once end users have established their locations or start the building flow. On the other hand, personalised behaviours are crucial when the number of sensors and actuators increases. This is more important in scenarios where sensors and actuators are restricted, e.g., access control in secured areas.

The Algorithm 1 shows an overall description of the update functionality in Appdaptivity. Once the IoT infrastructure is discovered, or there is any change in the environment, each active location in the location and building flow is checked to see if its corresponding IoT infrastructure has been changed. In the case that any flow has been changed, its behaviours affected will be updated and user applications will receive an up-to-date representation of them. When location or building flow are activated by user applications, they take the current status of the IoT infrastructure with its corresponding CoAP resources and will be up-to-date through this algorithm during its lifecycle.

4.2.5.1 Personalised Behaviours

In personalised behaviours, the communication between the PoCo and the CoAP networks uses the standard DTLS with client certificate authentication, and the communication between user applications and the PoCo uses SSL/TLS. Therefore, the communication channel on both sides can be encrypted, enabling user authentication, and location access control. Security and privacy problems in the IoT, such as applying security patches, physical attacks on the sensor, data leakage, side channel attacks and intrusive sensing are beyond the scope of this contribution. To establish a relationship between CoAP networks and physical spaces, the system makes use of the user location to find the nearby infrastructure for each user. The location can be established by end users through the Near Field Communication (NFC) technology. NFC is a short-range identification technology which provides a secure and easy communication and is widely available in mul-

Algorithm 1 Overview of the update algorithm in Appdaptivity

Data: Local or user applications discovery of the IoT Infrastructure
Result: Notifications to user applications

```

while true do
  IoTInfrastructure_ ← localDiscovery() || discoveryByUserApp()
  if changed(IoTInfrastructure_) then
    IoTInfrastructure ← IoTInfrastructure_
    if activeLocationFlows then
      | checkLocations(activeLocations, IoTInfrastructure)
    end
    if activeBuildingFlows then
      for building ← activeBuildingFlows do
        | activeLocations ← getLocation(building, IoTInfrastructure)
        | checkLocations(activeLocations, IoTInfrastructure)
        if newLocations(building, IoTInfrastructure) then
          for location ← newLocations do
            | resourcesLocation ← getResources(location, IoTInfrastructure)
            | behaviours ← activateBehaviours(resourcesLocation)
            | notifyBehavioursUserApps(location, behaviours)
          end
        end
      end
    end
  end
end
end

Procedure checkLocations(locations, IoTInfrastructure)
  for location ← activeLocations do
    | resourcesLocation ← getResources(location, IoTInfrastructure)
    | if changed(location, resourcesLocation) then
      | behaviours ← updateBehaviours(resourcesLocation)
      | notifyUpdateUserApps(location, behaviours)
    end
  end
end

```

multiple devices such as smartphones, thereby it can be used in multiple devices without external hardware requirements. Once a location has been established, the PoCo gets the resources that the user has rights to through a DTLS connection, with client certificate authentication for the CoAP network in the case of a secure configuration. These resources are converted into behaviours for the user applications in the Appdaptivity flows. In a non-secure configuration, the PoCo just returns the available resources of the established location. Tracking user presence in real-time is a hard problem and is also beyond the scope of this work.

The secure configuration is done through access control lists (ACL) by the physical infrastructure, in this case by the CoAP++ framework; refer to this paper [130] for more information. Encrypted and read-only NFC tags only tell Appdaptivity that a certain user is in a certain location, which can also be done with other identification technologies. Nevertheless, NFC is an identification technology present in many today smartphones, which provides capabilities for security identification. In the case that a user has rights over the

resources of this location; the user will receive the associated behaviours to these resources. Therefore, once the location has been established and the user has rights to that location, the authorisation is given indefinitely, i.e., if the user does not establish a new location he/she can move to other physical locations and will have the previously established logic location. Moreover, in the case that users want access to all the locations where he/she has rights in a CoAP network, the building flow give users the opportunity to interact with all their available locations, thus Appdaptivity is not location-strict.

Therefore, in the case that a smartphone application is targeted such as a user application, end users can establish their locations through their smartphones and the NFC tags located in each physical space. When the location has been established, user applications automatically obtain the resources belonging to the location in the terms defined in the behaviours in PoCo, e.g., list of values in real-time, charts, and so on. As discussed, this also requires a configured location resource in each CoAP server to know each server location, but it can be done without a performance lost with EC-IoT [129].

An example of the communication process to interact with a light actuator in a physical location is presented in Figure 4.3, specifically the system has been deployed in local mode with access control enabled. The communication starts with a connection between a user application and the PoCo (1). Once the connection has been established, the user application starts to discover RDs in the network (2). When a new RD has been discovered, it is sent to the PoCo (3) and an observation to the resources and end points is established (4). From this moment on, end users can establish their location with the NFC tags of each physical Personalised Behavioural space. In the interaction process, the location 'Room 1' has been established (5) and sent to the PoCo (6). From this moment, the PoCo starts the location flow in the established location (7) and the user application starts to receive the behaviours defined such as charts, groups and lists of resources allowed (8). Lastly, the end user decides to interact with a light actuator received (9) and a CoAP PUT request is sent directly to the CoAP server with the value selected by the user on the smartphone application.

4.2.5.2 PoCo Components

The available nodes in the PoCo for data flow programming with which users can define application behaviours are known as PoCo components. PoCo components have a set of inputs and outputs and a given behaviour, e.g., a URI filter component which filters the list of resources received based on the URI information provided in the component. There are some components that need a certain input, as for example, a group component waits for a set of resources to create a new group of these resources. Others can receive inputs from multiple components, such as the component in charge of sending all generated messages to the user applications. Components can be connected with each other in the correct way, forming work flows. Depending on their connections and the components involved, a large number of flows can be generated only with a few components. Therefore, with the same components or a subset of them and different component' connections, multiple behaviours can be obtained. To avoid functionality interruptions, there are necessary components and connections that are critical to the system (e.g., the discovering process) but there are also many optional components, such as filters which can be included in flows allowing different behaviours.

For the creation of a new component, an HTML file, with its style, and a JavaScript file, with its func-

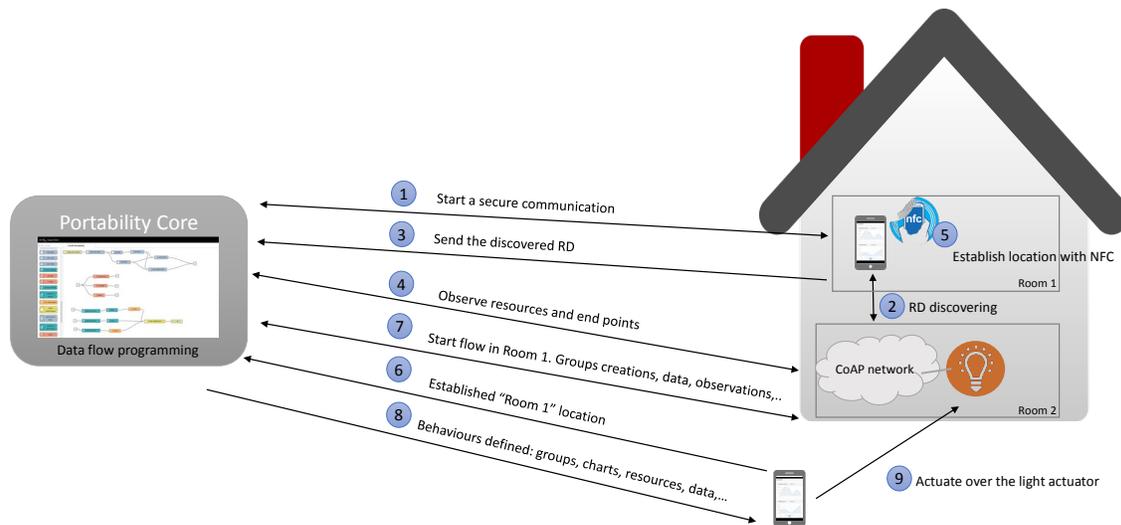


Figure 4.3: System interaction to actuate with a light actuator in a physical location.

tionality (e.g., filtering CoAP resources), only have to be defined; and then incorporating them to the system configuration. From that moment on, the component will be part of the palette enabling its inclusion in the development of IoT applications. Next, the developed components to define the PoCo behaviours are given:

- **Filters:** A set of components that filter the list of resources received based on the information established in the components or received in the messages. The filter (resource type, location or URI) information can be configured in each component. Filtering follows open standards for device interaction such as the OMA LWM2M standard [133], thereby it enables a common language in the integration with external systems and user applications.
- **Groups:** Group is a new entity which aims to address several CoAP resources as a group instead of addressing each resource individually. Groups offer a set of operations for the values of their resources and the possibility to observe them. End users can interact with a set of resources through a group-specific CoAP resource created by this component.
- **Group operations:** As mentioned above, groups enable a set of operations that can be applied on them. Among these operations are included the minimum, maximum, average and real-time list of resources values. This component makes an observation to the group received with the operation selected and returns asynchronously the information received.
- **Observations:** Resources can also be observed individually without a group association. This component enables the observation in the resources received and return the data obtained asynchronously.
- **Alerts:** Sometimes end users can be interested in receiving warnings when something changes in the environment, e.g., a door has been opened. This component checks the data received from the

observations and the group operations components and sends an alert message to user applications when the data matches the filters established.

- **Actuation trigger:** The reaction to conditions established in the environment is addressed by this component. Once an alert has been received based on the established conditions, this component triggers the target resources with the value established in the component to be actuated by the Send request component.
- **Charts:** This component adds extra information to the data received in order to inform user applications that data should be rendered such as a chart. Otherwise, data is shown without a rendered component.
- **Function:** Although the goal of Appdaptivity is not to program source code except visually, this component enables adding source code for specific tasks. For instance, a data mining algorithm can be added for data analysis.
- **CoAP requests:** Once resources or groups have been received in the user applications, they can be requested directly without a PoCo interaction. However, some user applications cannot provide a CoAP support to interact directly with resources or groups. In this case they can send a message to the PoCo to make the requests on behalf of the user application. This component makes a request for the resources or groups received and returns back the operation result.
- **Access control:** Lastly, this component provides access control to create personalised behaviours. Although this component can be mandatory in most uses cases, sometimes the access control is not necessary, e.g., public sensors or collaborative actuators. If this component is deployed, the access control is enabled in the system, otherwise it is disabled.

Monitoring applications are not the only application logic that can be created in Appdaptivity. Actuation trigger and Alerts components enable the definition of automatic actuation and configurable notifications. For instance, a user can define the actuation with some devices (e.g., windows) when there is change in the environment (e.g, high temperature). Users can also receive notifications when the underlying infrastructure matches a configurable criteria. This application logic allows automatic behaviours and actuation without any user interaction. Thereby, the main goals of the IoT, sense and actuate over the physical world [2] have been included. Furthermore, in the cases where a specific logic is necessary, the Function component enables the inclusion of specific source code like a data mining algorithm in the JavaScript programming language. As can the rest of the components in Appdaptivity, the Function component can be added in run-time and the application logic will be adapted with it.

When behaviours become complex, programming can become a tedious task due to the number of PoCo components and necessary data flows to define them. For that reason, behaviours can be reused creating a subflow component. Subflow can be defined by selecting a group of PoCo components and selecting the option “create a subflow”. Once a subflow has been defined, a new PoCo component with the selected behaviours will be available in the system. Therefore, subflow components enable the programming reuse in addition to reducing the complexity of large flows and facilitating programming for non-expert users.

Although the possible PoCo configurations are large, the configuration presented in Figure 4.4 covers most of the behaviours presented above. The top flow is responsible for the discovery of the IoT infrastructure, thanks to the observation of the end points (RD EP component) and resources (RD RES component). The next components allow it to get the end point locations and enable the location and building flows. Note that this flow can be grouped into a subflow to simplify the development. The middle block contains, from top to bottom, temperature, humidity and light filters respectively. Lastly, three different behaviours have been created in the bottom flow: an average chart of temperature, an alert of humidity, and a group of lights. As can be seen, Appdaptivity provides an intuitive way to program IoT applications with just a few components.

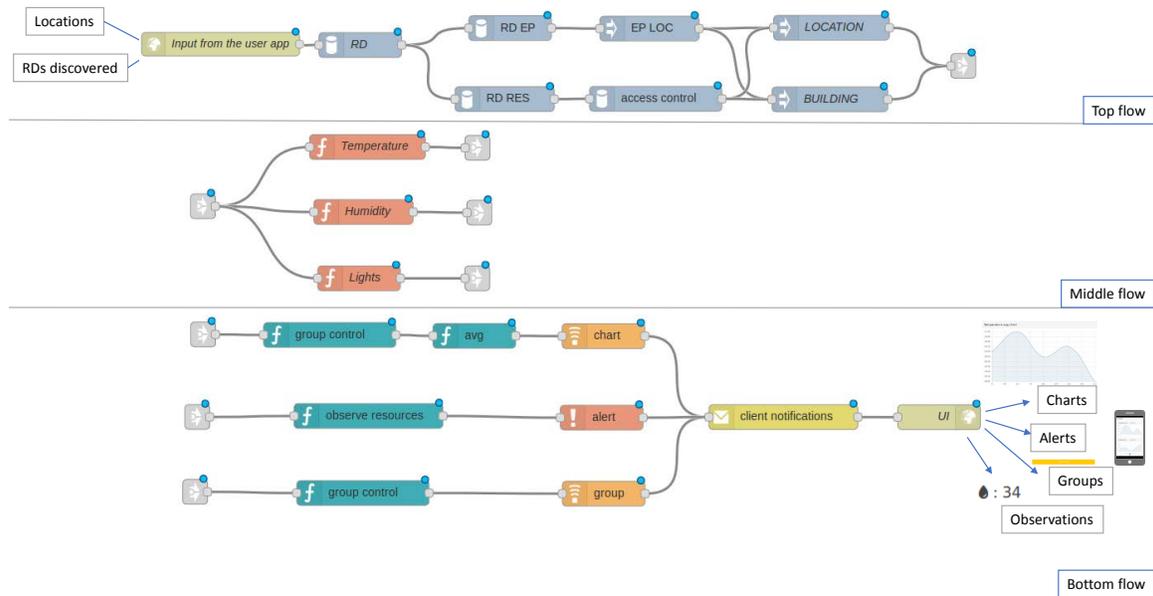


Figure 4.4: PoCo configuration with building and location flows, access control, and different behaviours.

4.2.6 User Applications

The Appdaptivity architecture has been designed to be abstracted from user applications, in such a way that any application can be integrated with Appdaptivity using its communication means and understanding its data formats. Hence, in addition to user applications created from scratch such as smartphone, web or desktop, Appdaptivity can also be integrated into other platforms or applications through its API. As discussed, behaviours are defined in the PoCo and user applications display the information and data received to end users in the way that they have been configured. Taking into account the system goals, user applications should provide the mechanisms to access and interact with the underlying physical devices without the efforts of programming and configuring. This goal has been the key philosophy in the design of user applications, therefore configuration and behaviours are defined in the PoCo, whereas user applications are only responsi-

ble for displaying the behaviours defined, discovering RDs when the system requires it and establishing the users' locations, which do not require any configuration. Consider, for example, a smartphone application launched as a user application in Appdaptivity, the only configuration required is installing the application if the application logic has previously been defined. Suppose a smart building, the administrator could create the application logic associated with the building infrastructure. Then, workers only have to install the application to start using the smart building functionalities.

Once the locations or the building flow have been established and started respectively, user applications start receiving the behaviours defined in the PoCo configuration, their corresponding data and the changes during the execution of the application. Figure 4.5 shows screenshots of a user application with an established location. Concretely, from top to bottom, Figure 4.5a displays an average temperature chart, a humidity average real-time value, and a group of light actuators for the interaction with. Figure 4.5b displays a pop-up window to manage a group of light actuators. The left button of the pop-up will turn on all lights belonging to the selected group once it has been pressed, whereas the center one will turn off all lights. User applications always contain up-to-date behaviours based on the state of the CoAP networks, and so in the case of a new or shutdown CoAP server deployment, user applications will be updated by the PoCo.

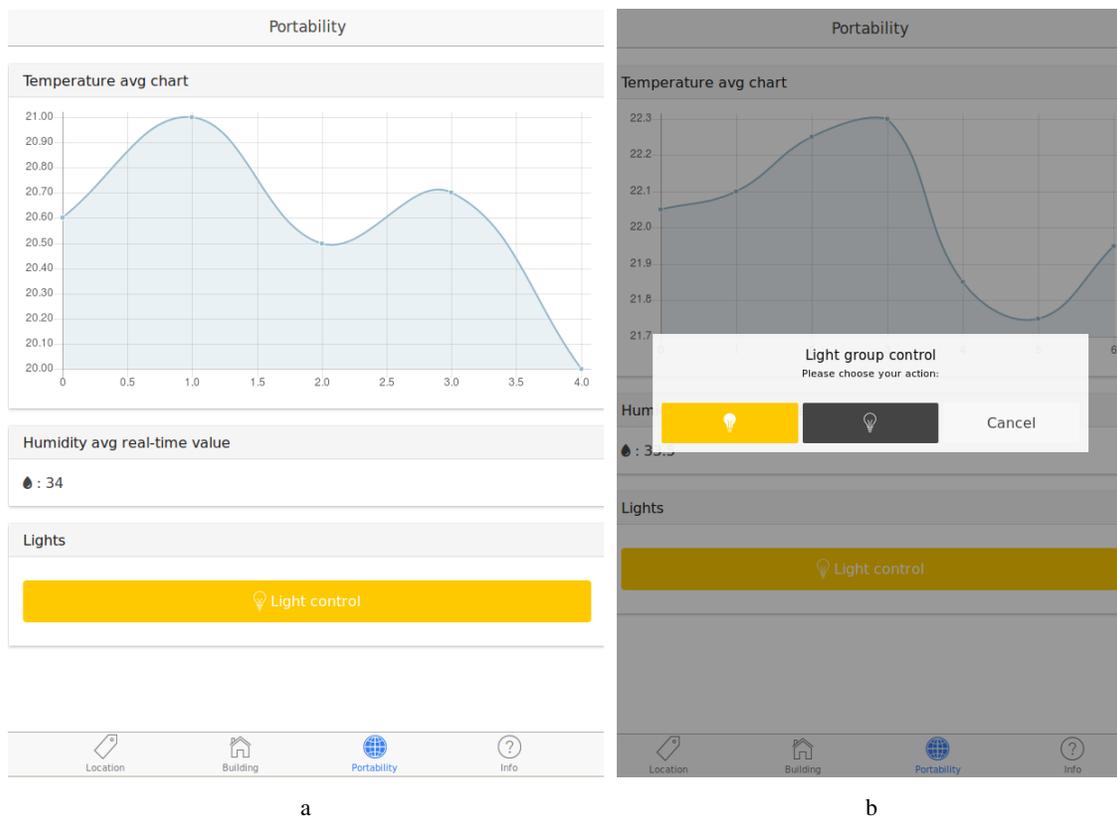


Figure 4.5: User application screenshots with some defined behaviours defined in the PoCo. (a) A chart, real-time values and a group of lights; (b) Pop-up window for interact with a light group.

The RD discovering can also be done by user applications, especially in cloud deployments where the PoCo cannot discover RDs by itself. It is important then that the discovery service takes into account the network changes, as for example, the connection to another Wi-Fi network, since RDs could be deployed in such networks. Lastly, although optionally, user applications can interact directly with the resources received with a CoAP client, otherwise they can use the request service available in the PoCo.

4.3 Implementation

4.3.1 PoCo and Interconnection

PoCo has been implemented as an extension of Node-RED [41], an open-source visual tool for data flow programming with a Web interface. Specifically, a set of nodes (PoCo components, communication and discovery) has been developed to be part of the Node-RED palette to address the requirements offered by the system. Node-RED comprises many components to define IoT applications thanks to the CoAP, WebSockets, GPIOs and many other available nodes. However, the components needed to discover CoAP networks; manage and observe CoAP groups and resources; provide secure CoAP communications; enable the location and building flows; and the logic for having an up-to-date representation of the behaviours defined in a transparent way for application developers, were not available in Node-RED. This is precisely the added value of Appdaptivity with regard to Node-RED, which enables the definition of portable, personalised, device-decoupled and adapted to changing contexts IoT applications. In addition, many other nodes (PoCo components) such as filters and visualisation components have been developed to facilitate the development of IoT applications. Each developed node has a specific function, such as the creation of a group of resources. The connection of these nodes comprises the behaviours that can be defined in Appdaptivity, as shown in Figure 4.4.

The communication between user applications and the PoCo has to be able to provide asynchronous communications since both components can send information asynchronously. Information can be dispatched from the PoCo to user applications continuously in groups or resources observations and sporadically when the CoAP network changes. User applications can also send asynchronous data with the RD discovered process and the location establishment. As a result, the communication channel has also to be bidirectional. Historically, applications that need bidirectional communication require different TCP connections, one for sending information and a new one for the reception. This entails the maintenance of multiple connections, mapping between each other and the abuse of HTTP polling for updates. WebSockets [135] was conceived as an alternative to HTTP polling providing two-way communication in a single TCP connection. In fact, WebSockets connections are done in the same ports as HTTP, nevertheless WebSockets and HTTP can only understand themselves in the handshake. Header and latency are usually smaller in WebSockets, thereby it is considered as a real-time communication channel. Moreover, WebSockets has also been integrated by default in most web browsers, the Web, and a large set of clients, so it opens the door to the integration of a large variety of clients and applications. Node-RED nodes have been developed in the PoCo to have a Websockets channel. The communication between user applications and the PoCo is encrypted leveraging the TLS/SSL

support in WebSockets. Therefore, the main interconnections in the system, the interconnection between the PoCo and user applications, and the PoCo and CoAP networks, are done through WebSockets and CoAP respectively.

The data format is another key challenge in the development of applications, since bizarre and random data formats carry out into large efforts of development, code difficult to maintain and a door opened to multiple issues. We have therefore chosen JSON—an open-standard and de facto format in a large amount of applications and development frameworks—as message format. A semantic has been defined for the message definition. All messages exchanged from PoCo to user applications contain an operation type and most of them, a location, which indicates the operation has to be done in user applications and the location where the event has taken place, respectively. Optional values can also be included, for example, a message status, the resource type of a resource, the value of a request and information for rendering components. An example of a message exchanged between the PoCo and user applications is presented in Figure 4.6, specifically an alert message of the change of a light actuator. The message is an alert operation in the location Room 1, indicated in the fields `op` and `location` respectively. Other fields indicate the message status and the resource type and URI of the component. Messages are sent once new behaviours have been created for user applications, with the observed data from resources and operation results.

```
{ op: 'alert',  
  msg: 'Status changed on component',  
  rt: 'urn:oma:lwm2m:ext:3311',  
  uri: 'coaps://[1234::7]:5684/3311/0/5850',  
  location: 'Room 1'  
}
```

Figure 4.6: Alert message sent from the PoCo to a user application.

4.3.2 User Applications

Appdaptivity does not focus on a certain user application, but it paves the way to the integration of them into the system. To integrate external end users with Appdaptivity, an application should be defined with a WebSockets communication, which should understand the API defined. Moreover, the application should provide the capabilities to visualise the behaviours defined (e.g., real-time charts) and enable end users to interact with the system (e.g., group of actuators). Lastly, NFC reading capabilities will be required to change the location. To validate Appdaptivity we have created a smartphone user application. Smartphones have revolutionised the way in which we interact with each other and we use applications. Moreover, they are commonly used nowadays, and through them the release of a smartphone can be quickly extended amongst end users. To avoid the programming efforts in programming smartphone applications for different OSs, we chose a multi-platform framework for developing smartphone applications called Ionic [136]. The Ionic framework enables the application programming using web technologies such as HTML and JavaScript. It

is based on AngularJS, a well-known framework for creating web applications. Once applications have been implemented in Ionic, they can be compiled for the target mobile OS such as Android and IOS. This reduces the development effort since only one source code is maintained.

A developed smartphone application enables end users to switch locations using the NFC technology and render the behaviours as defined in the PoCo (e.g., charts, group of actuators and real-time values). Apart from the location flow, end users can also activate the building flow obtaining all the behaviours defined in the building where they are. Furthermore, the smartphone application provides support for network discovery required in Appdaptivity cloud deployments. Figure 4.5 shows some screenshots of the resulting application running some behaviours defined in the PoCo such as charts and group of lights once a location has been established. Although the application is intended to be installed in multiple mobile OSs, currently the application only offers support for the tested OS: Android.

In certain situations, user applications have to perform a network discovery, like, for example, the RD discovery in a cloud deployment. The RD can be discovered through IPv6 multicast using its defined interface. However, during the development of a smartphone app as a user application in Appdaptivity, we realised that IPv6 multicast was not working properly in some devices, especially in smartphones. In this case, we extended the Appdaptivity with multicast Domain Name System (mDNS), a zero-configuration service—it does not require manual configuration nor special servers—which shares similarities with DNS and resolves hostnames to IP addresses. mDNS is supported by Android and IOS OSs, and the open-source implementation known as avahi, which has become the de facto standard implementation in Linux. In fact, the mDNS support in Appdaptivity uses the avahi implementation to publish a service on an established IP where clients can discover the RD in those cases where IPv6 multicast does not work properly. Smartphone-based user applications can make use of this service in order to reduce the issues using IPv6 multicast as in our case.

4.3.3 CoAP Network

CoAP networks have been deployed using the CoAP++ framework [137], a CoAP framework which enables the definition of CoAP applications with extended capabilities such as support for group communication, ACLs, observe operation and conditional observe developed by the IDLab research group. The CoAP++ framework allows a run-time definition of ACLs that is used to manage which user can access which resources. This allows a custom control, as for example, only enable the data access of the resource (GET method) or the interaction with (PUT method). The ACL can also support cipher suites for DTLS secured communication, including the certificate and public key and virtual resources in the CoAP server. An example of resource virtualisation is the location resource which typically is a virtualised resource that is hosted in the gateway, and not on the device itself. The communication process usually involves high-requirements in terms of communication, processing and memory footprint, which are not available in resource-constrained equipment, such as class 1 devices. The CoAP++ framework can be deployed in more powerful devices such as Raspberry Pi or BeagleBone and allows the use of pre-shared keys in order to avoid using public key operations in DTLS communications and reducing the requirements for performance-constrained environments [138].

The group communication was presented as an RFC in [139], nevertheless the CoAP group and the observe specification required to observe groups, have not been defined to work together [140]. This means that if a client needs an up-to-date representation of the resources belonging to a group, it has to continuously pull the group or observe the resources individually. In both cases, the client loses the benefits of the observe option and the benefits of the group, respectively. In the CoAP++ framework, the observe option and groups have been integrated and enable the possibility to apply different operations on groups of resources (e.g., average, minimum, maximum and list of values). Although group operations can be optional in some use cases, like, for example, actuating environments, in other environments such as continuous monitoring of set of resources, the group communication in the CoAP++ framework provides an optimal solution. Any standard-based CoAP server and RD implementation could be used in Appdaptivity, however some optional functionalities not available in other implementations such as the group management and access control available in the CoAP++ framework could be required in some behaviours.

4.4 Results and Discussion

In this Section an evaluation of Appdaptivity is presented. To evaluate the performance of Appdaptivity we have chosen different test scenarios. On the one hand, CoAP servers in the CoAP++ framework together with a lightweight user application without a graphical user interface have been created to evaluate Appdaptivity with a large number of clients and CoAP devices. This scenario enables the inclusion of multiple user applications and devices to the CoAP networks without the need to use multiple hardware devices, thereby facilitating the realisation of performance tests. On the other hand, a real deployment using a smartphone user application and a CoAP network with physical devices including sensors and actuators has been deployed as use case of an emerging IoT area. Moreover, one use of Appdaptivity is illustrated in a smart home at the Ghent University. The performance tests have been done in a 4GB Xubuntu OS virtual machine running over a 8 GB Windows 7. Note that this virtualisation could affect the evaluation done, especially in the real deployment. However, a virtualised environment was chosen for both the development and the evaluation processes in Appdaptivity. Due to its capabilities of replication, snapshot and isolation from the host OS.

4.4.1 Underlying IoT Infrastructure

In the tests, the total time until the resources and its locations from the CoAP network are obtained has been measured. This process includes the resources and end point collection from observation of the RD discovered, and the location collection from each end point obtained. Tests have been done with access control enabled (thereby with secure communication) and disabled in order to evaluate Appdaptivity in the different environments that it can be deployed in. Each CoAP server used in the CoAP network has 4 associated resources: temperature and humidity simulated sensors, a light simulated actuator and a location resource to obtain its physical location. Figure 4.7 shows average response times obtained from the performance tests done. As can be seen, the discovery time with access control enabled varies a little, whereas in the secure communication the discovery time increases more notably with respect to the number of CoAP servers.

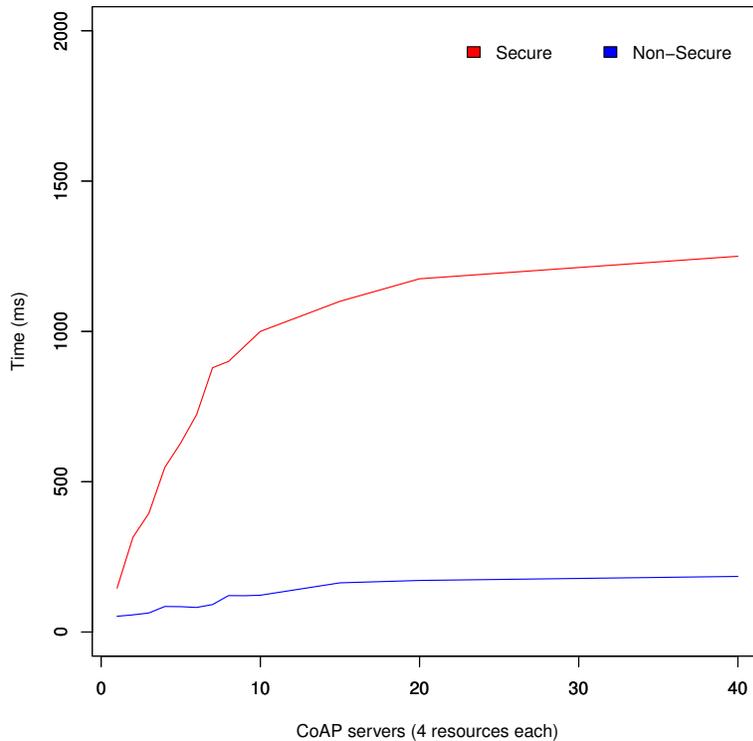


Figure 4.7: Location discovery response time with respect to the CoAP end points discovered.

Secure communications use DTLS which implies higher delays than UDP due to the necessary steps to secure the channel. However, Appdaptivity can manage these CoAP networks without large delays, and therefore it can be applied in large IoT deployments. In this test, it was also intended to evaluate the requirements of portable application development, thanks to the portability of IoT infrastructure (done thanks to the discovery) and personalised applications (secure configuration).

Deploying complex behaviours which involve a large number of PoCo components can reduce the scalability of the system. For that reason, the location discovery response time has once again been measured with 7 CoAP servers (4 resources each) and a different number of PoCo components. Figure 4.8 shows the results obtained with respect to a different number of PoCo components. In the same chart, the location discovery response time obtained in Figure 4.7, is also included which has fewer less than 20 PoCo components. As can be seen, the number of deployed PoCo components does not affect, to a great degree, to the scalability of Appdaptivity.

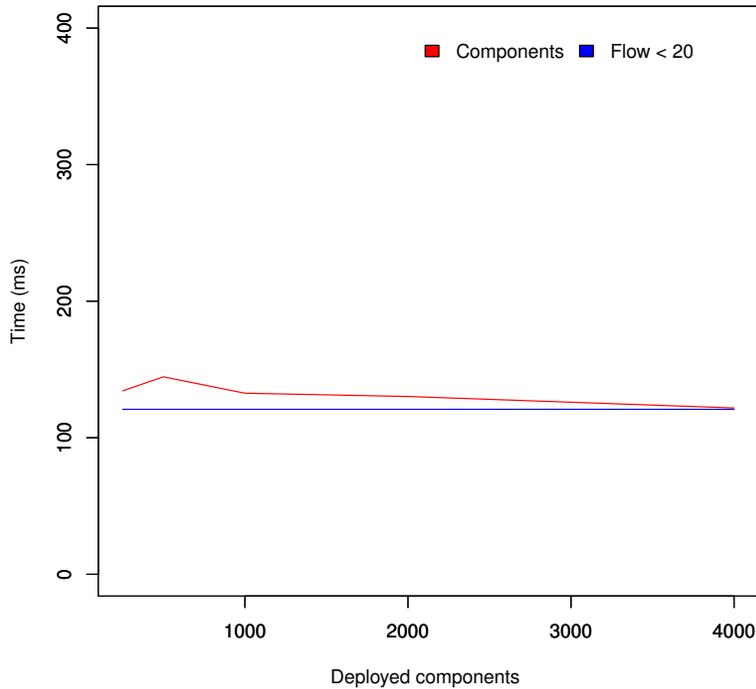


Figure 4.8: Location discovery response time with respect to the PoCo components deployed.

4.4.2 User Applications

The number of connected users that Appdaptivity can handle can limit the possibilities of applying Appdaptivity in the IoT. In this test, we will evaluate the number of user applications that can interact concurrently with Appdaptivity. The tests have measured the response time since user applications send an RD discovered until the information of the behaviours defined (list of resources) is received with multiple user applications concurrently. Therefore, the location discovery time measured above is also included. The CoAP network comprises 8 CoAP end points along with their 4 corresponding resources as presented above. In the same line of the connection with the CoAP network, Appdaptivity also guarantees a secure and private communication with user applications through the fundamental protocol in the Internet transport security: TLS/SSL. In the comparison, the scenarios with a secure and non-secure communication channel have been included. Figure 4.9 displays the average response time, and the 95% confidence intervals in the aforementioned scenario. The variation between a secure and non-secure channel varies slightly until it remains stable with 80 user applications concurrently. Confidence intervals show a higher difference due to the use of secure communications. We have taken 80 users as the upper bound as an example of workers of a medium-size building. Therefore, in this case it is recommendable to activate the secure channel at all times. Appdaptivity can comfortably handle a large number of user applications. The worst case time may seem a bit high, however it only has

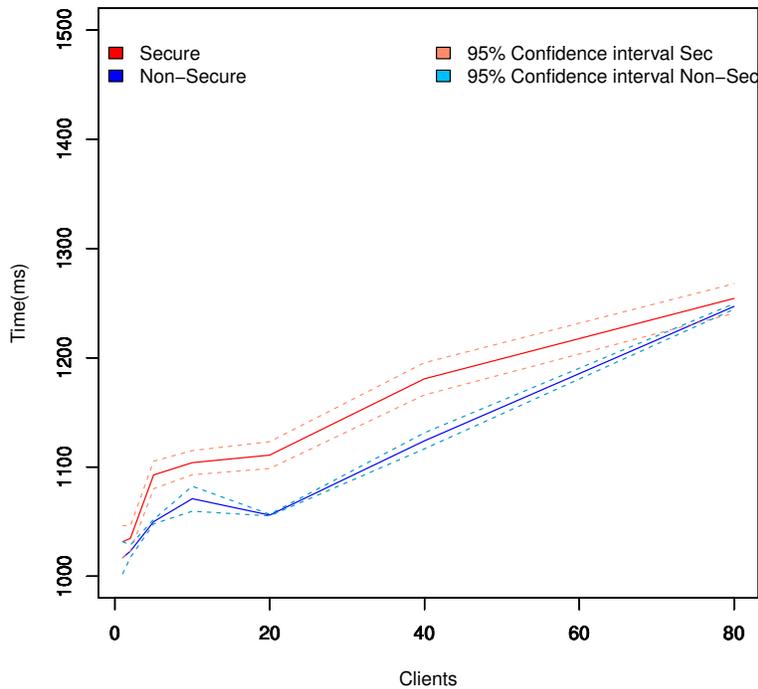


Figure 4.9: Response time with different numbers of user applications and the behaviour list of resources.

to be done once and includes the communication response from all the clients, the discovery of the CoAP network and a CoAP request for each one of the 8 CoAP servers to get their locations.

4.4.3 Smart Cities: Portability of IoT Services in Different Districts

The design of personalised applications represents one the main potentials of Appdaptivity. The data flow programming approach and the range of behaviours that can be defined with the PoCo components enable the definition of applications in a large set of areas in the widely expanded IoT field. To demonstrate this potential, we define a real use case and translate it into Appdaptivity. The goal of this use case is to demonstrate how IoT applications can be created intuitively and agnostically from the underlying IoT infrastructure with Appdaptivity. In addition, the portability of the solution and personalised behaviours using resource-constrained devices.

Consider a smart city with two districts. Both districts have different IoT services to allow the population to interact with, however the IoT services and networks can vary over the time. Therefore, to maintain the service continuity to the population without the need to install multiple applications or write specific and non-reusable code it is necessary to have a system that can cope with these challenges, and this is where

Appdaptivity can be applied.

As mentioned above, the districts contain different IoT services. Let us suppose that in district D1, a light actuator is activated automatically when a presence is detected by a passive infrared sensor (PIR) or an illuminance sensor. The population could also query the luminosity measurements. On the other hand, in district D2, end users can control light actuators in a single way or individually, and see the average temperature in a chart.

As the use case has different locations, the resulting data flow in the translation to Appdaptivity should differentiate the behaviours for each district. In the D1 district, the luminosity control is managed automatically through an Actuation trigger component which receives the list of lights to interact with and an alert once a presence has been detected or the illuminance is lower than a given limit (e.g., 30%). This component sends the list of lights to trigger the send request component which is responsible for making these requests. The population can query the illuminance sensors directly in their applications through a direct connection of the illuminance sensors to the user applications. In the D2 district, the population obtains a chart of average temperature. Lastly, the lights can be controlled directly, sending the light resources to the user applications, and controlled in a group-way with the creation of a group. Figure 4.10 displays the resulting Appdaptivity flow from the smart city use case presented. The mandatory components to enable the discovery of the underlying IoT and exchange information with the user applications have been grouped into two subflow components (Discovery and Output), so that the rest of components are PoCo components to define the use case behaviours. The development with just 20 components, shows the potential of Appdaptivity for the intuitive development of an IoT use case.

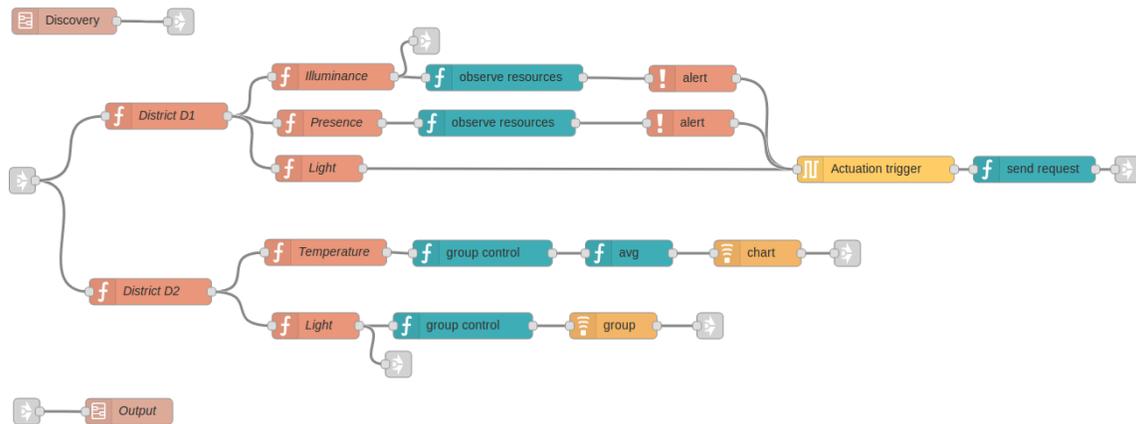


Figure 4.10: Smart-city use case translated into an Appdaptivity flow.

6LoWPAN has been chosen as the communication medium to deploy the smart city use case. 6LoWPAN enables the transmission of IPv6 packets over constrained networks, thereby providing an IPv6 address for each device that will be IP-accessible through the Internet. The CoAP network comprises four Zolertia RE-Mote (Zolertia RE-Mote platform: <https://github.com/Zolertia/Resources/wiki/RE-Mote>) devices, two for each district. The Zolertia RE-Mote is a hardware development platform which

provides support for 2.4-GHz and 863–950-MHz IEEE 802.15.4 and ZigBee compliant radios with low consumption. Zolertia RE-Mote also provides support for the well-known open-source OS for embedded devices, Contiki, and intrinsically for its 6LoWPAN implementation. Erbium, the Contiki REST engine and the CoAP implementation have been used to deploy the CoAP servers into Zolertia RE-Motes. As stated, Appdaptivity is decoupled from devices, thus Zolertia RE-Mote and Contiki have been used to validate the system but other devices and CoAP frameworks like Intel Edison and Californium can also be used with Appdaptivity. The fragmentation and reassembly mechanisms needed in 6LoWPAN to enable the communication with IPv6 networks are done by edge routers using RPL. We have chosen the Zolertia Orion router (Zolertia Orion Ethernet: <https://github.com/Zolertia/Resources/wiki/Orion>), an IPv4/IPv6 and 6LoWPAN routing device with Ethernet interface and 2.4 GHz and 863–950 MHz IEEE 802.15.4 radio support. The CETIC 6LBR (CETIC 6LBR: <https://github.com/cetic/6lbr/wiki>) border router solution has been installed into the Orion routers (one per district) to transparently enable users to communicate between IPv6 and 6LoWPAN networks. CETIC 6LBR also enables a Web UI to manage the router configuration and see its connected devices. A standard digital PIR sensor, the TSL2561 lux sensor with I2C connection, two DHT22 temperature and humidity digital sensors, and four standard digital LEDs were used as sensors and actuators in the use case. The experiments were performed in two of our laboratories (one per District), in the Research Building Ada Byron, University of Málaga. In each laboratory a NFC tag was placed to change the location. The use case was active for one hour, with default max-age in the observe operation, thus generating approximately 300 CoAP messages.

The CoAP++ framework has been used to allocate the RD, where the Zolertia RE-Motes register their resources and end points and Appdaptivity observes it to get the status of the CoAP networks. Appdaptivity has been deployed in local model along with CoAP++ framework in a 4GB Xubuntu OS virtual machine running over a 8GB Windows 7. The border routers and the PC are connected to a switch router by Ethernet and Wi-Fi respectively. Finally, the user application has been installed on an Android smartphone which is also connected by Wi-Fi to the switch router and can change the district with the NFC tags located in each one. Figure 4.11 shows an overview of the smart city use case proposed. As mentioned before, each district has a border router with a 6LoWPAN network. End users can change the location with the NFC tags located in each district independently of the networks without interrupting the service continuity of Appdaptivity.

Figure 4.12a,b display the RAM and Flash usage in the Zolertia RE-Motes and Orion routers respectively. This information has been obtained from the Contiki OS once the sketches have been uploaded to the devices. As can be seen, the Zolertia Re-Motes have 90% free space of Flash and 67% of RAM, allowing the creation of a large variety of new resources and reducing the power consumption thanks to the lightweight sketches. This test corroborates with the requirements of an embedded solution to reach resource-constrained devices, and the adoption of a current standard (CoAP). On the other hand, the Orion router needs more requirements: 75% of free space in Flash and 6% in RAM, however their sketches do not suffer many changes and they are usually connected without batteries. The power consumption evaluation resulted in an average 42 mA in the Zolertia motes which decreases the consumption more than two times of other hardware development platforms using 802.15.4 radios, as described in Chapter 3. The power consumption of the Zolertia Orion

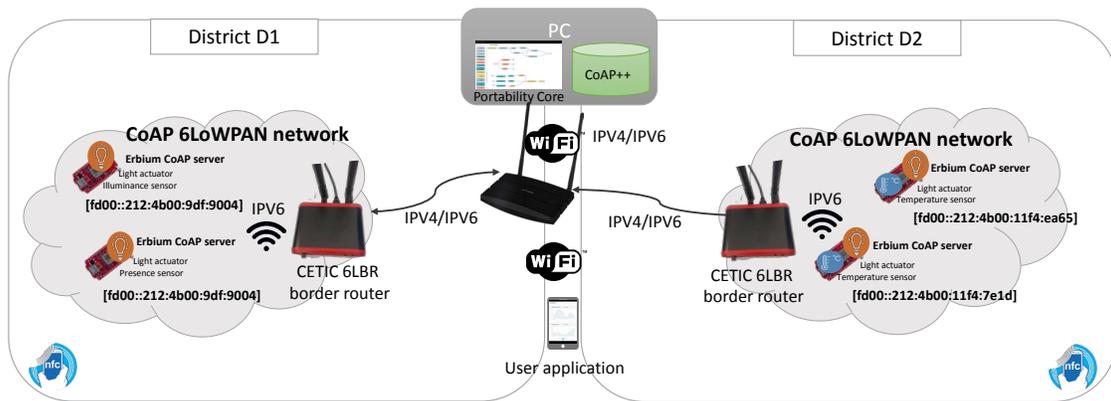


Figure 4.11: Smart-city use case deployment scenario.

routers has not been measured since they require large power capabilities with the use of Ethernet and it is assumed that they are powered without batteries. Finally, the discovery process in Appdaptivity takes an average of 286.4 ms in a non-secure mode.

4.4.4 Appdaptivity in HomeLab

Appdaptivity has also been evaluated in a real scenario. In this video [141] you can find a demo tutorial of Appdaptivity for its use in HomeLab. HomeLab [142] is a standalone house at the Ghent University, designed as a test environment for IoT services and smart living. In the video you can see charts from power meter and temperature sensors in different rooms changed using the building flow. Moreover, it shows how the lights can be controlled with the inclusion of a new button component at run-time, showing the corresponding flows in the PoCo.

4.5 Differences between CoAP, the CoAP++ Framework and Appdaptivity

To illustrate, consider the following scenario. A person wishes to create an application to manage and monitor certain conditions at his/her workplace or home. One option would be to deploy CoAP servers together with the relative sensors and actuators. These sensors and actuators could be acquired for direct use from lock-in vendors, but it is clear that this creates a vertical silo in the IoT. Therefore, resource-constrained devices are installed to work with CoAP, e.g., Zolertia Re-Motes. The low-level details for accessing these sensors and actuators in the Erbium framework would then have to be programmed. This would allow the interaction with these sensors and actuators through CoAP using, for example the well-known CoAP web client Copper (Cu). However, other functionalities may also be required, such as an HTTP-CoAP proxy, an RD, or being able to create groups of resources and conditional observations, and therefore the person decides to use the CoAP++

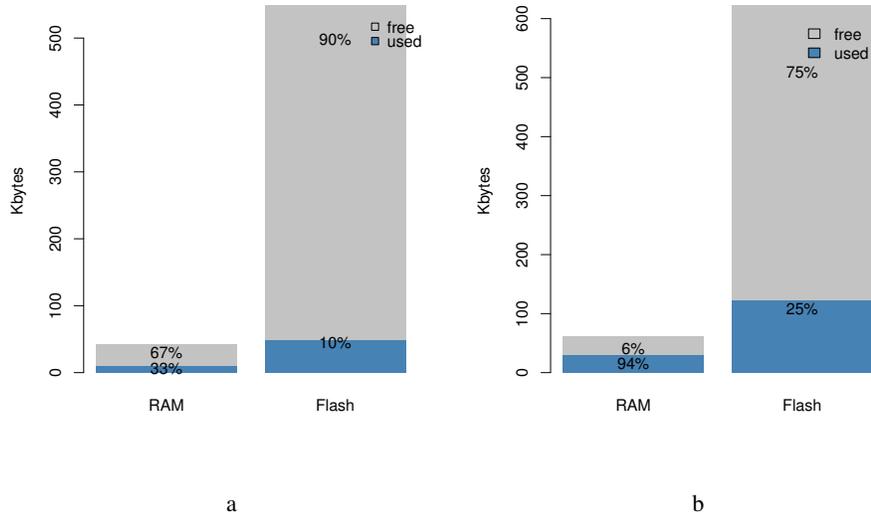


Figure 4.12: RAM and Flash memory usage in the Zolertia RE-Motes and the Zolertia Orion router. (a) Memory average usage in the sketches of the Zolertia RE-Motes; (b) Memory usage in the sketch of the Zolertia Orion Router.

framework, and installs it on a Raspberry Pi. Again, he/she can interact with the physical infrastructure with extended functionalities and has the possibility to use any HTTP client to interact with the CoAP network, e.g., a web browser. Nevertheless, in this case the person wishes to use a smartphone application to do the same tasks. They therefore decide to use the CoAP framework Californium and the Android platform to develop one. At this stage the application logic has to be programmed. Automatic tasks can be created, which open the windows when the measurements of the CO₂ sensors reach above a set level, or a button to control all the lighting, and charts and alerts set to show a given space's conditions, such as temperature and humidity. Note that until this point Appdaptivity has not been used nor required. Now, once the system has been deployed correctly, using open standards, the person decides to add the control of another room with new sensors and actuators. The common situation in this case is that any specific functionalities, programmed in the existing devices will necessitate having to reprogram and re-compile the application to incorporate the new devices. Alternatively, the application may be able to automatically discover new CoAP servers together with the sensors and actuators of the new room, but usually an application with these new components would have to be created. In these cases, device-coupled applications are created. Nevertheless, the ideal would be to have an application that automatically discovers the new devices in the CoAP networks and adapts the application logic. This is where Appdaptivity comes into its own.

Consider that the illumination of a given space can be controlled. The group communication of the CoAP++ framework can be used to create a group to manage this. In the case that there are new lights or

changes to them, Appdaptivity will update the corresponding behaviours, and also update their corresponding groups to ensure an up-to-date configuration. Furthermore, the rest of the behaviours that depend on these changes will also be updated, e.g., the trigger component that checks presence sensors to actuate over these lights. With Appdaptivity, if n components are controlled and a new CoAP server with the same component is deployed, then $n+1$ components can be controlled automatically. This can be extrapolated to other behaviours, e.g., if statistics about the conditions of some sensors are being displayed, each change in these sensors will be detected by Appdaptivity and the associated configuration will be updated. Appdaptivity will also enable the portability of the IoT applications to new CoAP networks in the future. In the case of personalised applications, the challenges are greater. In this case, a smartphone application can be installed and will always count on an up-to-date representation of the CoAP networks, with the result that the applications are adapted to the context changes, which makes them personalised and portable. Therefore, the aim of Appdaptivity is to complement CoAP and the IoT in the development of personalised, portable and adapted to changing context IoT applications.

5

An Internet of Things and Cloud Computing Integration

In this chapter, the λ -CoAP architecture is presented. This architecture proposes an integration between the IoT and cloud computing with edge computing capabilities. The chapter starts with an analysis of the integration components to compose the integration (Section 5.2). The problem statement and research goals of the λ -CoAP architecture are given in Section 5.1. The analysis of the integration components, which establish the guidelines to define the λ -CoAP architecture are given in Section 5.3, and its implementation in Section 5.4. Then, the architecture is evaluated in Section 5.5. Lastly, this chapter summarises some challenges and open research issues in Section 5.6.

5.1 Problem Statement and Research Goals

The IoT is intended to generate large amounts of data, which will keep on growing in the coming years [143]. The limitations of its devices make addressing current paradigms such as big data or deep learning very complicated, as they require large capacities in terms of processing and storage. These paradigms are designed to extract knowledge and generate actions over raw IoT data. In the last few years, the integration of the IoT with disruptive technologies such as cloud computing has provided the capabilities needed in the IoT to address these paradigms [2]. Nevertheless, this has resulted in an increase of latency in IoT communications in some situations (e.g., an actuation after cloud computing processing) and a dependency on many factors such as availability, routing and networks, which entail the use of external services.

Certain IoT applications, such as structural health monitoring [144] enable damage to be detected just by monitoring multiple, simple points of infrastructures. Higher-level concepts can be extracted from lower-level ones, which in addition can help define the higher-level concepts. This process is known as deep learning [145]. Deep learning is fed by multiple sources and large amounts of data from the extraction of complex patterns. This paradigm can require high capacity in terms of processing and storage that can be achieved with cloud computing. However an extreme event such as an earthquake or unanticipated blast loading would require timely action whilst reducing the latency to a minimum. A new paradigm known as edge computing [10], is intended to reduce the response time in IoT communications and reduce the upload bandwidth to the clouds, moving the computation to the edge of the networks. Edge computing can be interchangeable with fog computing, but edge computing is more focused on the things side [10]. For instance, the next generation of cars will generate 1GB of data per second and so real-time processing to make the right decisions will be required [10]. Although the next generation of mobile networks, 5G, is intended to increase the bandwidth and considerably reduce the latency, networks in the car use case like many others, e.g. airplanes, would represent a bottleneck for cloud-based computing. The adoption of an edge computing approach is required in these scenarios to optimise the latency of the communications and avoid network saturation. This does not mean that cloud computing will disappear from IoT applications, rather it will be used only when it is required.

The aim of this work is to provide an integration framework for the IoT and cloud computing to resolve the aforementioned problems. This framework will enable the development of IoT applications using both edge and cloud computing. The resulting applications will take advantage of real-time interactions at the edge and a long-term analysis on cloud computing.

5.2 Integration Components

The integration of the IoT and cloud computing started with an analysis of the possible components for that integration. The integration components were classified into two categories taking into account the needs of both the IoT and the cloud. On the one hand, the analysis addressed multidisciplinary cloud platforms to satisfy IoT limitations and to offer new business opportunities and more scalability. On the other hand, the analysis addressed the comparison of several IoT middleware to abstract the underlying heterogeneous IoT

devices.

Cloud computing and IoT integration provides new storage, processing, scalability and networking capabilities which until now have been limited in the IoT due to its characteristics. Furthermore, new opportunities like complex analysis, data mining and real-time processing will be enabled in the IoT, hitherto unthinkable in this field. Finally, through the IoT Middleware, the IoT devices will have a lightweight and interoperable mechanism for communication with each other and with the cloud systems deployed. Figure 5.1 shows the integration components surveyed in this analysis.

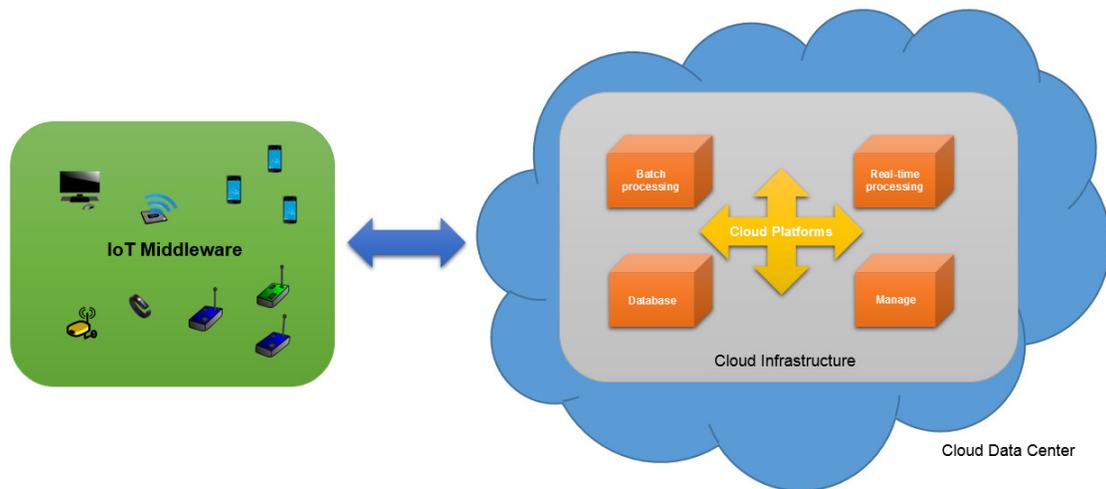


Figure 5.1: Cloud and IoT integration

5.2.1 Cloud Platforms

Recently and in the future, the number of users and data from the IoT will grow significantly as the number of connected devices increases [143]. For a long time, DBMS (Database Management Systems) have been used to store and access data in a great number of applications. Nevertheless, the growth in users and data means that a large number of DBMS are unsuitable. Hence, a platform which can tackle this is required in order to offer high scalability, storage and even processing. In this subsection, we summarise different platforms for storing, processing and accessing large amounts of heterogeneity data, which has become known as Big Data [143].

5.2.1.1 Batch Processing

Processing and analysing large amounts of data is one of the requirements that should be addressed in the integration of the IoT and cloud computing. Batch processing components are responsible for the execution of a series of jobs without manual intervention, allowing a greater distribution of them and high throughput.

An open-source framework to manage large amounts data is approached by Hadoop [146]. Hadoop comprises three main components: HDFS, MapReduce and YARN. HDFS is a distributed file system responsible for storing distributed and replicated data through a server cluster, providing reliability, scalability and high bandwidth. HDFS also balances the disk space usage between servers with a master/slave architecture. MapReduce [147] is also a master/slave framework used in Hadoop 1 to process and analyse large data sets, master job scheduling and cluster resource management on slaves. The coupling of a specific programming model like MapReduce with the cluster resource management, and concerns about centralised handling jobs, have encouraged the appearance of YARN [148] in Hadoop 2. YARN aims to decouple resource management functions from the programming model, incorporating a new layer which abstracts resource management at the same time as allowing new programming models. Presently, MapReduce is only in the form of an application running on top of YARN. However, the complexity of developing MapReduce programs has originated the appearance of new enriched systems that translate code into MapReduce, like Apache Hive [149], an SQL-like data warehouse for managing and querying large datasets, and Apache Pig [150], a platform for analysing large datasets with a high-level language for expressing programs.

Although Hadoop and other platforms are suitable for processing and analysing large amounts of data, in some situations such as machine learning algorithms and interactive ad-hoc query on large datasets, latency is considerable due to having to reload continuous data from a disk [151]. A good solution is a Map Reduce framework that keeps data in memory to reuse for multiple jobs and to reduce disk latency. Apache Spark [151], [152] is a fast and general framework which processes large amounts of data with low latency. Spark incorporates fault-tolerant and large data processing like Hadoop MapReduce, but also introduces a novel abstraction, the RDD (Resilient Distributed Dataset). RDDs are fault-tolerant collections of elements distributed across nodes than can be created, parallelising an existing collection or referencing a dataset in data store. The main advantages of RDDs are that they can persist in memory and can rebuild lost data without replication, executing 100x faster than Hadoop MapReduce in a logistic regression algorithm [152]. Moreover, Spark has a driver-worked architecture unique to each application, allowing isolation between applications, but data sharing applications must be written in external storage. Spark is also compatible with any Hadoop data store, such as HDFS, Apache Cassandra, Apache Hive, Apache Pig or Apache HBase, Chukwa and Amazon S3.

5.2.1.2 Distributed Databases

Focusing more on storing and querying large amounts of data, distributed databases offer mechanisms to enable some traditional features DBMS in a distributed environment with high availability and low latency.

Druid [153],[154] is an open-source distributed and column-oriented platform for storing and accessing large data sets in real-time. The need for storage and computing have originated platforms that store large amounts of data like Hadoop. However, these platforms do not guarantee how quickly data can be accessed and stored nor do they query performance, which can be required in some systems. Druid aims to resolve these problems and offers a platform which is suitable for applications that require low latency on ingestion and query data. The Druid architecture comprises two main components: historical nodes, which store and

query non-real-time data and real-time nodes, which ingest stream data and respond to queries with them. Moreover, other components such as the coordination nodes coordinate historical nodes to ensure that data is available and replicated, the brokers receive queries and forward them to real-time and historical nodes, and lastly, the indexer nodes ingest batch and real-time data in the system. The main functions of real-time nodes are to respond with real-time data and built segments for aged data that they send to the historical nodes. Historical nodes persist data in deep storage and download it in memory when the coordination node requires it. Druid also utilises Apache Zookeeper—a well-known Hadoop component for synchronising elements in a cluster—to manage the current cluster state, and MySQL for the maintenance of metadata about data segments.

Apache HBase [155] is another open-source distributed database suitable for random and real-time read/write large amounts of data. Apache HBase emerged from Google's Bigtable, a distributed storage system for structured data provided by Google File System. Specifically, Apache HBase uses HDFS for its storage system, and can be seen as the Hadoop Database. In HBase, data is stored in tables like traditional Relational Database Management Systems (RDBMS). However, HBase tables are composed by multiple rows, where each row has a set of columns which can store mixed and indeterminate key-value sets unlike traditional RDBMS. The tables can also be stored in different namespaces to restrict resources, and offer different security levels and region groups. The rows are identified and stored lexicographically by row key, allowing related rows, or rows which can be read together. The HBase architecture is HMaster/RegionServer, where the HMaster monitors and manages all RegionServers, and each RegionServer serves and manages the underlying regions. The regions are the basic components for distributing and providing availability and fault-tolerance in HBase tables. Initially, if a pre-splitting policy has not been established, a table is composed with only one region. Then, when the table rows grow, a set of policies can be established to split the regions in the RegionServers. If a table has been split into three regions, each region contains a sorted keyset chunk of the table. Regions' replication is provided by HDFS to enable fault-tolerant. The HMaster is responsible for assigning regions to RegionServers, balancing regions between RegionServers and restoring regions if a RegionServer goes down. Furthermore, HBase has allowed interaction with MapReduce, so MapReduce jobs can interact with the HBase data store. HBase provides a set of operations to modify and query tables, but if you want to use an SQL functionality and JDBC connector with HBase, you can use Apache Phoenix [156] or Cloudera Impala [157].

However, in some situations such as time series, the sorted key can lead to a major issue because when a table is split into two regions, the new writers will be written in the last region due to sorting, so a hot region will always be present. This problem is solved by OpenTSDB [158]. OpenTSDB uses an HBase table and a key format composed by a metric type—a string composed by a timestamp and a set of key/value—to store all stream data across various regions in a table. OpenTSDB has a broker architecture formed by the Time Series Daemons (TSD). The TSDs are the key components in OpenTSDB since they are responsible for managing and querying all data saved on HBase. A set of aggregation operations over the metrics as max, min or avg can be applied to OpenTSDB. All the features of HBase such as fault-tolerant, replication, split into tables, and so on, are used by OpenTSDB. Furthermore, a Web UI and other tools have been integrated to

query and graph all the time series stored. Nevertheless, a storage limitation is defined in OpenTSDB, since each metric and its types and values are assigned a unique id with 3 bytes, so $2^{24} - 1$ metrics and $2^{24} - 1$ types and values on each metric can be stored.

5.2.1.3 Real-time Processing

There are situations that require not only access to data, but also processing them in real-time, like decision making in critical systems or trending topics in social media.

Apache Storm [159] is a popular open-source distributed system for processing data streams in real-time. Storm contains a nimbus/supervisor architecture coordinated through Zookeeper, and is based on directed graphs, where the vertices represent computational components, and the edges represent the data flow among components. Spouts are the Storm components that receive data in the topology and are transmitted to Bolts which are responsible for processing the data and transmitting it to the next set of Bolts or data storage as required. Moreover, Storm offers semantic guarantees about data processing like ‘at least once’ or ‘at most once’ in addition to fault-tolerance. The main drawback is that it is not able to dynamically re-optimize at run-time between nodes, but this is considered to be future work [159].

An extension of Apache Spark for fault-tolerant streaming processing is known as Spark Streaming [160], [161]. Spark Streaming follows a different philosophy to Apache Storm. In Spark Streaming, the data received is stored in memory for a specific interval or window, and then is processed and stored in a Spark RDD. A D-Stream (Discretised stream) is the representation by times series of RDD, and it lets users manipulate them through various operators in real-time. Spark Streaming is properly integrated with Apache Spark, as it uses the same data representation, allowing reuse code and a hybrid architecture through the combining of batch with stream processing with the same data. The main drawback is the length of the sliding windows, as if they are too large, the real-time can disappear whereas if they are too small it can generate multiple RDDs. Furthermore, in most cases, stream data is received over the network, so to achieve data received fault-tolerance, Spark Streaming replicates data among the worker nodes.

5.2.1.4 Distributed Queues

Albeit in multiple systems, data is obtained from internal data stores, in some systems, especially in the IoT, data is dispatched by many devices or sub-systems. To deploy a large number of things in an IoT system requires large amounts of received data to be handled and eventually dispatched in real-time to the stakeholders. Distributed queues are summarised in this subsection in order to solve these problems.

Apache Kafka [162], [163] is a distributed messaging system that consumes and dispatches large amounts of data with low latency. Kafka is based on a publish/subscribe queue with ‘at least once’ and ‘at most once’ semantics that guarantees its suitability for streaming data and performing offline analysis. Topics are the stream of messages in Kafka, wherein producers can publish messages and consumers can subscribe to receive them. Load balancing and fault-tolerance is performed by partitions of the topics, where each topic can be divided into multiple partitions, and each partition can have multiple replicas. Communication

between producers and consumers to the server is through agnostic TCP, so it is available in many languages. When a message is sent by a producer to Kafka, it is stored on a disk in determined partitions and its replicas are consumed for a prefixed time. Moreover, Kafka provides group-consumers with a way of introducing load balancing ingestion data between consumers. However, Kafka does not contain a master node, it does however contain a set of brokers which locate the system partitions and are synchronised through Apache Zookeeper. The latter prevents concerns of master failures. Furthermore, producers and consumers can operate synchronously or asynchronously with batch data, but this originates a great dilemma between latency and throughput.

RabbitMQ [164], [165] is another open-source messaging queue that contains an Erlang-based implementation of AMQP (Advanced Message Queuing Protocol) v0.9.1 protocol. AMQP is an open standard for exchanging messages over TCP. RabbitMQ uses the Mnesia database, which is an in-memory persistent embedded database of Erlang for data persistence. RabbitMQ also uses brokers which are middleware applications of AMQP that receive messages from producers and sends them to other brokers or consumers. Brokers can be replicated for high availability, and send messages to other brokers in other clusters. When a message is sent by producers, it is received by a component of the brokers, called an exchange, which routes it, according to its routing rules—point-to-point, publish-subscribe, headers or multicast—and it is sent to a queue. When a consumer or consumer group receives a message, it is deleted from the queue. Moreover, RabbitMQ contains a pluggable system, where plugins can extend the deployment with new components like a Web UI management and monitoring. Therefore, RabbitMQ follows a different philosophy to Kafka, it incorporates message confirmation and more forms of communication than a topic. Producers can also obtain the message acknowledgment of consumers, so RabbitMQ can behave as the RPC protocol. RabbitMQ can also be more flexible than Kafka, however throughput and latency in Kafka is more desirable.

5.2.1.5 Management, Monitoring and Deployment

Many systems such as Apache Storm, Apache Hadoop or Apache Spark contain a Web UI for monitoring and visualising the cluster deployed. Nevertheless, in deployed clusters with multiple, deployed platforms, what is needed is a single platform that can deploy, monitor and manage all platforms.

In the Hadoop ecosystem, there is a platform for provisioning, monitoring and managing Hadoop clusters, called Apache Ambari [166]. Apache Ambari provides an open-source management Web UI, enabling system administrators to easily deploy new Hadoop services in the cluster, add new host nodes, manage existing services deployed and monitor the health and status of the Hadoop cluster. Ambari follows a master/agent architecture, where master contains a set of components responsible for monitoring the agent status and planning actions. Ambari agents send heartbeat events to master every few seconds with the agent status and actions and it receives actions to execute in the response. Moreover, Ambari utilises a database to store the system status, so in case of master failure, the status is rebuilt. Ambari uses Ganglia—a distributed monitoring high-performance system—for metrics collections, and Nagios—a monitoring system for IT infrastructure—for system alerts and sending emails. Furthermore, Ambari also integrates all capabilities in other systems through the Ambari REST APIs. Ambari is therefore a powerful tool for managing and moni-

toring Hadoop components with only one platform, and it can be easily integrated into other systems.

Tables 5.1 and 5.3 compare the different cloud platforms analysed. The components cover multiple cloud computing needs such as batch and stream processing, distributed storage and distributed queue which can be integrated independently or jointly. Most components follow a master/slave architecture and present a point of failure if the master goes down, but this can be mitigated with master replication or a broker architecture. Replication and load balancing are contained in most components, except load balancing in Apache Storm and Apache Ambari, and replication in Apache Ambari and Apache Spark that relies on the data store for this. Fault tolerance is one of the basic cloud computing features and is present in most components.

For batch processing, Apache Spark is recommended for low-latency applications with the same data, whereas Apache Hadoop is a fairly consolidated platform and is more integrated and compatible with external systems. Considering compatibility aspects, Spark Streaming is suitable if Apache Sparks is chosen as the batch processing component. Apache Storm does not necessarily require data storage as Spark Streaming does, and it contains more external integration, but nevertheless it does not contain load balancing. Distributed queuing, depends on the application, you may require confirmation messages and more options than a single topic, so RabbitMQ is the best solution in this case, but if you want a low-latency and throughput solution, Apache Kafka should be chosen.

Although HBase is an established solution, the need to design based on the data and the lack of real-time, mean that Druid will become the more promising solution. Apache Ambari has been designed to solve multi-platform deployment problems and is seemingly the solution for this. Apache HDFS is the most compatible distributed file system, whilst components like Java Apache Thrift and Hadoop Streaming allow multi-language service development. On the other hand, Apache Hadoop has the largest number of official companies using it, more than 170 companies, in comparison with Druid and OpenTSDB with 10 and 30 companies in 2016 respectively. Lastly, all components surveyed have an open-source license.

Table 5.1: Cloud Platforms comparison I

Cloud Platform	Finality	Architecture	Replication	Load balancing	Fault-tolerant	Memory	Programming Languages	Data store	External integration	Access	Documentation	Official companies that use it	Last update	License
Apache Hadoop MapReduce	Batch processing	Master/slave	Yes	Yes	Yes, if not master	Yes	Java, C++, Hadoop Streaming and REST API	HDFS		Web UI and CLI	***	+170	dec-15	Apache License Version 2.0
Apache Hadoop HDFS	Distributed file system	Master/slave	Yes	Yes	Yes, if not master	Yes, optionally	Java, Apache Thrift and REST API	HDFS		Web UI and CLI	***	+170	dec-15	Apache License Version 2.0
Druid	Real-time data store	Multiple nodes	Yes	Yes	Yes, if not coordinator	Yes, real-time nodes	Ruby, Python, R and Node.js	S3, HDFS, local mount and Cassandra	Hadoop, Storm and Kafka	CLI	**	+10	dec-15	GNU General Public License Version 2.0
Apache HBase	Distributed data store	HMaster/HServerRegions	Yes	Yes	Yes, if not Hmaster	Yes, optionally	Java, C/C++, Apache Thrift and REST API	HDFS, local filesystem	Apache Hadoop MapReduce	Web UI and CLI	***	+40	nov-15	Apache License Version 2.0
OpenTSDB	Distributed time series data store	Brokers	Yes	Yes	Yes, if not Hmaster	No	REST API, R Client, Erlang, Ruby, Go	HBase	RabbitMQ, Etsy Skyline and Apache Pig	Web UI and CLI	**	+30	nov-15	GNU LGPLv2.1+

Table 5.3: Cloud Platforms comparison II

Cloud Platform	Finality	Architecture	Replication	Load balancing	Fault-tolerant	Memory	Programming Languages	Data store	External integration	Access	Documentation	Official companies that use it	Last update	License
Apache Spark	Batch processing	Driver/worked	No in memory, implicit in storage	Yes	By application, if driver not fail.	Yes	Java, Python and Scala	HDFS, S3, Cassandra and HBase		Web UI and CLI	***	+80	nov-15	Apache License Version 2.0
Apache Kafka	Distributed queue	Brokers	Yes	Yes	Yes	No	Several languages		Several systems	CLI	**	+70	N/A	Apache License Version 2.0
Apache Storm	Stream processing	Nimbus/supervisor	Yes	No	Yes, if not nimbus	Yes	Java and Apache Thrift		Several systems	Web UI and CLI	**	+70	nov-15	Apache License Version 2.0
Apache Spark Streaming	Stream processing	Driver/worked	Yes, at reception	Yes	By application, if driver not fail.	Yes	Java, Python, Scala	HDFS, S3, Cassandra, Hbase		Web UI and CLI	***	N/A	nov-15	Apache License Version 2.0
RabbitMQ	Distributed queue	Brokers	Yes, with plugin	Yes	Yes	Yes	Several languages		MQTT and STOMP	Web UI and CLI	**	N/A	dec-15	Mozilla Public License Version 1.1
Apache Ambari	Provisioning and monitoring and managing clusters	Master/agents	No	No	Yes, if not master	No	REST API	HDFS	Several Hadoop Components	Web UI, CLI	**	N/A	dec-15	Apache License Version 2.0

5.2.2 Middleware for IoT

In IoT deployments there are usually many heterogeneous devices with different functionalities, capabilities, and multiple programming languages to access them. So an abstraction layer is necessary to abstract this heterogeneity in order to achieve a seamless integration with anything. Through a middleware, users and applications can access the data and devices from a set of interconnected things, hiding communication and low-level acquisition aspects [55]. We now summarise different IoT middleware in this context. In the comparison we have also added the web service protocols DPWS and CoAP, which although they are not middleware as such, they do incorporate several middleware features and are important protocols to promote the IoT towards the WoT.

GSN [167], [26] is a middleware that originated in 2005 as a platform for processing data streams generated by WSN which provides a platform for flexible integration and deployment of heterogeneous WSNs. The key concept in GSN is the virtual sensor, which describes sensors across XML files, is able to abstract from implementation details of access to sensor data and structures through different configurations of the data stream which the virtual sensor consumes and produces. GSN has a container-based architecture comprising several layers: the virtual sensor manager (VSM) which manages virtual sensors and underlying infrastructure; the query manager which parses and executes and planning SQL queries. It also has a configurable notification manager to configure alerts; and lastly, at the top, an interface layer for access through web services and via Web. VMS manages connection with devices through wrappers, currently there are wrappers for TinyOs platforms, several devices and generic wrappers. Moreover, different instances of GSN can establish communication through remote wrappers.

Although GSN handles the heterogeneity devices and acquisition level problems, there is actually a large heterogeneity data problem when data have to be interpreted and understood [168], [55]. An extension of GSN middleware, called XGSN, is presented in [55] to address this problem by facilitating the discovery and search tasks in an IoT environment. XGSN provides semantic annotation for GSN middleware, enriching virtual sensors with semantic data by means of an extension of the SSN ontology. The main semantic annotations of XGSN are concerned with sensors, sensing devices and their capabilities, and the measurements produced which have not been previously described in GSN. Moreover, in the same way as GSN, XGSN offers interfaces to query data and manage virtual sensors, and even integration with the Linked Sensor Middleware for storing and processing stream data.

DPWS (Device Profile for Web Services) [28], [169] is an OASIS web services specification for embedded devices and devices with few resources. DPWS is based on SOA (Service-Oriented Architecture), where each device is abstracted as a set of services. DPWS defines a protocol stack built over web services standards: SOAP (Simple Object Access Protocol) 1.2, WSDL (Web Services Description Language) 1.1, XML Schema and WS-Addressing; and also defines several protocols for discovery, security, messaging and eventing. The DPWS services can be specified using XML Schema, WSDL and WS-Policy like a web service, and are discovered by the WS-Discovery protocol in a plug-and-play mode. Services are discovered, specifying the type of the device, the scope in which devices reside or both. The WS-Discovery protocol uses multicast with SOAP over UDP in order to reduce network traffic overhead and during the discovery process

each device displays its metadata information such as its end-point reference or a set of messages that can be sent or received. Moreover, the WS-Eventing protocol defines a publish/subscribe protocol allowing devices to register to receive messages about other devices. Lastly, the WS-Security is responsible for providing different levels of security, such as authentication and confidentiality in the devices. Currently, there are several implementations of DPWS like the WS4D open-source implementation [170] in C/C++ and different implementations in Java, and the adoption of Microsoft in their OS from Windows Vista and Windows Server 2008 [171]. However, due to the DPWS protocol stack, the integration of DPWS in embedded devices may be too heavy, so a solution like the DPWS-compliant gateway proposed by Cubo et al. [172], could abstract the underlying IoT components while taking advantage of DPWS interoperability and semantics on embedded devices.

The OMG Data-Distribution Service (DDS) for real-time systems [173] is a data-centric publish/subscribe OMG (OMG: <http://www.omg.org>) specification for real-time and embedded systems. The specification defines both the communication semantics (behaviour and QoS) and the APIs for efficient delivery of information between producers and consumers. In the same way as Apache Kafka, data publishers in DDS publish typed data-flows over topics which consumers can subscribe to. However, DDS does not contain a server to connect publishers and producers, rather it provides dynamic and extensible applications by means of dynamic discovery of publishers, subscribers and data types. DDS follows the data-centric publish-subscribe model, through typed interfaces, providing the information needed to tell the middleware how to manipulate the data and also providing a level of safety type. A domain is the way to abstract different entities (publishers, consumers, data types) in the same group. At the beginning of each application, it is necessary to define the data types for the communication, although publishers and consumers have been designed to support multiple data types. Unlike other middleware, DDS defines QoS policies like data durability, latency maximum or data ownership. Currently, there are several official implementations of DDS which include multiple programming languages and SOs as well as different license types.

For communication between devices WSs (Web Service) are usually used. The WSs offer a way to communicate, share and access data information over the Internet. Nowadays, there are many WSs offering a variety of functionalities over the Internet, like current currency exchange, information on zip codes and cities, connecting with social networks, and so on. The main ones are RESTful and SOAP WSs. However, WSs normally operate over HTTP, or in the case of SOAP, they work with exchange XML, which is a major limitation for constrained devices. CoAP [174], [7] is a transfer protocol designed for constrained devices. As aforementioned, CoAP follows the same style of RESTful architecture, but it uses UDP thereby reducing TCP connection and transfer overheads. CoAP defines GET, POST, DELETE and PUT operations and responses like normal RESTful WSs, allowing a seamless integration with HTTP platforms. However, in HTTP the transactions are usually initiated by clients with pull mode, so this scenario is too expensive for devices with power limitations, battery and network. CoAP, in contrast to HTTP, uses an asynchronous mode to push information from servers to clients following the observer design pattern. Moreover, to allow interoperability between CoAP end points, CoAP includes a technique for advertising and discovering resource descriptions, and depending on the situation, CoAP allows several types of confirmation messages.

Nonetheless, HTTP is widespread across the Internet in many applications and systems, unlike CoAP. A proxy design to leverage the compatibility with HTTP and the seamless adaptation of CoAP on embedded devices is presented in [121]. The proxy shows that the WebSockets protocol is more suitable than HTTP for long-lived communications, whilst the final CoAP devices are not overloaded since they are only connected with the proxy following the observer pattern, handling the proxy all consumers' connection. The proxy also provides HTTP compatibility, and leveraging the observer pattern, uses a cache to reduce the number of requests to CoAP devices. On the other hand, Castro et al. [175] avoid using external intermediate application servers for interconnecting clients with end devices, and propose CoAP Embedded Java Library for interconnecting Web browsers directly with end devices. There are different implementations of CoAP both with private and public licenses, as in several programming languages [45]. CoAP can also considerably reduce power consumption with respect to HTTP in many situations as show in [30].

LinkSmart [176] emerged from a European research project to develop a middleware based on a service-oriented architecture for network embedded systems. LinkSmart has been designed, taking into account the common problem of compatibility between proprietary protocols and devices. The result of the project is a software gateway which acts as a bridge between the digital world and the underlying IoT devices. Loosely coupled OSGi based webs services are powered by LinkSmart in order to allow applications to control, observe and manage physical devices through the LinkSmart gateway. The use of OSGi provides a components system which allows the deployment and control of components without requiring a reboot. LinkSmart is so far the only one which allows dynamic deployment and control of components during execution, a key issue in critical devices which cannot be rebooted. By default, LinkSmart has components installed, like the Network Manager, which can be used to register and discover services in the LinkSmart Network and for direct communication between LinkSmart nodes. Additional, optional components provide a topic-based publish-subscribe service, device discovery and secure identity and encrypted communications. LinkSmart requires a JVM (Java Virtual Machine) and a minimum memory of 256MB so it is too high for embedded devices.

Table 5.4: IoT Middleware comparison

Middleware	Communi- cation model	Discovery	Connection with de- vices	Real-time	Devices require- ments	Virtual de- vices	QoS sup- ported	Security	License
DDS	Peer-to-peer	Yes	Protocol instantiation	***	High/ Medium/ Low	No	Yes	DDS-Security (Authen- tication, Privacy, Ac- cess Control, Logging and Data Tagging)	Different implementa- tions (enterprise and open-source)
GSN	Peer-to-peer	Yes	Wrappers for TinyOs platforms, HTTP generic, several de- vices, and generic UDP and serial	*	Medium/ Low	Yes	No	N/A	GNU General Public License Version 3 (or later)
DPWS	Client-server	Yes	Protocol instantiation	*	High	No	No	WS-Security (In- tegrity, Confidentiality, Authentication)	Different implementa- tions (enterprise and open-source)
CoAP	Client-server asyn- chronous	Yes	Protocol instantiation	***	Low	No	No	DTLS (Authentication, Privacy) Different implementations (enterprise and open- source)	Different implementa- tions (enterprise and open-source)
LinkSmart	Peer-to-peer	Yes	Protocol instantiation or connection through the proxy	*	High/Low	Yes	No	Crypto Manager (Au- thentication, Privacy)	GNU Affero GPL v3
LooCI	Peer-to-peer	Yes	Protocol instantiation or wrapper with devices	*	High/ Medium /Low	No	No	Specification on Se- cLooci	GNU General Public License version 3

For the latter, a proxy component has been incorporated to abstract different communication protocols and interfaces such as ZigBee, Bluetooth and USB. The publish/subscribe mechanism provided by the Event Manager component is another key component which exchanges XML messages between senders and receivers, and is suitable for asynchronous environments. Moreover, IOS, Android and Windows tools have been incorporated to control and manage the IoT resources deployed.

The component infrastructure middleware introduces a novel paradigm in IoT, where the underlying devices can be reconfigured, as well as offering common mechanisms for deploying and managing components in run-time. However, the requirements of JVM in OSGi make it unsuitable for constrained devices. LooCI [177] is a middleware for building component-based applications in WSN which takes into account this restriction. LooCI provides interoperability across various platforms: a Java micro edition for constrained devices known as Squawk, the open-source OS for IoT Contiki, and the OSGi. Like LinkSmart, LooCI has a distributed event bus to allow publish/subscribe in addition to direct bindings for communication between all the components deployed. All the communication is based on type events, so it leads to type-safe communications and services and binding discoveries. In contrast to LinkSmart, LooCI instead of having a proxy component to abstract different communication protocols or interfaces, leaves the implementation of the device communication in each component and standardises the networking services through a Network framework. Recently [178], a new middleware known as SecLooCI has been proposed to expand the LooCI middleware in order to enable a secure sharing of resource-constrained WSN devices. However, the SecLooCI middleware's implementation is not available for download as it has not yet been released.

Table 5.4 shows the results of the IoT Middleware comparison. All middleware surveyed have a mechanism to discover other systems and services. Most middleware contain a peer-to-peer model communication which uncouples the main point of failure in client-server models. DPWS has a client-server model and CoAP has a client-server with an asynchronous model since the information can be pushed from server to clients asynchronously. Several QoS mechanisms are only addressed by DDSs which, together with the real-time support, is maybe the main reason for the adoption of DDS in critical and governmental systems. The device virtualisation is addressed by GSN and LinkSmart through the proxy component. GSN, LinkSmart and LooCI are maybe the most versatile middleware, since GSN and LinkSmart contain support for several protocols and devices, and LooCI can implement any device communication through its corresponding interface. For the device requirements, CoAP is the most lightweight middleware, while the different instantiations of the different wrappers as the middleware instantiation itself mark the requirements in GSN, LinkSmart and LooCI. GSN is the weightiest middleware due to its protocol stack, and DDS does not define the communication stack so the weight of DDS may vary in each implementation.

DDS, DPWS and LinkSmart contain secure implementations, including security services like authentication and data confidentiality. On the other hand, CoAP can use DTLS for security, GSN does not mention anything about security and LooCI has released a new middleware for security but it is not yet available. DDS, LinkSmart and LooCI require that all data applications must be defined at the start, but at the same time this offers a safety type which is not present in the remaining middleware. Moreover, LinkSmart and LooCI are good choices when dynamic management and deployment of components are required and DDS

and CoAP are better for real-time support. Lastly, GSN, LinkSmart and LooCI have open-source licenses and the rest have both enterprises and open-source implementations.

5.3 The λ -CoAP Architecture: an IoT and Cloud Computing Integration

The λ -CoAP architecture provides an integration of cloud computing with the IoT, taking into account edge capabilities. On the one hand, the IoT heterogeneity inside the underlying devices involved is abstracted at the same time as the lightweight web services are enabled so as to access them through CoAP. On the other hand, the Lambda Architecture enables the real-time processing, actuation and analysis of the large amount of data generated by the IoT. An edge layer that resides between the IoT and the Lambda Architecture, aims to reduce the bandwidth and latency in IoT-cloud communications. The λ -CoAP architecture is complemented with a Smart Gateway to interconnect the Lambda Architecture, HTTP and CoAP; a CoM (CoAP Middleware) to install the CoAP in resource-embedded devices; and external components such as a Web UI and a component to manage and actuate over the underlying IoT, respectively.

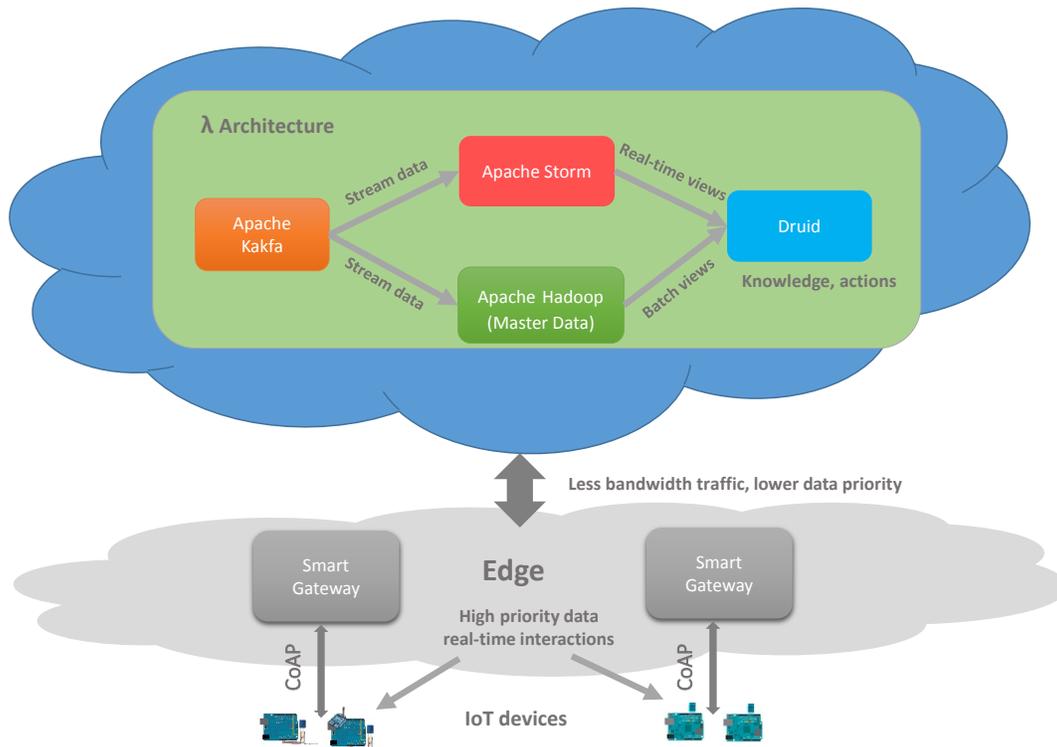


Figure 5.2: The λ -CoAP architecture: an overview

Figure 5.2 shows an overview of the λ -CoAP architecture. At the bottom, end devices incorporate a

middleware based on CoAP whose main function is to abstract the communication with the upper layers for querying data or actuate over them as well as discovering and low-level acquisition aspects. This middleware has also been used in Chapter 3. In the middle resides the Smart Gateways and the edge layer. Initially, the Smart Gateway acted as a bridge to interconnect the cloud platform, the underlying devices and the end users (Chapter 3). However, the Smart Gateway has been adapted to support edge computing architecture in addition to being an autonomous device with processing capabilities. With its computation power it can reduce the communication latency and lighten the networks. Therefore, the edge layer comprises the Smart Gateways deployed in the architecture. Lastly, at the top, the cloud is responsible for allocating and serving the components of the Lambda Architecture.

5.3.1 Lambda Architecture

The Lambda Architecture (LA) [179] is a paradigm composed by cloud platforms—a distributed queue, batch and stream processing and distributed data stores—designed to offer arbitrary queries over arbitrary real-time data. Immutable data—data that does not change as time passes, can only be queried or added—besides the precomputed views are key concepts in the LA. Due to the temporal characteristics of IoT data, and the need to extract knowledge and predictions of historical data, even in some situations with real-time, the LA is a suitable paradigm for the IoT, and will bring with it the following benefits: storage, real-time processing, scalability and machine learning. Moreover, in other scenarios, the true knowledge resides in the data, since it can be used to prevent certain situations from arising and to act in advance, and the LA stimulates these scenarios with immutable data and pre-computed views. The abilities of the LA with regard to large-scale smart environment management and big data storage/analytics are show in [180]. Apart from these benefits, LA is not centered on any particular technology, so each user can choose the necessary technology based on his/her needs.

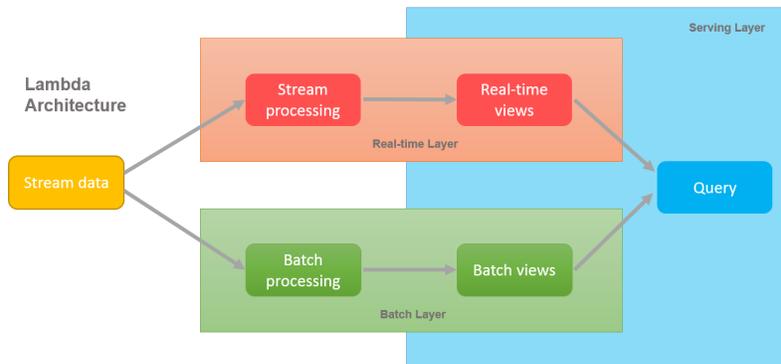


Figure 5.3: Lambda Architecture

All data received in the LA is stored in the master data and converted into another form during the processing, so the initial data received never change over time. The latter prevents human errors such as accidental deletions and updates. Additionally, cloud computing LA components replicate the data and offer

mechanisms for high availability and fault-tolerance, so the probability of data lost is small. Generating arbitrary functions and complex analysis over large amounts of data may require a considerable latency, an issue for real-time systems. Precomputed views have been designed to have available the complex analysis required besides extracting knowledge over the system data. The LA is decomposed into the following layers for these purposes:

- **Batch layer:** responsible for processing the historical data and generating the batch precomputed views through batch processing. The batch layer also stores all system data in an immutable form, known as master data, so it allows generating new precomputed views with new functions at any time.
- **Real-time layer:** makes up the high latency of the batch layer, processing the stream data and generating real-time precomputed views.
- **Serving layer:** responsible for abstracting the batch and real-time views offering a unique way of accessing them.

When stream data is received in the LA, a distributed queue distributes the stream data to the batch and real-time layers (see data flow in Figure 5.3). The batch layer stores the stream data received and periodically generates the batch views with the historical data. The batch layer carries out a complex analysis over the historical data and generates actions for the IoT based on this analysis. On the other hand, the real-time layer generates the real-time views and actions with the stream data received. Lastly, the serving layer merges the batch and real-time views for future queries.

5.3.2 Smart Gateway

The Smart Gateway is the component in the λ -CoAP architecture responsible for abstracting the IoT in addition to connecting it with the LA. The Smart Gateway is also responsible for incorporating a cross-layer proxy to interconnect the IoT with the Web. Each Smart Gateway comprises the following components:

- **LA Connector:** connects the Smart Gateway with the LA. When a stream provided by the Smart Gateway is received, the LA Connector sends it to the distributed queue in the cloud for further processing in the LA.
- **HTTP proxy:** is responsible for managing the HTTP requests from external sources such as an action provided by the LA and a query of a client. The HTTP proxy firstly checks the cache to see whether the data required is available. If the data required is not present, the HTTP proxy obtains the address of the device associated with the request, and uses it to make a CoAP request to obtain or act over the device that is sent to the CoAP Connector.
- **CoAP Connector:** receives requests from the Web Server to request data or actuate over a device. The CoAP Connector also connects the CoMs with the Smart Gateways to obtain the data or send the actions requested.

- **Resource Directory:** has two main functions, in fact, it is both a cache and a lightweight database. On the one hand, the Resource Directory stores information about end devices and their resources. On the other hand, a cache also stores data of devices in order to provide a cache and protects the end devices from an overload of requests.
- **Web UI:** provides an interface to manage and monitor the IoT devices for end users. Moreover, this interface enables the run-time management of sensors and actuators (presented in Chapter 3) and the management of containers.
- **Edge computing framework:** enables the development of the IoT application logic in the edge. This framework provides a visual interface, which makes the development of these applications intuitive.

5.3.2.1 Lightweight Virtualisation

Initially, Smart Gateways in the λ -CoAP architecture had a closed infrastructure comprising a set of components to interconnect the LA, the CoMs and end users. Upon a new development or a component modification, it is therefore necessary to reconfigure or install these components manually, without component isolation. Lightweight virtualisation technologies have garnered a lot of attention in the last few years due to their features: fast processes of building containers, high density of services per container and a high isolation between instances [181]. In contrast to traditional hypervisors, lightweight virtualisation technologies implement virtualisation of processes through containers in the operating system. This reduces the overhead of the hardware and virtual device virtualisation in traditional hypervisors, allowing the deployment of a high density of containers [182]. Therefore, the use of this virtualisation technology could help in the management of components in an IoT Gateway.

A lightweight virtualisation architecture has been applied to the Smart Gateway. The resulting Smart Gateway with its containers is shown in Figure 5.4. Two new components have been added to the Smart Gateway: a Web management UI for managing devices and containers and an edge computing framework to develop the edge logic. This architecture enables the management of processes at run-time and the reuse and backup of containers. Therefore, this virtualisation will facilitate the process update, backup and management in the Smart Gateways. In addition, each container is isolated from the rest.

5.3.2.2 IoT Devices and Virtualisation Management Web UI

A Web management UI has been incorporated inside the Smart Gateway to manage IoT devices and containers. This enables the management of Smart Gateways without the involvement of a cloud infrastructure. Sensors and actuators can be created and managed at run-time in the IoT devices deployed through the Web UI using the work of Chapter 3. This enables the management of the physical components in IoT devices without necessitating a reconfiguration and stopping their services. In this case, CoAP has been used, but the container-based architecture paves the way towards the integration of new IoT protocols.

Containers can also be managed in the Smart Gateway through a lightweight virtualisation integration. Containers represent the processes running in the Smart Gateways. The management allows a better control

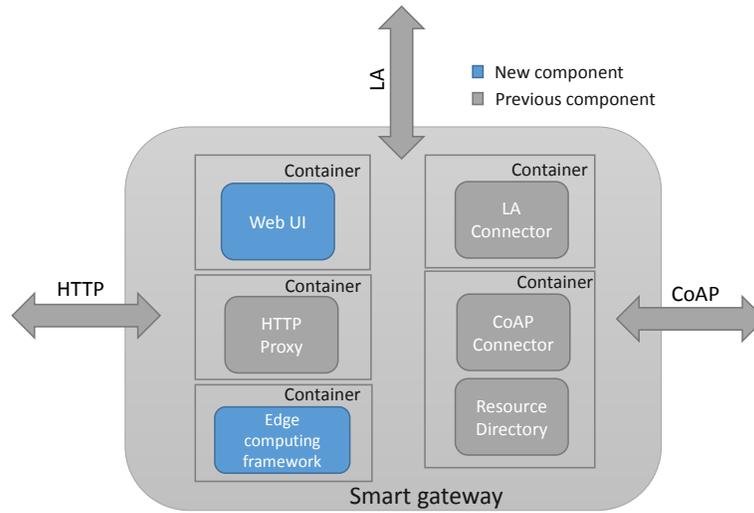


Figure 5.4: Smart gateway with lightweight virtualisation, evolution from Chapter 3 to the Λ -CoAP architecture

over their current components, facilitating their maintenance and enabling their reuse. As can be seen in Figure 5.4, the components are a part of containers and in turn can be managed by this component container. Apart from enabling the management of containers, the Web UI enables the submission of new container images that are automatically installed and available as new containers in the Smart Gateway.

5.3.2.3 Edge Computing Framework

Edge computing is enabled thanks to the inclusion of application logic at the edge of the network, which reduces the communication latency and alleviates the bandwidth with the cloud. This component enables a framework for the application development at the edge, in the Smart Gateway in this case. More specifically, the framework is based on a browser-based editor that allows visual data flow programming. A set of components have been created to filter and aggregate IoT data, so that application developers only need focus on the application logic. This component decides what data should be processed immediately or should be sent to the cloud. In the case of an IoT actuation, the communication latency will not depend on the communication with the cloud infrastructure, therefore it should be reduced. Discovery and IoT interactions are transparently done by the edge computing framework, thus application developers just need to focus on the application logic. Moreover, resulting flows can be easily portable between Smart Gateways.

The edge computing framework has a component-based architecture enabling the reuse of components in flows and allowing new components to be created for the application development. The development is done visually with the creation component flows. Visual programming can provide a better user experience, alongside a reduced perceived workload and a higher perceived success [43].

5.3.3 CoAP Middleware

The CoM (continuation from Chapter 3) is responsible for abstracting communication, low-level acquisition and discovering aspects in the IoT devices. The CoM provides an abstract layer for accessing low-level components like sensors as well as actuating over the actuators components in the IoT devices.

The CoM also manages CoAP resources which can be identified by a URI. CoAP resources are components responsible for the abstract operations in the IoT devices such as obtaining the value of a specific sensor and performing the action over an actuator component. Querying the underlying sensors installed in each device is performed through a resource with CoAP GET requests, whereas actuating over a component is performed through a resource with CoAP PUT requests. The CoAP requests are generated by the CoAP Connector of the Smart Gateways based on the HTTP requests. Therefore, through the Smart Gateways, users can query and actuate over the IoT.

The discovery process is initiated when a new CoAP middleware is deployed in the system. The discovery process has been designed to help the Smart Gateways to know how many CoM are deployed, which are alive and which are their resources. When a CoM is deployed in the system, it sends a broadcast/multicast message to discover a Resource Directory in the network. The Smart Gateway, upon the reception of the message, responds to inform that a Resource Directory is available. Finally, CoMs register a new end point along with their resources in the Resource Directory and keep an up-to-date representation of their registrations.

5.4 Implementation

5.4.1 Lambda Architecture

Based on the comparison done in Section 5.2.1, the Lambda architecture has been designed as the composition of the following cloud computing components: Apache Kafka as the distributed queue; Apache Storm for stream processing; Apache Hadoop for batch processing; and finally, Druid as the database for the serving layer. Therefore, Apache Kafka is responsible for receiving all IoT data from the Smart Gateways, which distributes the data stream to the real-time layer (Apache Storm) and the batch layer (Apache Hadoop). Real-time and batch views are generated by Apache Storm and Apache Hadoop, respectively. Apache Hadoop is also responsible for storing the master data. Finally, Druid unifies all pre-computed views with the last up-to-date representation.

Managing and deploying these cloud computing components individually can lead to huge efforts of maintenance and installation. To facilitate the management and deployment of these components, we have used Apache Ambari. Apache Ambari is intended to make Apache Hadoop clusters management simpler through an intuitive and easy-to-use management Web UI. Figure 5.5 shows a screenshot of the Apache Ambari management Web UI with the components of this Lambda architecture. Apache Ambari also monitors the status of each component and displays some metrics in the dashboard, which facilitates the monitoring of the components, otherwise it should be done individually. Note that Druid does not belong to the Hadoop cluster and therefore is not available in Apache Ambari. However, Druid has been integrated as a new service

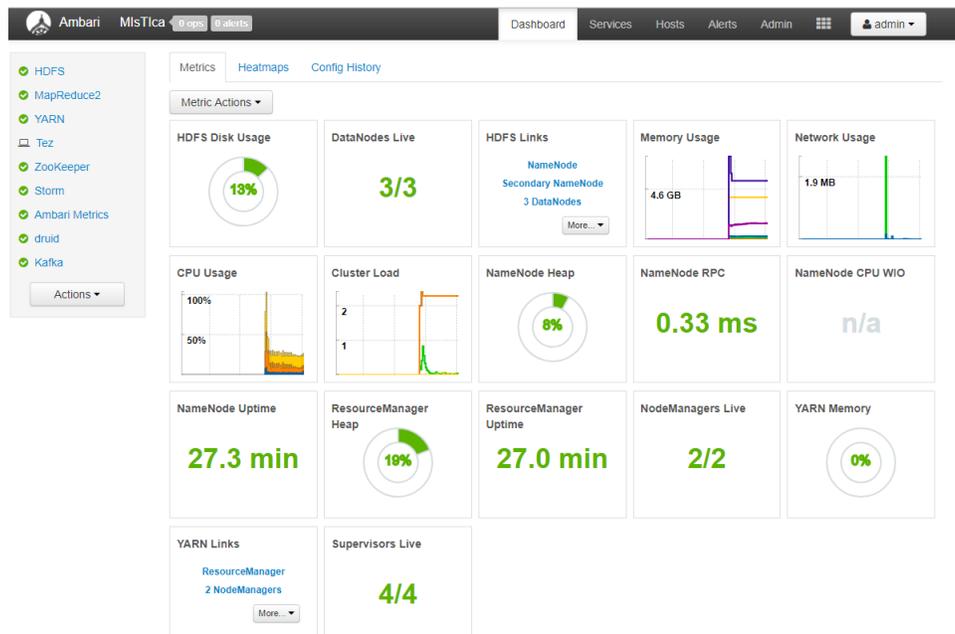


Figure 5.5: Lambda architecture management with Apache Ambari

in Apache Ambari to be managed and deployed like the rest of components of the Lambda architecture.

Two Apache Kafka consumers have been implemented to consume the Kafka log in Apache Hadoop and Storm. Whereas Apache Storm continually processes stream data and generates pre-computed views to the serving layer, Apache Hadoop generates pre-computes views periodically. The application logic should be the same in both the real-time and the batch layer.

5.4.2 Smart Gateway

As presented in Chapter 3, most of the Smart Gateway has been implemented in Python with the frameworks Django (Web UI) and Wheezy Web (HTTP proxy).

Although there are various lightweight virtualisation technologies, Docker¹ has been chosen for its compatibility with the target IoT Gateway e.g., Raspberry Pi² and the REST API support, which will be integrated into the Web UI. Containers will be fully managed with this API through an integration with the Web UI. Moreover, the API also paves the way to third party integration to manage the Smart Gateways.

Thanks to the experience gained and good results obtained in Appdaptivity (Chapter 4), Node-RED has been used for the edge logic development. A Websockets connection has been established between Node-RED and the components in the Smart Gateway, obtaining a full duplex channel to receive data and actuate over the IoT. As is true for Appdaptivity, application developers just need to focus on the application logic,

¹<https://www.docker.com/>

²<https://blog.hypriot.com/>

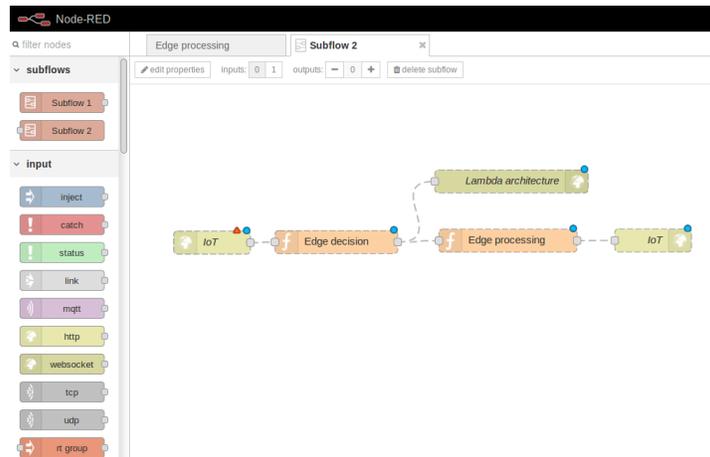


Figure 5.6: Edge computing framework web UI

while low-level aspects are managed by the framework. Figure 5.6 shows an overview of the framework for the edge logic development. From the left-hand side, application developers can select nodes from the palette to be part of the system, forming data flows as in Appdaptivity.

5.4.3 CoAP Middleware

The CoM has been extended from the solution in Chapter 3 to support new sensor libraries. Specifically, libraries for supporting the TSL2561 light sensor, the CCS811 air quality sensor, the DHT22 humidity and temperature sensor, the INA219 current sensor and a PIR sensor have been included. All of them have been integrated into a portable solution with ZigBee communication (Figure 5.7b). Figures 5.7a and 5.7b show the schematic and the integrated solution of the portable CoM. With the inclusion of the current sensor, the energy consumption of the CoM can be observed, enabling the evaluation of the impact of the different sensors deployed to the node in run-time.

5.5 Evaluation

In this section the λ -CoAP architecture is evaluated. The performance of the architecture has been evaluated using different test scenarios. First, IoT traffic was generated from a developed data source (not in real hardware) in order to generate multiple and high data rates without the need to use multiple hardware devices. This scenario enables the Lambda architecture to be evaluated in larger deployments than those that could be deployed in our research lab. Second, a real deployment with different IoT devices including the listed sensors and communication has been deployed to evaluate the CoM in IoT devices.

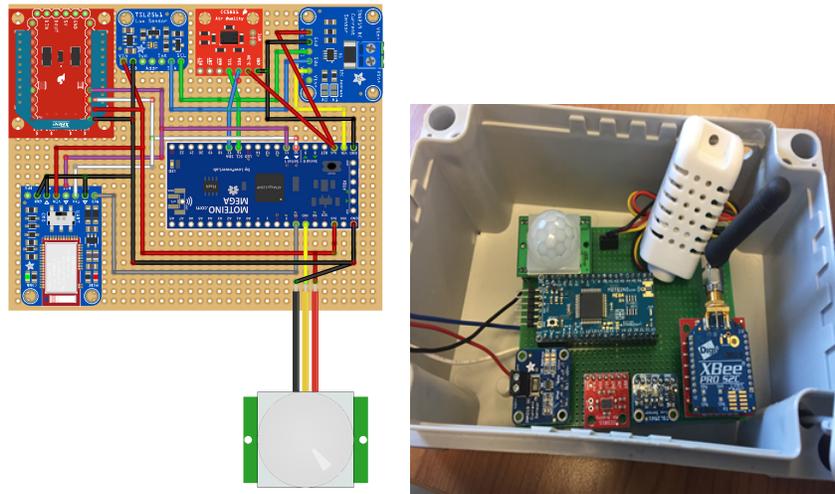


Figure 5.7: Portable solution with the listed sensors and communication. **(a)** Schematic of the portable solution; **(b)** Portable solution integrated.

5.5.1 Lambda Architecture

The Lambda architecture has been deployed in the virtualisation infrastructure of the Ada Byron Research Centre, at the University of Malaga. Specifically, 15 virtual machines have been deployed in VMware vCloud with the following configuration:

- 4x Centos6 (x86_64) 6 OS, 8GB RAM, 2x core, 100GB harddisk
- 12x Centos6 (x86_64) 6 OS, 4GB RAM, 2x core, 100GB harddisk

Table 5.6 shows the components that comprise the Lambda Architecture. These components are responsible for the batch and stream processing, in addition to consuming IoT data and storing the results. Note that some components such as HDFS, MapReduce2 and YARN comprise Apache Hadoop, whereas other components such as Apache ZooKeeper are required for the synchronisation and naming among other tasks for the rest of components. In any case, all components are monitored and managed through Apache Ambari, thereby they can be easily distributed or re-balanced when required.

The Lambda architecture has been evaluated through a developed client, which generated multiple data records (a total of 5.000.000 records in each experiment) with different message sizes. The client was executed in a Windows 7 PC with 8GB RAM, with an Intel core i7-4790 with 4 cores, and 1GB Ethernet, in the research building where the Lambda architecture is deployed. This experiment aims to evaluate the Lambda architecture with a high stress, and show the impact of the message size on its performance. Figures 5.8 and 5.9 show the throughput obtained in terms of bandwidth (MB/s) and records per second, respectively. Results demonstrate that the bandwidth increases when the message size is higher. This is due to the overhead of processing each message, which dominates the ingestion time in most messaging systems. On the other hand, the number of records processed per second decreases with respect to the message size. As can be

Table 5.6: Lambda Architecture configuration used in the Λ -CoAP architecture

Units	Component	Platform
1	NameNode	HDFS 2.7.1.2.3
	SNameNode	
3	DataNodes	
	NFSGateways	
	ZooKeeper Server	ZooKeeper 3.4.6.2.3
2	ZooKeeper Client	
1	History Server	MapReduce 2.7.1.2.3
	MapReduce2 Client	
1	App Timeline Server	YARN 2.7.1.2.3
	ResourceManager	
	YARN Client	
2	NodeManager	
1	DRPC Server	Storm 0.10.0
	Storm UI Server	
2	Nimbus	
4	Supervisors	
1	Metrics Collector	Ambari 2.1.2.1
	Ambari Server	
15	Metrics Monitors	
1	Broker Node	Druid 0.9.1
	Coordinator Node	
	Historical Node	
	Overlord Node	
	Tranquility Server	
2	Middleware Node	Kafka 0.8.2.2.3
	Kafka Broker	

seen, the message size creates a dualism between bandwidth and records per second. Taking into account that the average message size in the λ -CoAP architecture is 45 bytes, this configuration could manage 137725,87 records per second with a bandwidth of 5,91 MB/s. Therefore, the λ -CoAP architecture could be deployed in a global scenario with this configuration.

Finally, Figure 5.10 shows the average latency in the communications with the different message sizes. Likewise for the records per second, when the message size is higher, the latency has worse results. With the average message size in the λ -CoAP architecture, a total of 1804, 34ms is obtained, a considerable latency, especially for critical scenarios. In the edge layer we will see how this latency can be decreased.

Note that the total processing time of a final application has not been included, since it depends the particularity of the application and the complexity of its algorithms. As future work an evaluation of this architecture with multiple algorithms and IoT scenarios is planned.

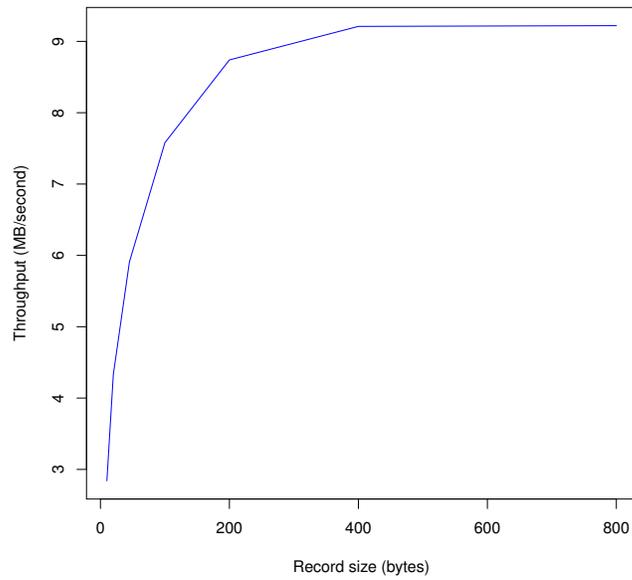


Figure 5.8: Ingestion throughput in the Lambda architecture: Record Size vs Throughput (MBs)

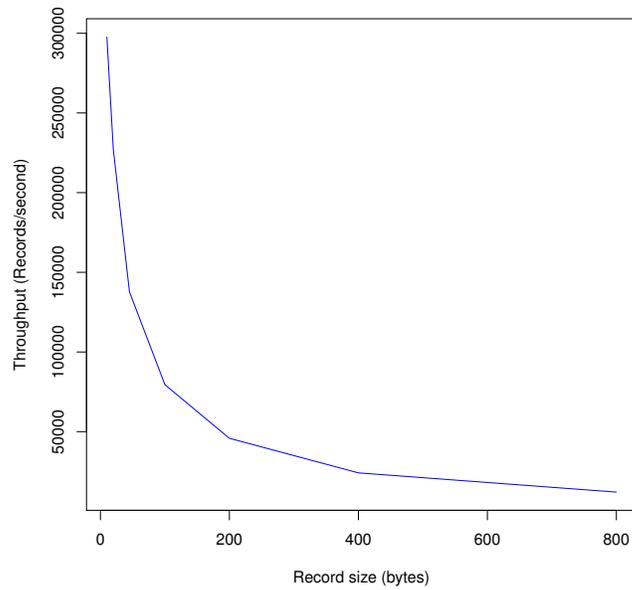


Figure 5.9: Ingestion throughput in the Lambda architecture: Record Size vs Throughput (records)

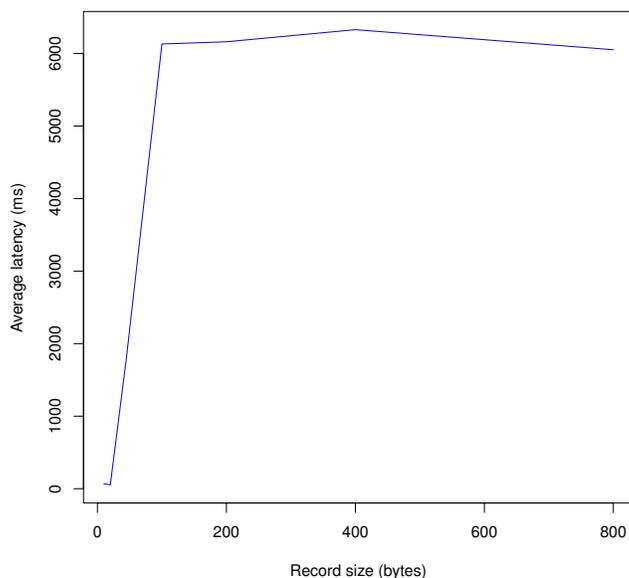


Figure 5.10: Average latency in the Lambda architecture with multiple message size

5.5.2 Smart Gateway

As mentioned, the Smart Gateways comprise the edge computing layer, enabling the edge logic development through visual data flow programming. This evaluation aims to demonstrate the benefits of using the edge architecture, and how the latency can be reduced in IoT-cloud communications. The evaluation of the Smart Gateway has been performed on a Raspberry Pi 3 model B with Raspbian Stretch OS and equipped with an SD card of 16GB. In a similar way as the evaluation of the Lambda Architecture, a client generated stream data to evaluate the performance. In this case, due to the lower capabilities of the IoT device, a total of 5000 records (more than intended to be generated by a Smart Gateway) were generated in each test.

Figure 5.11 shows the latency results obtained and its comparison with the Lambda Architecture. As can be seen, excluding the processing time, the ingestion time is mostly lower than for the Lambda Architecture (95, 4% lower in the average CoM packet). Therefore, this architecture reduces the latency when IoT data is generated and the bandwidth, a key challenge for the upcoming mobile networks. On the other hand, Figures 5.12 and 5.13 show the throughput results in terms of MB/s and Records/s respectively. In this case, the Lambda Architecture has better throughput in both cases. Although, this was expected due to the higher capabilities of the Lambda Architecture, this is not the goal of the edge layer, rather than reducing the latency and the bandwidth. In some situations, like processing the whole historical data, the edge layer may not be suitable, but critical response situations could benefit from its lower latency.

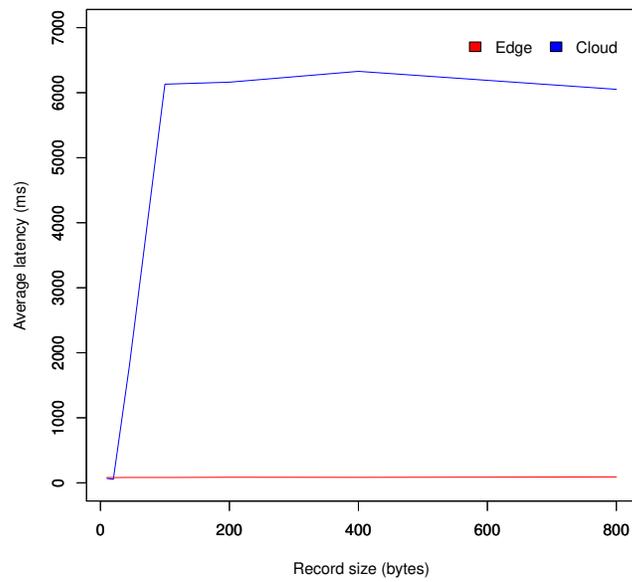


Figure 5.11: Average latency: cloud vs edge

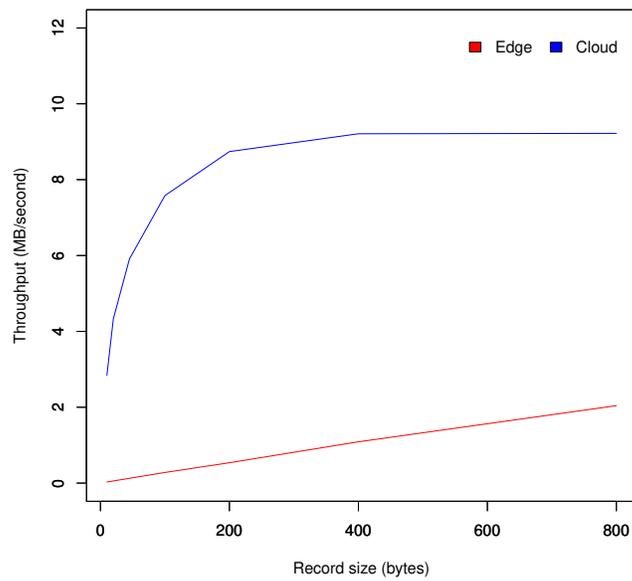


Figure 5.12: Ingestion throughput (MBs): cloud vs edge

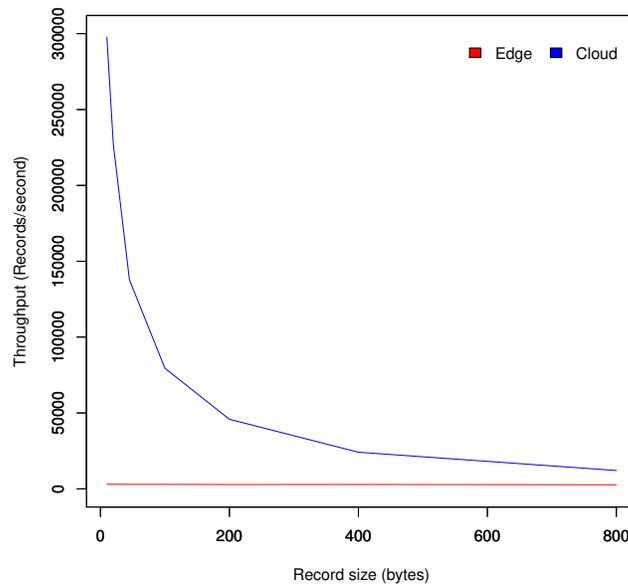


Figure 5.13: Ingestion throughput (records): cloud vs edge

5.5.3 CoAP Middleware

In this case, the evaluation was performed with the CoM installed in different platforms: Arduino Mega 2560 and Moteino Mega with the solution configuration shown in Figure 5.7. As was pointed out in Chapter 3, the power consumption in Arduino Mega 2560 was rather high, whereas Moteino Mega was chosen as an alternative to alleviate this. The results concur with this assumption. Figure 5.14 shows the power consumption of both platforms with the inclusion of the sensors listed in Section 5.4.3. As can be seen, the power consumption in Moteino Mega is at least 40% lower in all the cases.

Figures 5.15a and 5.15b show the Flash and SRAM of the CoM sketch in Arduino Mega and Moteino Mega respectively. As can be noted, there is a difference between the two platforms. Arduino Mega has double the size of flash capacity, whereas Moteino Mega has double the size of SRAM capacity. The results show that that the CoM can run comfortably in both platforms. The increase in consumption with respect to the results obtained in Chapter 3 is due to the inclusion of new sensors and libraries, which decrease the available memory. It should be pointed out that the SRAM is usually the most restricted part in the development, especially when debugging is included, so Moteino Mega has a good design, increasing its capacity. This along with its power consumption and its price (half that of Arduino Mega) establish Moteino Mega as a good candidate for the CoM.

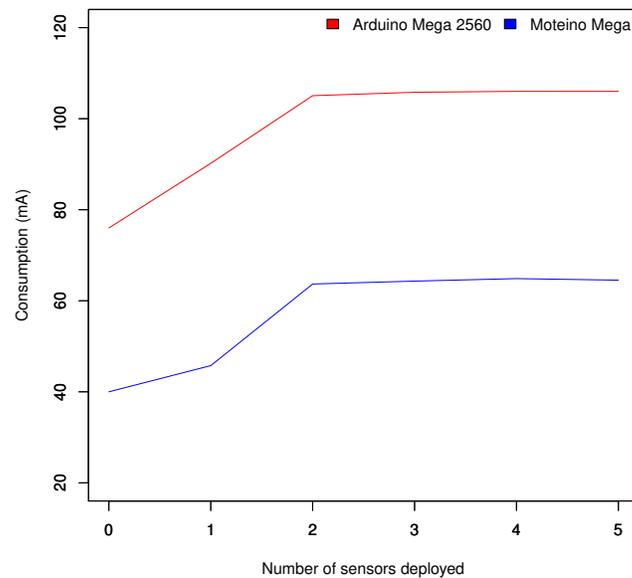


Figure 5.14: Power consumption of the CoM in Arduino Mega 2560 and Moteino Mega with different sensors

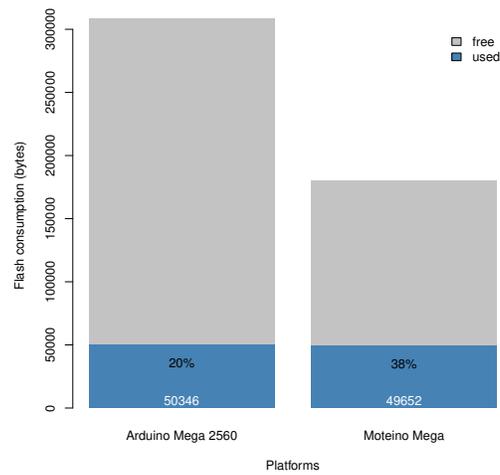
5.6 Challenges and Open Research Issues

Security and Privacy are key challenges in the deployment of IoT infrastructures. IoT devices are normally associated with constrained devices, so they are more vulnerable to attacks and threats. On the other hand, in many situations IoT systems use sensitive information like personal information or critical infrastructures, thus privacy with devices, cloud and network are key aspects. Roman et al. [183] mentioned the importance of security and privacy to push the IoT distributed approach into the real world. They enumerated the security mechanisms that can be integrated in the IoT: protocol and network security that offer end-to-end secure communication mechanisms; identity management to achieve authentication and authorisation to assure that the data is produced by a certain entity and to restrict the access control; privacy over generated data; trust between entities and users interaction and governance to support political decisions and stability; and fault tolerance to prevent and to detect attacks. Lastly, and the most difficult to avoid are attack models in the IoT such as denial of service, physical damage, eavesdropping, node capture to extract information and controlling entities. Moreover, security and privacy are one of the main concerns in cloud adoption.

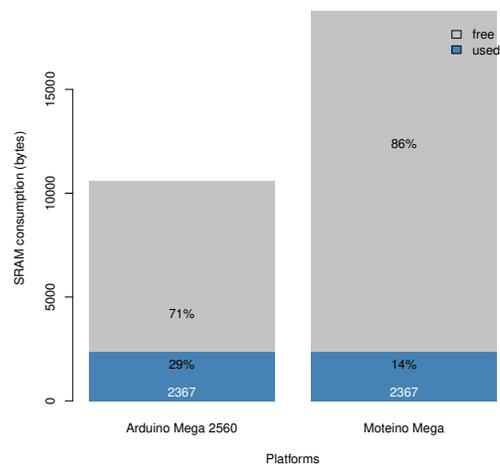
IPv6. The Internet is the main component of the Internet of Things, and everyone knows its addressable limitations in IPv4. Moreover, the adoption of technologies like CoAP, which allow interaction with embedded devices directly from the Internet, and the continued growth of the latter over the coming years, represent a greater effort to get rid of network address translation mechanisms and to address each thing or service in the world with a unique IP address. IPv6 has been designed to solve this problem through an IP

128 bit address, bringing with it several advantages such as a native integration with the Internet, end-to-end connectivity and a compliance with open REST interfaces [184]. In order to adopt the IPv6 in the embedded devices of the IoT, 6LoWPAN is a suitable specification. Nevertheless, at the present time there are not very many commercial platforms that have implanted these specifications. With the adoption of IoT, we are moving from networks with human initiated activities towards networks in which machine-to-human and machine-to-machine communications will be more numerous and IPv6 will pave the way. Moreover, current efforts to put IPv6 over Low Power Wide-Area Networks (LPWAN) will enable the use of 6LoWPAN in current disrupting network technologies like Lora and Sigfox, and with the LPWAN Static Context Header Compression (SCHC), the use of CoAP on them.

Context-Aware Computing. Context-Aware computing has gained in importance thanks to the growth of the IoT and its potential in it. Context-Aware computing allows us to store context information linked to sensor data allowing an easy and meaningful interpretation [185]. Moreover, the European Union identified context awareness as an important research area for Context-Aware IoT computing [186]. Context-Aware computing will furnish the IoT with new information that can be used for new applications and obtaining better-founded knowledge. Context-Aware computing also reopens the original use of ontologies as sources of knowledge, although its proper integration with cloud computing remains a pending issue.



a Flash consumption



b SRAM consumption

Figure 5.15: Memory footprint of the CoM in Arduino Mega 2560 and Moteino Mega

6

On Blockchain and its Integration with the Internet of Things

This chapter analyses the current challenges in Blockchain, and the challenges inherent in its integration with the IoT. Blockchain is a revolutionary technology that can be integrated with the IoT to keep its data reliable. The problem statement and research goals are defined in 6.1. Section 6.2 introduces the blockchain technology and analyses its main challenges. In Section 6.3 IoT and Blockchain integration is addressed, analysing the challenges that this integration involves. A state of the art in blockchain platforms for the IoT is presented in Section 6.4. Lastly, an evaluation and comparison of the performance of different blockchain components in an IoT device is performed in Section 6.5.

6.1 Problem Statement and Research Goals

In many areas where an exhaustive traceability of assets during their life cycle is required by regulations, data immutability becomes a key challenge. Concretely, European Union regulations require food producers to trace and identify all raw materials used in the elaboration of their food products in addition to the final destination of each of them. For example, in the case of a large food company with thousands of manufacturing suppliers and millions of clients, the information needs to be digitised and its processing automated to comply with regulation. One example of strong regulation in terms of traceability is the pork supply, regulated in many countries. In this scenario, in addition to tracing the raw material used in pig feed and treatments and the final destination of the pork, the transportation of the animals between factories must also be registered by law.

These scenarios involve many participants, some of them still relying on non-automated information handling methods. In the case of food contamination, which has been an important issue for the world population's health throughout history, information that is lost or difficult to find implies delays in the location of the problem's focus. This can also result in the public's mistrust of the contaminated products and a large decrease in their demand. According to the World Health Organisation, it is estimated that every year around 600 million people in the world suffer illness from eating contaminated food, of which 420,000 die from the same cause [187]. So, missing or inaccessible information can affect food security and customer health. In these kinds of scenarios the IoT has the potential to transform and revolutionise the industry and society, digitising the knowledge so that it can be queried and controlled in real time. This technology can be used to improve current processes in many areas such as cities, industry, health and transportation.

Apart of having the information available, it also has to be reliable. Keeping information reliable and immutable during its life cycle is critical in these environments. One way to provide trustworthiness in IoT data is through a distributed service trusted by all its participants that guarantees that the data remains immutable. If all participants have the data and they have the means to verify that the data have not been tampered with since the first definition, trustworthiness can be achieved. For instance, having a system that guarantees data reliability would allow governments to share and securely transfer information with citizens. This is exactly what Blockchain does. This new disruptive technology aims to revolutionise the way in which we share and access information. However, due to its novelty, the integration with the IoT is not yet well addressed. This contribution aims to analyse current challenges in the integration of the IoT and Blockchain, the potential advantages of their combined and the way they can be integrated.

6.2 Blockchain

The problem of trust in information systems is extremely complex when no verification nor audit mechanisms are provided, especially when they have to deal with sensitive information, such as economic transactions with virtual currencies. In this context, Satoshi Nakamoto, in 2008 [188] presented two radical concepts that have had a great repercussion. The first of these is Bitcoin, a virtual cryptocurrency that maintains its value without support from any centralised authority or financial entity. Rather, the coin is held collectively and

securely by a decentralised P2P network of actors that make up an auditable and verifiable network. The second of the concepts, whose popularity has gone even further than the cryptocurrency itself, is Blockchain.

Blockchain is the mechanism that allows transactions to be verified by a group of unreliable actors. It provides a distributed, immutable, transparent, secure and auditable ledger. The blockchain can be consulted openly and fully, allowing access to all transactions that have occurred since the first transaction of the system, and can be verified and collated by any entity at any time. The blockchain protocol structures information in a chain of blocks, where each block stores a set of Bitcoin transactions performed at a given time. Blocks are linked together by a reference to the previous block, forming a chain.

To support and operate with the blockchain, network peers have to provide, the following functionality: routing, storage, wallet services and mining [189]. According to the functions they provide, different types of nodes can be part of the network. Table 6.1 summarises the most common node types in the Bitcoin network.

Table 6.1: Bitcoin nodes and functionality

Wallet	Storage	Mining	Routing	
x	x	x	x	Bitcoin core
	x		x	Full node
	x	x	x	Solo miner
x			x	Light wallet

The routing function is necessary to participate in the P2P network, this includes transaction and block propagation. The storage function is responsible for keeping a copy of the chain in the node (the entire chain for full nodes, and only a part of it for light nodes). Wallet services provide security keys that allow users to order transactions, i.e., to operate with their Bitcoins. Finally the mining function is responsible for creating new blocks by solving the proof of work. The nodes that perform the proof of work (or mining) are known as miners, and they receive newly generated bitcoins, and fees, as a reward. The concept of proof of work is one of the keys to enable trustless consensus in blockchain network. The proof of work consists of a computationally intensive task that is necessary for the generation of blocks. This work must be complex to solve and at the same time easily verifiable once completed.

Once a miner completes the proof of work, it publishes the new block in the network and the rest of the network verifies its validity before adding it to the chain. Since the generation of blocks is carried out concurrently in the network, the block chain may temporarily fork in different branches (produced by different miners). This discrepancy is solved by considering that the longest branch of blocks is the one that will be considered as valid. This, together with the intensive nature of the block generation process provides a novel, distributed-trustless-consensus mechanism. It is very computationally expensive for a malicious attacker to modify a block and corrupt the block chain since the rest of the trusted miners would outrun the attacker in the block generation process and therefore the trusted branch of blocks will invalidate the one generated by the attacker. In technical terms, in order for a manipulated block to be successfully added to the chain, it would be necessary to solve the proof of work faster than the rest of the network, which is

computationally too expensive- it requires having control of at least 51% of the computing resources in the network. Due to the large computational capacity needed to modify the blockchain, the corruption of its blocks is practically impossible. This means that, even if the participants are not completely honest about the use of Bitcoin, a consensus is always reached in the network as long as most of the network is formed by honest participants. The solution proposed by Nakamoto was a great revolution in the reliability of unreliable actors in decentralised systems. More details about the blockchain architecture can be found in [188, 190].

Blockchain has also provided a technology where the concept of smart contract can be materialised. In general terms, a smart contract refers to the computer protocols or programs that allow a contract to be automatically executed/enforced taking into account a set of predefined conditions. For example, smart contracts define the application logic that will be executed whenever a transaction takes place in the exchange of cryptocurrency. In smart contracts, functions and conditions can be defined beyond the exchange of cryptocurrencies, such as the validation of assets in a certain range of transactions with non-monetary elements, which makes it a perfect component to expand blockchain technology to other areas. Ethereum [111] was one of the pioneer Blockchains to include smart contracts. Today smart contracts have been included in the majority of existing blockchain implementations, such as Hyperledger [191], a Blockchain designed for companies that allows components to be deployed according to the needs of users (smart contracts, services or consultations among others) with the support of large companies such as IBM, JP Morgan, Intel and BBVA.

All of this has contributed to the expansion of blockchain technology to a large number of areas where the features offered by this technology are needed: reliability, immutability and auditability. In fact, Blockchain is currently one of the top research topics of recent times, with more than 1.4 billion dollars invested by startups alone in the first 9 months of 2016 according to PwC [192].

6.2.1 Challenges

Although the key idea of Blockchain is simple, its implementation poses a great number of challenges. This Section introduces the main ones that its use brings about.

6.2.1.1 Storage Capacity and Scalability

Storage capacity and scalability have been deeply questioned in Blockchain. In this technology, the chain is always growing, at a rate of 1MB per block every 10 minutes in Bitcoin, and there are copies stored among nodes in the network. Although only full nodes (a node that can fully validate transactions and blocks) store the full chain, storage requirements are significant. As the size grows, nodes require more and more resources, thus reducing the system's capacity scale. In addition, an oversized chain has negative effects on performance, for instance, it increases synchronisation time for new users.

Transaction validation is a key component of the distributed consensus protocol as nodes in the blockchain network are expected to validate each transaction of each block. The number of transactions in a block and the time between blocks, modulate the computational power required and this has a direct effect on transaction confirmation times. Hence, the consensus protocol has a direct effect on the scalability of blockchain

networks.

Taking into account the trust model of Bitcoin and its scalability limitations, Bitcoin-NG [193] proposes a new Byzantine-fault-tolerant blockchain protocol which improves the consensus latency with respect to Bitcoin. Litecoin [67] is technically identical to Bitcoin, but features faster transaction confirmation times and improved storage efficiency thanks to the reduction of the block generation time and the proof of work, which is based on scrypt, a memory intensive password-based key derivation function. GHOST [194] is also intended to improve the scalability of Bitcoin by changing its chain selection rule. Off-chain solutions [195] are intended to perform transactions off the chain, increasing the bandwidth at the same time as it increases the probability of losing data. Another proposal suggests reducing the propagation delay in the Bitcoin protocol [196], however it can compromise the security of the network. Rather than increasing the scalability on blockchain, BigchainDB [197] adds blockchain characteristics to a big data distributed database. BigchainDB combines the high throughput and low latency characteristics of big data distributed databases with the immutability and decentralised system of Blockchain. Another important development is the Inter Planetary File System (IPFS) [198]. IPFS is a protocol designed to store decentralised and shared files enabling a P2P distributed file system to make the web safer, faster and more open. IPFS is intended to increase the efficiency of the web at the same time as it removes duplication and tracks version history for each file.

6.2.1.2 Security: Weaknesses and Threats

The Bitcoin protocol has been thoroughly analysed [199], and various vulnerabilities and security threats have been discovered. The most common attack is the 51% attack or majority attack [200]. This attack can occur if a blockchain participant is able to control more than 51% of the mining power. In this situation he/she can control the consensus in the network. The boom and fast evolution of mining pools (with GHash.io4 temporarily reaching 51% of the Bitcoin mining power in 2014), has increased the probability of this attack happening, which in turn could compromise the integrity of Bitcoin. In addition, the authors in [201] discuss the possibility of reaching a majority of mining power through bribery. The solo mining incentive or P2P mining would help to alleviate this problem. Many other consensus mechanisms proposed for blockchains are also susceptible to majority attacks, especially those that centralise the consensus among a limited number of users.

The double-spend attack consists in spending the same coin twice [202]. In Bitcoin a transaction should be considered confirmed only after the block where the transaction is stored has a certain depth in the blockchain, typically 5 or 6. This takes between 20 and 40 minutes on average [203]. There is a large variance in the confirmation time since it depends on many factors. In fast payment scenarios the trader cannot afford this wait. Therefore, in these scenarios, double-spend attacks are still possible.

Similarly, race attacks can work in these scenarios. To carry out this attack the user sends a transaction directly to the merchant, who accepts the transaction too quickly. Then the user sends multiple conflicting transactions to the network transferring the coins of the payment to himself. The second transaction is more likely to be confirmed, and the merchant is cheated. Similarly, the Finney [204] attack is a more sophisticated

double spend, since it requires the participation of a miner.

The well-known attacks Denial of Service (DoS), man in the middle or Sybil can also obstruct the network operation. Most P2P protocols and IoT infrastructures are vulnerable to these kinds of attacks, since they strongly rely on communications. In the eclipse attack [205], attackers can monopolise a node's connections, isolating it from the rest of the network and altering the view of the network for this node.

Code updates and optimisation in blockchain networks are usually supported by part of the cryptocurrency community and are intended to improve their underlying protocols. These improvements are known as soft and hard forks in blockchain terminology. On the one hand, soft forks provide an update of the software protocol that recognises backward-compatibility with the previous blocks. This requires an upgrading of the majority of the miners to the new software. However, upgraded functionality can also be rejected by the majority of the nodes keeping to the old rules. On the other hand, hard forks bring a radical change to the protocol, with no compatibility with previous blocks and transactions. Consequently, all the nodes have to upgrade to the latest update, and nodes with older versions will no longer be accepted. The community can be divided when a hard fork happens, resulting in two different forks of the network. Hard forks can also be cancelled if they have not built sufficient consensus like SegWit2x [206]. Noted examples of that division are Ethereum and Ethereum Classic; and Bitcoin, Bitcoin Cash and Bitcoin Gold. The aforementioned hard forks have progressed at the same time as the original networks and nowadays they are competing with each other. Nodes and users have to decide on a version, and the fork continuity will depend on these decisions. Hence forks, especially hard ones, can divide the community into two completely different blockchains and this can represent a risk to the blockchain users.

A common problem of virtual currencies, beyond the controversy surrounding their real value, is the problem of coin loss. If the wallet key is forgotten, there is no mechanism to operate with these coins. It has been estimated that 30% of bitcoins are lost.

Finally, quantum computing could be seen as a threat to Bitcoin, since the computing power of these computers could break the security of digital signatures. Furthermore, technology progresses over time and every day new bugs and security breaches are discovered. These improvements and bugs can compromise public blockchains with encrypted data since blockchain data is immutable.

6.2.1.3 Anonymity and Data Privacy

Privacy is not enforced in the Bitcoin protocol by design. A key feature of Bitcoin is its transparency. In Blockchain each transaction can be checked, audited and traced from the system's very first transaction. This is indeed an unheard of new level of transparency that doubtlessly helps to build trust. However this transparency has a knock-on effect on privacy, even though there is no direct relationship between wallets and individuals, user anonymity seems to be compromised despite the mechanisms that Bitcoin provides, such as pseudonymous and the use of multiple wallets. In this sense, some effort has been made to provide stronger anonymity features in Bitcoin. On the other hand not just open virtual currencies, but many applications based on public blockchain technology require a higher level of privacy in the chain, specifically those that deal with sensitive data.

Popular attempts to tackle the anonymity problem in Bitcoin are Zerocash [207] and Zerocoin [208] which propose that Bitcoin extensions have completely anonymous transactions, hiding the sender, the receiver and the information itself. Monero [74] uses a ring of signatures to make transactions untraceable, so that they cannot be easily traced back to any given person or computer.

Similarly, transaction mixing services or tumblers, provided by Bitcoin Fog [209] and Bit Laundry can increase the anonymity. These services break up transactions into smaller payments and schedule them to obfuscate transactions for a fee. However, these kinds of services are said to be prone to theft. Likewise, the coin-mixing approach, originally proposed in CoinJoin [210] helps to anonymise Bitcoin. The idea is that users agree on joint payments, so that it can no longer be assumed that transaction inputs are from the same wallet. However the previous negotiation required between users, typically performed by mixing servers, could lack the required anonymity depending on the implementation. Later, this approach inspired Dark Wallet [211], a browser plugin that allows completely private anonymous Bitcoin transactions; Dash [73], known as the first cryptocurrency focused on anonymity and privacy; MixCoin [212], that adds cryptographic accountability mechanisms and randomised mixing fees to increase security; CoinShuffle [213] that proposes a modification of CoinJoin in order to increase security; CoinSwap [214] that proposes a four-transactions mechanism based on the inclusion of intermediaries that receive coins and make the payment with unconnected coins; and Blindcoin [215] that increases the anonymity of the mixing server. In general, these attempts to increase anonymity in Bitcoin typically embrace the idea of maintaining a deregulated Bitcoin, and are therefore usually accused of encouraging illicit activities, such as the acquisition of illegal products on the Darknet or money laundering.

In order to increase privacy, data in the blockchain can be encrypted. Hawk [216] stores encrypted transactions. The Hawk compiler is responsible for translating the generic code written by programmers into cryptographic primitives that enable information anonymity in transactions. The Enigma project [217], in addition to encryption, splits data into unrecognisable chunks and distributes them through the network, in a way that no node ever has access to the data. It uses a decentralised off-chain distributed hash-table (DHT) accessible through the blockchain to store data references.

The problem of privacy in private blockchains can be tackled differently, since by definition they must provide authentication and authorisation mechanisms. However, even inside a private blockchain, participants want to preserve the privacy of their data. Quorum [218] for instance, is a private permissioned blockchain based on Ethereum that uses cryptography to limit the visibility of sensitive data and segmentation to increase privacy of data. Multichain [219] integrates user permissions to limit visibility and to introduce controls over which transactions are allowed and which users can mine. Rockchain [220] is also based on Ethereum and follows a data-centric approach, where public calculations can be performed on private data and accumulative results can be obtained preserving data privacy. This approach offers a distributed file system that allows users to manage data privacy through smart contracts in Ethereum. Hyperledger Fabric [191] provides a distributed and scalable ledger focused on enterprise environments. To provide blockchain networks with privacy control, Hyperledger Fabric provides an identity control service and access control lists through private channels where users can control and restrict the access to their shared information in the network.

Thanks to this mechanism, members of the network know each other through their public identities, but they do not have to know the information that it is shared in the network.

Another approach to tackle data privacy is to store sensitive data outside the chain, commonly referred to as the off-chain solution [221]. This kind of solution favours systems that manage large amounts of data, since it would be impractical to store them inside the blockchain. In addition, they are particularly suitable for systems that deal with highly sensitive data that should have a tighter access control, such as health care applications. In this way the public blockchain can be used to store anchor data, so that proof to verify the integrity and time stamps of data is available. Users can verify data without relying on authorities, just by checking the blockchain, and data are safely stored outside. Obviously, these off-chain sources must be fault tolerant and should not introduce bottlenecks or single points of failure. In [222] the authors propose using a Kademlia, a well-known DHT to store key-value pairs (user identities and permissions) to access, control and storage data. In [223] a pointer and a hash to validate the data obtained are stored in the chain. In this way, the data in the chain are links to the private data, and the hash is the mechanism that verifies that the information obtained has not been altered. Access control mechanisms for off-chain sources are provided to ensure that only authorised parties can access the information. The information can therefore be obtained from external sources in a secure and verified way with blockchain.

6.2.1.4 Smart Contracts

In 1993, Nick Szabo defined the smart contract as “a computerised transaction protocol that executes the terms of a contract”. One of the key features of a smart contract is that it has a way to enforce or self-execute contractual clauses. Until the emergence of blockchain technology, this was technologically unviable. Blockchain has turned out to be the ideal technology to support smart contracts. In addition, smart contracts have contributed significantly to the momentum of Blockchain, moreover this coupling has led to a second generation of blockchains, commonly known as Blockchain 2.0. The combination of automatically executed contracts in a trusted environment without centralised control promises to change the way current business is done.

Basically, the smart contract code is stored on the blockchain, and each contract is identified by a unique address, and for users to operate with it, they just send a transaction to this address. The correct execution of the contract is enforced by the blockchain consensus protocol. Smart contracts introduce a set of advantages such as cost reduction, speed, precision, efficiency, and transparency that have fostered the appearance of many new applications in a wide variety of areas. Although Bitcoin offers a basic scripting language, it has turned out to be insufficient, which has led to the emergence of new blockchain platforms with integrated smart contract functionality.

The most prominent smart contract blockchain platform is Ethereum [111]. Ethereum is a blockchain with a built-in Turing-complete programming language, that allows the definition of smart contracts and decentralised applications. The code in Ethereum’s contracts is written in “Ethereum virtual machine code”, a low-level, stack-based bytecode language.

Frequently, financial smart contracts require access to data about real-world states and events. This data

is provided by the so-called oracles. These entities are crucial for the successful integration of smart contracts within the real world, but they also create more complexity, since authentication, security and trust in oracles have to be provided [224].

The advantages of smart contracts do not come without cost, as they are vulnerable to a series of attacks [225, 226, 227] that bring new exciting challenges. Delegating contract execution to computers brings with it some problems, since it makes them vulnerable to technical issues such as hacking, bugs, viruses or communication failures. Bugs in contract coding are especially critical because of the irreversibly and immutable nature of the system. Mechanisms to verify and guarantee the correct operation of smart contracts are necessary for them to be widely and safely adopted by clients and providers. The formal validation of the contract logic, and its correctness are research areas where contributions are expected to be made in the years to come [228].

In addition, real-life contracts usually have clauses or conditions that are not quantifiable. In this sense, there is still a lot of work to be done in order to model the conditions of the contracts in smart contracts, so that they are representable and quantifiable for a machine to execute them. Additionally, efforts to provide tools for users to be able to specify and understand smart contracts are needed [229].

6.2.1.5 Legal issues

The absence of a central authority, the non-existent minting entity, and therefore the total dearth of censorship in Bitcoin is an attractive and at the same time dangerous peculiarity. Bitcoin users are commonly accused of using the network for fraudulent purposes, and thus the technology is suspected of promoting or facilitating illegal conduct. Bitcoin, as the first decentralised cryptocurrency has generated a lot of dispute [230]. On the one hand, with regard to its value, some experts claim it is a fraud [231] and that it will totally collapse [232], while at the same time others estimate that its value will reach 100.000 dollars in 10 years [233]. The European Central Bank, has warned of its volatility risk but have also admitted its potential as a financial innovation [234]. However, with regard to the lack of governance, many countries are developing new laws in an attempt to regulate the use of virtual currencies. A map of Bitcoin regulation status is available at [235]. This situation creates a lot of uncertainty, and seems to be the reason behind its recent fall [236].

Banks and governments will have their say as to whether or not the currency becomes legal tender. Legal implications in the context of currencies is an important concern, as they can directly and negatively affect blockchain applications based on that currency.

Many private and permissioned blockchain applications have recently emerged. These are blockchains that grant write permissions to a predefined peer or set of peers. This can bring some benefits to authorisation and authentication mechanisms, for instance key recovery or transaction redemption, and can also contribute to simplifying the problem of privacy and to reducing transaction latency. In fact, interesting market opportunities have arisen in insurance coverage for Bitcoins that would no longer be necessary if the authorities handled this responsibility. The threat of mining pools controlling the network, together with other vulnerabilities, favours the development of such blockchains, and governments are obviously interested in a regulated and controlled use of this technology in many applications. However, this means that the trustless network

will regress to a third-party trust network, losing part of its essence. In addition, this can potentially create bottlenecks if solutions include centralised entities. The features of these blockchains are closer to the features of distributed databases.

On the other hand, the key to increasing confidence in this technology could be the involvement of governments and/or large consortium of companies in their development. Ongoing initiatives in this direction will be helpful [237, 238]. Currently, personal information is distributed among different entities: government, universities, companies and so on. The information is spread across many entities and accessing it is time consuming, even when said entities answer to the same authority, e.g., the government. This leads to obstacles in accessing information in addition to a lack of a trustworthy service that guarantees the information. A trustworthy and global identity service with the information of each person would be disruptive at the present time. Nevertheless, each country has its own laws and regulations.

Initiatives such as Alastria [239] aim to involve multiple entities in the development of a national regulated blockchain, from public notaries to universities and private companies. They plan to enable a public and legal wallet for each person. Companies can also be part of the network. Therefore, each personal wallet could be a digital proof of possessions, companies where he/she has worked, college degrees and so on. This information could be used in a legal and trustworthy way for many services. For instance, in the case of a job interview, candidates could share their college degrees and work experience information with the interviewers. This information is reliable and verifiable by definition. These initiatives are the key to expanding blockchain inside government institutions, and the first step in creating a common and regulatory framework for blockchain systems. Moreover, this could also facilitate the administrative transactions of the population to obtain their information, the data transfer between countries, the reduction of corrupt information and a seamless integration between population, companies, government and universities. However, this also brings about an easy way to obtain highly private information, so the privacy and security considered in the rest of the work should necessarily go hand in hand with these initiatives.

6.2.1.6 Consensus

Consensus mechanisms [240, 241, 242] are responsible for the integrity of the information contained in blockchain, while defending against double-spend attacks, and therefore are an essential part of blockchain technology. The final goal is to achieve consensus in a distributed network without central authorities and with participants who do not necessarily trust each other.

The consensus based on the proof of work (PoW), which has worked so successfully in Bitcoin, forces miners to solve a computationally-intensive easily-verifiable task in order to create a new block. Once solved, the solution is published and the new block is added to the chain. The new block is spread across the network and the rest of the participants verify it and append it to its local blockchain copy. This process can simultaneously occur in different parts of the network. This is why the chain is in fact a tree. There are several, valid branches existing simultaneously in the blockchain network. When peers append a new block, they also have to check that the branch is the one with the most accumulated work (difficulty), that is, the longest chain which is assumed to be the valid one. This allows consensus to be achieved quickly. A key

drawback is that PoW makes Bitcoin dependent on energy consumption. The aforementioned 51% attack is a potential attack on the Bitcoin protocol. Additionally, the incentives in PoW are unexpectedly promoting centralisation as the proliferation of mining pools confirm. This together with the mint reduction, reward diminution and fee increase could compromise the system [243] in the future. Certainly, PoW has certain drawbacks such as high latency, low transaction rates and high energy expenditure that makes it unsuitable for many applications. As stated, the latency, or block frequency of 10 minutes may also be impractical in many scenarios. Despite this, some platforms do use or have adapted PoW, such as NameCoin, Litecoin, Ethereum, Dogecoin and Monero. Primecoin, for instance, mitigates the energy loss by proposing useful computationally-intensive tasks, such as the prime numbers search that can have applications alongside.

Not surprisingly many attempts to change PoW have recently been proposed, probably underestimating the complexity that this change implies, nevertheless it is not clear if they expose the security properties in the same way as PoW. The most popular alternative approach to consensus in Blockchain is the Proof of Stake (PoS). It is based on the fact that those users who own more coins, are more interested in the survival and the correct functioning of the system, and therefore are the most suitable to carry the responsibility of protecting the system. Basically, the idea behind using PoS is to move the opportunity costs from outside the system to inside the system. The algorithm randomly determines the user responsible for the creation of each block, based on the number of coins he/she owns. A common critique is that this approach does not provide incentives for nodes to vote on the correct block (known as the nothing-at-stake problem). Additionally, it is negative in the sense that it promotes enrichment of the rich. PoS was originally used by Peercoin and later in Nextcoin, Nxt [68], Crave and Ethereum. A variant of PoS is the Delegate PoS (DPoS) of BitShares, Monax, Lisk or Tendermint. In BitShares [70], a number of selected witnesses validate signatures and time stamps of transactions by including them in blocks. The election is performed by voting, and each time a witness successfully produces a block it is rewarded. This approach allows delegates to set the block latency, block size and confirm transactions in just a second.

The Leased Proof of Stake allows users to lease funds to other nodes, so that they are more likely to be selected for block creation, increasing the number of electable participants, and therefore reducing the probability of the network being controlled by a single group of nodes. Rewards are proportionally shared.

The Proof of Burn (PoB) [244] proposes burning coins, that is, sending them to a verifiable unspendable address, in order to publish a new block. Like PoW, PoB, is hard to do and easy to verify, but in contrast requires no energy consumption. In addition, PoB has some economic implications that contribute to a more stable ecosystem.

Nem's [245] approach to consensus, called Proof of Importance, associates an importance value with each account, in this way building a reputation system in the network. The chance of being chosen to create a block depends on this value, the computation of it also takes into account the number of coins and the number of transactions performed. In other words, productive network activity is also rewarded, not just the amount, promoting the useful behaviour of the users.

Other extended variants are the activity test (PoA), a hybrid approach that combines both PoW and PoS, and the Elapsed Time Test developed by IBM that uses a random choice of manager to mine each block based

on runtimes within reliable execution environments. The Proof of Capacity [246], also known as proof of storage or space, uses available hard drive space instead of computing resources. This approach is used in Permacoin, SpaceMint and Burstcoin.

Private blockchains have specific features, since the number of participants is usually lower than public blockchains and are semi-reliable. They are usually registered in the system with a predefined set of permissions. These systems, therefore, require specific consensus mechanisms that fit these characteristics.

Some of these alternative mechanisms are Paxos [247], developed by Lamport and Microsoft based on state machine replication; Chubby [248], based on the former and developed by Google, which is defined as a distributed blocking service. These approaches have the advantage that they are adaptations of formal algorithms, and therefore their features have been formally proved. RAFT [249] which separates key elements of consensus such as choice of leaders, record replication and security, and forces a greater degree of consistency to reduce the number of states that need to be considered. The Practical Byzantine Fault Tolerance (PBFT) algorithm, which is based on state machine replication and replicate voting for consensus on state change, is used in Hyperledger and Multichain. SIEVE [250], treats the blockchain as a black box, executing operations and comparing the output of each replica. If there are divergences between the replicas, the operation is not validated. Another variant of the PBFT is the Byzantine agreement Federated Byzantine Agreement (FBA). In FBA each participant maintains a list of trusted participants and waits for these participants to agree on a transaction before being considered liquidated. It is used in Ripple [66]. Stellar [251] is another variant that employs the quorum and partial quorum concept. The quorum is a set of nodes, enough to reach an agreement, the partial quorum is a subset of a quorum with the ability to convince another given node about the agreement. HDAC [252] is a system, currently being implemented, which proposes an IoT Contract & M2M Transaction Platform based on Multichain. HDAC is specially tailored to IoT environments. It uses the ePow consensus algorithm whose main goals are to motivate the participation of multiple mining nodes and to prevent excessive energy waste.

Finally, the HC Consensus, proposed in Hydrachain, is based on a list of validators of which no more than one-third are unreliable.

To sum up, consensus mechanisms in public blockchains have been widely proposed but poorly, formally proved. It is mandatory to understand the guarantees they offer and their vulnerabilities prior to using them [253]. In this sense, the research community together with the industry have to work towards the validation of these mechanisms in order to demonstrate their legitimacy. To the contrary, private blockchains have adopted formal well-known solutions, but the limited list of participants in these blockchains also limit the diversity and potential of applications.

6.3 IoT and Blockchain Integration

The IoT is transforming and optimising manual processes to make them part of the digital era, obtaining volumes of data that provides knowledge at unheard of levels. This knowledge is facilitating the development of smart applications such as the improvement of the management and the quality of life of citizens through the

digitisation of services in the cities. Over the last few years, cloud computing technologies have contributed to providing the IoT with the necessary functionality to analyse and process information and turn it into real-time actions and knowledge [2]. This unprecedented growth in the IoT has opened up new community opportunities such as mechanisms to access and share information. The open data paradigm is the flagship in these initiatives. However, one of the most important vulnerabilities of these initiatives, as has occurred in many scenarios, is the lack of confidence. Centralised architectures like the one used in cloud computing have significantly contributed to the development of IoT. However, regarding data transparency they act as black boxes and network participants do not have a clear vision of where and how the information they provide is going to be used.

The integration of promising technologies like IoT and cloud computing has proven to be invaluable (Chapter 5). Likewise, we acknowledge the huge potential of Blockchain in revolutionising the IoT. Blockchain can enrich the IoT by providing a trusted sharing service, where information is reliable and can be traceable. Data sources can be identified at any time and data remains immutable over time, increasing its security. In the cases where the IoT information should be securely shared between many participants this integration would represent a key revolution. For instance, an exhaustive traceability in multiple food products is a key aspect to ensure food safety. Food traceability could require the involvement of many participants: manufacturing, feeding, treatment, distribution, and so on. A data leak in any part of the chain could lead to fraud and slow down the processes of the search for infection which can seriously affect citizen's lives and incur huge economic costs to companies, sectors and countries in the case of a foodborne outbreak [254]. A better control in these areas would increase food safety, improving the data sharing between participants and reducing the search time in the case of a foodborne outbreak, which can save human lives. Moreover, in other areas such as smart cities and smart cars, sharing reliable data could favour the inclusion of new participants in the ecosystems and contribute to improve their services and their adoption. Therefore, the use of Blockchain can complement the IoT with reliable and secure information. This has started to be recognised as mentioned in [255], where blockchain technology is identified as the key to solve scalability, privacy, and reliability problems related to the IoT paradigm.

From our point of view the IoT can greatly benefit from the functionality provided by Blockchain and will help to further develop current IoT technologies. It is worth noting that there are still a great number of research challenges and open issues that have to be studied in order to seamlessly use these two technologies together and this research topic is still in a preliminary stage.

More specifically, improvements that this integration can bring include (but are not limited to):

- **decentralisation and scalability:** the shift from a centralised architecture to a P2P distributed one will remove central points of failures and bottlenecks [112]. It will also help prevent scenarios where a few powerful companies control the processing and storage of the information of a huge number of people. Other benefits that come with the decentralisation of the architecture are an improvement of the fault tolerance and system scalability. It would reduce the IoT silos, and additionally contribute to improving the IoT scalability.
- **identity:** using a common blockchain system participants are able to identify every single device. Data

provided and fed into the system is immutable and uniquely identifies actual data that was provided by a device. Additionally, Blockchain can provide trusted distributed authentication and authorisation of devices for IoT applications [256]. This would represent an improvement in the IoT field and its participants.

- **autonomy:** blockchain technology empowers next-gen application features, making possible the development of smart autonomous assets and hardware as a service [118, 257]. With Blockchain, devices are capable of interacting with each other without the involvement of any servers. IoT applications could benefit from this functionality to provide device-agnostic and decoupled-applications.
- **reliability:** IoT information can remain immutable and distributed over time in Blockchain [116]. Participants of the system are capable of verifying the authenticity of the data and have the certainty that they have not been tampered with. Moreover, the technology enables sensor data traceability and accountability. Reliability is the key aspect of Blockchain to bring in the IoT.
- **security:** information and communications can be secured if they are stored as transactions of the blockchain [100]. Blockchain can treat device message exchanges as transactions, validated by smart contracts, in this way securing communications between devices. Current secure standard-protocols used in the IoT can be optimised with the application of Blockchain [258].
- **market of services:** blockchain can accelerate the creation of an IoT ecosystem of services and data marketplaces, where transactions between peers are possible without authorities. Microservices can be easily deployed and micro-payments can be safely made in a trustless environment [108, 109, 110]. It would improve IoT interconnection and the access of IoT data in blockchain.
- **secure code deployment:** taking advantage of blockchain secure-immutable storage, code can be safely and securely pushed into devices [112, 259]. Manufacturers can track states and updates with the highest confidence [100]. IoT middlewares can use this functionality to securely update IoT devices.

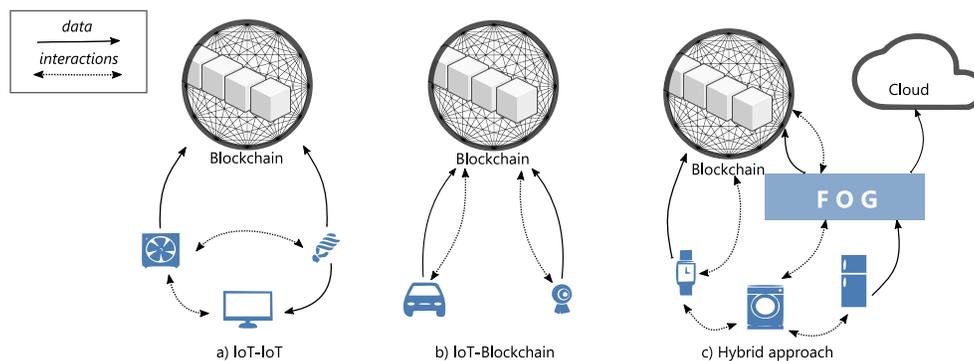


Figure 6.1: Blockchain IoT interactions

Another aspect to take into account is related to the IoT interactions, i.e., the communication between the underlying IoT infrastructure. When integrating Blockchain, it needs to be decided where these interactions

will take place: inside the IoT, a hybrid design involving IoT and Blockchain, or through Blockchain. Fog computing [260] has also revolutionised the IoT with the inclusion of a new layer between cloud computing and IoT devices and could also facilitate this integration. Below, these alternatives (shown in Figure 6.1) are described together with their advantages and disadvantages:

- **IoT-IoT:** this approach could be the fastest one in terms of latency, and security since it can work offline. IoT devices have to be able to communicate with each other, which usually involves discovery and routing mechanisms. Only a part of IoT data is stored in Blockchain whereas the IoT interactions take place without using the blockchain (Figure 6.1.a). This approach would be useful in scenarios with reliable IoT data where the IoT interactions are taking place with low latency.
- **IoT-Blockchain:** in this approach all the interactions go through Blockchain, enabling an immutable record of interactions. This approach ensures that all the chosen interactions are traceable as their details can be queried in the blockchain, and moreover it increases the autonomy of IoT devices. IoT applications that intend to trade or rent such as Slock.it can leverage this approach to provide their services. Nevertheless, recording all the interactions in Blockchain would involve an increase in bandwidth and data, which is one of the well-known challenges in Blockchain (Figure 6.1.b). On the other hand, all IoT data associated with these transactions should also be stored in Blockchain.
- **Hybrid approach:** lastly, a hybrid design where only part of the interactions and data take place in the Blockchain and the rest are directly shared between the IoT devices. One of the challenges in this approach is choosing which interactions should go through the Blockchain and providing the way to decide this in run time. A perfect orchestration of this approach would be the best way to integrate both technologies since it leverages the benefits of Blockchain and the benefits of real-time IoT interactions. In this approach fog computing could come into play and even cloud computing, to complement the limitations of Blockchain and the IoT. For example, fog computing involves fewer computationally-limited devices such as gateways and it is a potential place where mining can take place in the same way as other initiatives that use IoT devices [261, 262] (Figure 6.1.c).

In a typical IoT deployment, limited-resource devices are used as end nodes that communicate with a gateway that is responsible for forwarding sensor data to upper layers (cloud or server). When integrating Blockchain, if end nodes have to interact with the blockchain, cryptographic functionality could be provided in IoT devices. This is key to realise IoT autonomy, which comes at a cost of a more sophisticated hardware and higher computational expense (in Section 6.5 an evaluation of the cost of using Blockchain in IoT devices is presented). Integrating gateways in these solutions is also plausible, in the same way as traditional deployments do, however the benefits of using Blockchain in this way are fewer. Despite the expansion of Blockchain, it would make no sense to use it in many applications, where databases provide enough functionality. Deciding when it is worth using a blockchain will mainly depend on the requirements of the application. For instance, when high performance is required, Blockchain alone may not be the right solution, but a hybrid approach could be applied to optimise it. In [263] the authors present a methodology to identify whether

a blockchain is useful depending on problem requirements. Next we focus on how this can be technically achieved for IoT devices.

Alliances between well-known companies have started to appear, like the Trusted IoT Alliance [238], to bridge the gap between the IoT and Blockchain. Also there are an increasing number of devices with integrated blockchain capabilities available on the market such as [261, 262, 264]. EthEmbedded [261] enables the installation of Ethereum full nodes on embedded devices such as Raspberry Pi, Beaglebone Black and Odroid. Raspnode [262] and Ethraspbian [265] both support the installation of Bitcoin, Ethereum and Litecoin full nodes on a Raspberry Pi. Antrouter R1-LTC [264] is a Wi-Fi router that also enables mining Litecoin. Therefore, this type of router could be installed in smart homes and be part of a fog computing ecosystem. Raspnode also enables wallet support for Bitcoin and Litecoin. As stated by Raspnode, mining can be done on IoT devices, but it would be useless. Mining has become a specialised task for certain hardware (ASIC chips) and it is useless to try it on IoT devices. That is the main reason why there is little work on mining for IoT devices. There is still a lot of research to be done for enabling a wide integration of IoT devices as blockchain components. Table 6.2 shows a summary of the surveyed IoT devices to be used as part of blockchain platforms.

Full nodes must store the entire blockchain (currently more than 150 and 46 GB in Bitcoin and Ethereum, respectively) for a full validation of transactions and blocks, thus their deployment can be very limited in IoT devices. As stated, mining would be useless in the IoT due to its specific requirements. The consensus protocol could be relaxed to facilitate the inclusion of IoT devices, however this could compromise the security of the blockchain implementation. This could be used in consortium blockchains where consensus protocols are relaxed. In the deployment of lightweight nodes the authenticity of the transactions is validated without having to download the entire blockchain, therefore they can contribute to the blockchain and are easy to run and maintain in IoT devices. These nodes can be used in the IoT to be part of the blockchain network, reducing the gap between the two technologies. This always has to be backed up with full nodes to validate transactions and blocks. Nevertheless, many blockchains do not yet provide support for lightweight nodes as is the case of Ethereum, which it is under development. In any case, Blockchain could be used as an external service to provide a secure and reliable storage.

One clear alternative to the integration of Blockchain with the IoT is the integration between the IoT and cloud computing (Chapter 5). This integration has been used in the last few years to overcome the IoT limitations of: processing, storage and access. However, cloud computing usually provides a centralised architecture, which in contrast to Blockchain, complicates reliable sharing with many participants. The integration between Blockchain and the IoT is intended to address previous limitations in addition to maintaining reliable data. Fog computing aims to distribute and bring the computing closer to end devices, following a distributed approach like Blockchain. This can incorporate more powerful devices than the IoT such as gateways and edge nodes, which can then be reused as blockchain components. Therefore, fog computing could ease the integration of the IoT with Blockchain.

Table 6.2: IoT devices to be used as blockchain components

Source	IoT device	Mode	Blockchain
EthEmbedded	Raspberry Pi	full node	Ethereum
	BeagleBone Black		
	Odroid XU3/XU4		
	Wandboard		
	Ethcore Parity		
Ethraspbian	Raspberry Pi	light node	Bitcoin
Raspnode			
Bitmain	Antrouter R1-LTC	miner	

6.3.1 Challenges in Blockchain-IoT Integration

This Section studies the main challenges to be addressed when applying blockchain technology to the IoT domain. The integration of blockchain technology with the IoT is not trivial. Blockchain was designed for an Internet scenario with powerful computers, and this is far from the IoT reality. Blockchain transactions are digitally signed, and therefore devices capable of operating with the currency must be equipped with this functionality. Incorporating blockchain into the IoT is challenging. Some of the identified challenges are presented in this section.

6.3.1.1 Storage Capacity and Scalability

As stated, storage capacity and scalability of Blockchain are still under debate, but in the context of IoT applications the inherent capacity and scalability limitations make these challenges much greater. In this sense, Blockchain may appear to be unsuitable for IoT applications, however there are ways in which these limitations could be alleviated or avoided altogether. In the IoT, where devices can generate gigabytes of data in real time, this limitation represents a great barrier to its integration with Blockchain. It is known that some current blockchain implementations can only process a few transactions per second, so this could be a potential bottleneck for the IoT. Furthermore, Blockchain is not designed to store large amounts of data like those produced in the IoT. An integration of these technologies should deal with these challenges.

Currently, a lot of IoT data are stored and only a limited part is useful for extracting knowledge and generating actions. In the literature different techniques to filter, normalise and compress IoT data with the aim of reducing them have been proposed. The IoT involves embedded devices, communication and target services (Blockchain, cloud), thus savings in the amount of data that the IoT provides can benefit multiple

layers. Data compression can lighten transmission, processing tasks and storage of the high volume of IoT data generated. Normal behaviours do not usually require extra, necessary information, unlike anomalous data.

Last but not least, Blockchain, and especially its consensus protocol which causes its bottleneck, could also be adapted to increase the bandwidth and decrease the latency of its transactions thereby enabling a better transition to the IoT as demonstrated by the case of Bitcoin-NG [193].

6.3.1.2 Security

IoT applications have to deal with security problems at different levels, but with an additional complexity due to the lack of performance and high heterogeneity of devices. In addition, the IoT scenario comprises a set of properties that affect security, such as mobility, wireless communication or scale. An exhaustive analysis of security in IoT is beyond the scope of this work but detailed surveys can be found in [266, 183, 267, 268].

The increasing number of attacks on IoT networks, and their serious effects, make it even more necessary to create an IoT with more sophisticated security. Many experts see Blockchain as a key technology to provide the much needed security improvements in IoT. However, one of the main challenges in the integration of the IoT with Blockchain is the reliability of the data generated by the IoT. Blockchain can ensure that data in the chain are immutable and can identify their transformations, nevertheless when data arrives already corrupted in the blockchain they stay corrupt. Corrupt IoT data can arise from many situations apart from malicious ones. The well-being of the IoT architecture is affected by many factors such as the environment, participants, vandalism and the failure of the devices. Sometimes the devices themselves and their sensors and actuators fail to work properly from the start. This situation cannot be detected until the device in question has been tested, or sometimes it works properly for a while and changes its behaviour for some reason (short circuit, disconnection, programmed obsolescence, and so on). In addition to these situations, there are many threats that can affect the IoT such as eavesdropping, DoS or controlling [183]. For that reason, IoT devices should be thoroughly tested before their integration with Blockchain and they should be located and encapsulated in the right place to avoid physical damage, in addition to including techniques to detect device failures as soon as they happen.

These devices are more likely to be hacked since their constraints limit the firmware updates, preventing them from actuating over possible bugs or security breaches. Moreover, it is sometimes difficult to update devices one by one, as in global IoT deployments. Therefore, run-time upgrading and reconfiguration mechanisms should be placed in the IoT to keep it running over time. Initiatives such as GUITAR [32] and REMOWARE [33] enable network and firmware updates in run-time and are essential to ensure a secure integration of the IoT with Blockchain over time.

The IoT and Blockchain integration can also have repercussions on the IoT communications [258]. Currently, IoT application protocols such as CoAP and MQTT make use of other security protocols such as TLS or DTLS to provide secure communications. These secure protocols are complex and heavy in addition to requiring a centralised management and governance of key infrastructure, typically with PKI. In the blockchain network each IoT device would have its own global unique identifier and asymmetric key pair installed once

connected to the network. This would simplify current security protocols which usually have to exchange PKI certificates and would allow them to be used in devices with lower capabilities.

One notable IoT project in terms of security with a blockchain adoption is Filament [257]. Filament is a hardware and software solution that provides functionality for Bitcoin-based payments and smart contracts in IoT. Filament devices have embedded cryptoprocessors that support five protocols: Blockname, Telehash and smart contracts to operate, and additionally Pennyback and Bittorrent protocols. The device identity management is done with Blockname, while Telehash, an open-source implementation of Kademia DHT, provides secure encrypted communications, and smart contracts define the way in which a device can be used.

6.3.1.3 Anonymity and Data Privacy

Many IoT applications work with confidential data, for instance when the device is linked to a person, such as in the e-health scenario, it is essential to address the problem of data privacy and anonymity. Blockchain is presented as the ideal solution to address identity management in IoT, however as in Bitcoin, there may be applications where anonymity needs to be guaranteed. This is the case of a wearable with the ability to hide the identity of the person when sending personal data, or smart vehicles that safeguard the privacy of the itineraries of users.

The problem of data privacy in transparent and public blockchains has already been discussed, together with some of the existing solutions. However, the problem of data privacy in IoT devices entails more difficulty, since it starts at data collection and extends to the communications and application levels. Securing the device so that data are stored securely and not accessed by people without permission is a challenge since it requires the integration of security cryptographic software into the device. These improvements should take into account the limitation of resources of the devices and the restrictions related to economic viability. Many technologies have been used to secure communications using encryption (IPsec, SSL/TLS, DTLS). IoT device limitations often make it necessary to use less-constrained devices such as gateways to incorporate these security protocols. The use of cryptographic hardware could accelerate cryptographic operations and avoids the overload of complex secure software protocols.

Protection of data and privacy are key challenges for IoT, using blockchain technology the problem of identity management in IoT can be alleviated. Trust is another key feature of the IoT where the integration of Blockchain can play a role. In [269] the importance of trust in IoT systems is identified as one of the primary goals to ensure its success. Data integrity techniques are another option to ensure data access at the same time as they avoid overloading Blockchain with the huge amount of data generated by the IoT. This can result in public systems, but with an efficient and restricted access control. MuR-DPA [270] provides dynamic data updates and efficient verification through public auditing verification. In [271] the authors ensure the data content through another privacy-preserving public auditing system. For a broad review of integrity verification techniques, refer to [272].

Last but not least, there are laws that regulate data privacy, such as the EU's data protection directives that will need to be revised to cover the new models that the technology makes possible. The adoption of

Blockchain as a legal platform should address these regulations so as to ensure data privacy following the law.

6.3.1.4 Smart contracts

Smart contracts have been identified as the killer application of blockchain technology, but as mentioned there are several challenges yet to be tackled. IoT could benefit from the use of smart contracts, however the way they fit into IoT applications is diverse.

From a practical point of view, a contract is a collection of code (functions) and data (states) that reside in a specific blockchain address. Public functions in a contract can be called by devices. Functions can also fire events, applications can listen for them in order to properly react to the event fired. To change the state of the contract, that is, to modify the blockchain, a transaction has to be published in the network. Transactions are signed by senders and have to be accepted by the network.

The IoT has the ability to sense and actuate over the Internet in many areas [2]. In the food traceability example, food packaging would be equipped with sensors with the capability to measure environmental conditions and connect to the blockchain (sign transactions). In the blockchain a contract would provide functions to start shipping, finish shipping and log and query measurements. When measurements exceeded a predefined threshold, an event would be fired. Management applications would be listening to these events, and the shipping company, retailers, manufacturers and clients would be informed. If no events were raised, then the blockchain would guarantee that the shipment was carried out in optimal conditions. Smart contracts would provide a secure and reliable processing engine for the IoT, recording and managing all their interactions. Actions would be the result of a reliable and secure processing. Therefore, smart contracts can securely model the application logic of IoT applications. However, the following challenges should be addressed in that integration.

On the one hand, working with smart contracts requires the use of oracles which are special entities that provide real-world data in a trusted manner. Validating these smart contracts could be compromised since the IoT can be unstable. Moreover, accessing multiple data sources could overload these contracts. Nowadays, smart contracts are distributed and decentralised, but they do not share resources to distribute tasks and address a large amount of computation. In other words, execution of smart contracts is done in just a single node whereas simultaneously the code execution is done by multiple nodes. This distribution is only done for the validation process, instead of using it to distribute tasks. The IoT has leveraged the distributed capabilities of cloud computing and big data to increase its processing power. Since then, data mining techniques have been able to address the IoT data as a whole, enabling a better understanding of the IoT, i.e., the processing power enlarged by cloud computing. Big data has enabled the processing of large amounts of data simultaneously, allowing knowledge to be extracted from large datasets, which was previously very hard to do. In the integration of IoT with Blockchain, smart contracts should leverage their distributed nature to enable the processing capabilities provided in other paradigms (big data and cloud computing) and needed in the IoT.

Smart contracts should also take into account the heterogeneity and constraints present in the IoT. Filtering and group mechanisms should be complemented by smart contracts to enable applications to address the

IoT depending on the context and requirements. A discovery mechanism could enable device inclusion on the fly, making these applications more powerful. Lastly, actuation mechanisms directly from smart contracts would enable faster reactions with the IoT.

6.3.1.5 Legal issues

The vision of an unregulated Blockchain is part of its essence, and partly responsible for the success of Bitcoin. As seen, Blockchain, specifically in the context of virtual currencies, has brought with it a lot of controversy regarding legality. The need, or opportunity, to introduce control elements over the network has come in the form of permissioned, private and consortium blockchains.

The IoT domain is also affected by a country's laws or regulations regarding data privacy, for instance the data protection directive. Most of these laws are becoming obsolete and need to be revised, especially since the emergence of new disruptive technologies such as Blockchain. The development of new laws and standards can ease the certification of security features of devices, and in this way help build the most secure and trusted IoT network. In this sense, laws that deal with information privacy and information handling are still a big challenge to be tackled in IoT and will therefore be an even bigger challenge if used in combination with Blockchain.

As stated, the lack of regulation creates disadvantages, because mechanisms for private key retrieval or reset, or transaction reversion are not possible. Some IoT applications envision a global, unique Blockchain for devices, however it is unclear if this type of network is intended to be managed by manufacturers or open to users. In any case, it is expected it will require legal regulation. These regulations will have an influence on the future of Blockchain and IoT and as such could possibly disrupt the decentralised and free nature of Blockchain by introducing a controlling, centralised participant such as a country.

6.3.1.6 Consensus

In the context of IoT applications, the limited-resource nature of devices makes them unsuitable for taking part in consensus mechanisms, such as PoW, directly. As stated, there are a wide variety of proposals for consensus protocols, although they are, in general, immature and haven't been tested enough. Resource requirements depend on the particular type of consensus protocol in the blockchain network. Typically, solutions tend to delegate these tasks to gateways, or any other unconstrained device, capable of providing this functionality. Optionally off-chain solutions, which move information outside the blockchain to reduce the high latency in Blockchain, could provide the functionality.

Although there are initiatives to incorporate blockchain full nodes into IoT devices [261, 262], mining is still a key challenge in the IoT due to its limitations. IoT is mainly composed of resource-constrained devices but globally the IoT has a potentially huge processing power, taking into account that it is expected that the number of devices in it will reach anywhere between 20 and 50 billion by 2020. Research efforts should focus on this field and leverage the distributed nature and global potential of the IoT to adapt the consensus in the IoT.

In Babelchain [273] a novel consensus protocol called Proof of Understanding that aims to adapt PoW for IoT applications is proposed. With less energy consumption, the protocol, instead of using miners to solve hash puzzles, proposes translating from different protocols. In this way the effort is more concentrated on useful computation while simultaneously tackling a key problem in IoT communications. Peers in the network instead of agreeing on transaction status, agree on message meaning (format, content and action). Additionally, the blockchain data provide information to learn, like a learning set.

6.4 Blockchain platforms for IoT

Blockchain has been identified as a disruptive technology that can strongly affect many industries. The number of platforms is so high and in constant change that it is impossible to analyse them all, in this section we focus on the most popular and most suitable for IoT domains.

Bitcoin was the first cryptocurrency and the first blockchain platform. It provides a mechanism to carry out monetary transactions in a fast, cheap and reliable way, which can be integrated into applications as a secure payment system. In IoT domain, autonomous devices can use Bitcoins to perform micro-payments, working mainly as wallets. In general when the use of Blockchain is limited to micro-payments, applications are attached to the currency, which can be a drawback, since depreciation of the coin can negatively affect the application. As stated, using smart contracts is a common solution when integrating Blockchain with IoT. Bitcoin includes a scripting language that allows specific conditions to be set when carrying out transactions. However, the scripting is quite limited compared with other smart contract platforms.

As mentioned one of the platforms that has had an important impact in recent times is Ethereum [111]. Ethereum was one of the pioneer Blockchains in including smart contracts. Ethereum can be described both as a Blockchain with a built-in programming language (Solidity), and as a consensus-based virtual machine running globally (Ethereum Virtual Machine). The inclusion of smart contracts moves the Blockchain away from currencies and facilitates the integration of this technology in new areas. This along with its active and broad community makes Ethereum the most popular platform for developing applications. Most IoT applications use Ethereum or are compatible with it (see Table 2.3). The simplest approach is to define a smart contract where devices can publish their measures and policies that react to changes.

Hyperledger [191] has also had a great impact. Hyperledger is an open-source platform on which various projects related to Blockchain have been developed, among them Hyperledger Fabric, a Blockchain deprived of permissions and without cryptocurrency on which commercial implementations like IBM's Blockchain platform are based. It provides different components for consensus and membership. Distributed application can be developed in the blockchain using general purpose languages. IoT devices can supply data to the blockchain through the IBM Watson IoT Platform, which manages devices and allows data analysis and filtering. IBM's Bluemix platform eases the integration of blockchain technology by offering it as a service. The use of this platform speeds up application prototyping, and several use cases have been developed. There is an ongoing project on food traceability that uses this platform [75].

The Multichain platform allows the creation and deployment of private blockchains. Multichain uses

Table 6.3: Blockchain platforms for creating blockchain applications

Platform	Blockchain	Consensus	Cripto currency	Smart contracts
Ethereum	Public and permission-based	PoS	Ether (ETH)	yes
Hyperledger Fabric	Permission-based	PBTF/SIEVE	None	yes
Multichain	Permission-based	PBTF	Multi-currency	yes
Litecoin	Public	Scrypt	litecoins (LTC)	no
Lisk	Public and permission-based	DPoS	LSK	yes
Quorum	Permission-based	Multiple	ETH	yes
HDAC	Permission-based	ePoW, Trust-based	Multiasset	yes

an API that extends the core of the original Bitcoin API with new functionality, allowing the management of portfolios, assets, permissions, transactions, etc. In addition, it offers a command-line tool for interacting with the network, and different clients that can interact through JSON-RPC with the network such as Node.js, Java, C # and Ruby. Multichain is a fork of Bitcoin Core, its source code compiles for 64 bit architectures. In [274] multichain blockchain cluster is deployed on three nodes, one of them an Arduino board, as a proof of concept of an IoT blockchain application.

Litecoin [67], as stated, is technically identical to Bitcoin, but features faster transaction confirmation times and improved storage efficiency thanks to the reduction of the block generation time (from 10 minutes to 2.5) and the proof of work, which is based on scrypt, a memory intensive password-based key derivation function. This means that the computational requirements of Litecoin nodes are lower, so it is more suitable for the IoT.

Lisk [275] offers a blockchain platform in which sub-blockchains or sidechains can be defined with decentralised blockchain applications and a choice of cryptocurrencies to use (e.g. Bitcoin, Ethereum, etc.). Known as the blockchain platform for JavaScript developers, Lisk also offers support to create and deploy decentralised applications within the platform to be used directly by end users, creating an ecosystem of interoperable blockchain services. The applications developed can use LSK currency or can create custom

tokens. Lisk uses Delegated proof of stake consensus. Lisk is working with Chain of Things to examine whether blockchain technology can be effective in establishing security within the IoT.

Quorum [218] is a blockchain platform developed to provide the financial services industry with a permissioned implementation of Ethereum with support for transaction and contract privacy. It allows multiple consensus mechanisms and achieves data privacy through cryptography and segmentation. The platform has recently integrated ZeroCash technology to obscure all identifiable information about a transaction. The Quorum platform has been used by Chronicled [115] to create secure links between physical assets and Blockchain.

HDAC [252] is an IoT contract and M2M transaction platform based on Blockchain currently under development. The HDAC system uses a combination of public and private blockchains, and quantum random number generation to secure these transactions. The HDAC cryptocurrency-enabled public blockchain can be effectively used with multiple private blockchains. HDAC IoT contract proof of concept will be launched this year.

Table 6.3 shows a comparison of the blockchain platforms for creating IoT applications surveyed in this section. Smart contracts are present in most platforms, thus enabling application logic beyond cryptocurrency transactions. In the case of a Blockchain deployment, there is a dualism among Blockchains with and without a cryptocurrency. An established platform with a cryptocurrency like Ethereum can provide the needed infrastructure for a blockchain application. This can be seen as deploying your own cloud infrastructure or using AWS Amazon or Google Cloud Platform. However, in the case of Blockchain the distribution is the linchpin of its trustworthiness. On the other hand, using a non-cryptocurrency platform like Hyperledger Fabric requires joining a consortium and infrastructure to start a Blockchain. PBTF and PoS are the most used consensus. Permissions and privacy are in most platforms, therefore consortium and global applications can be created from them.

6.5 Evaluation

To test the viability of running blockchain platforms on IoT devices we installed and ran different nodes from different platforms on a Raspberry Pi 3 model B with Raspbian Stretch OS and Linux kernel 4.9.59-v7+. To perform these experiments the device was equipped with an SD card of 128GB (Samsung EVO PLUS MB-MC128GA/EU). Raspberry was connected to the Internet through Ethernet. The different types of nodes were installed and run, connected to the public network. We measured energy consumption and collected some measurements related to task performance for 30 minutes for each test. The energy consumption was obtained by measuring the current between the power source and the Raspberry Pi, with a Moteino Mega equipped with the INA219 DC current sensor. The results were obtained on a PC through a serial connection with the Moteino Mega. To evaluate the task performance, a background process was run collecting the following task data: percentage of use of CPU and memory, amount of virtual memory required by the task, and the bandwidth consumption (sent and received bytes). This was done with the top (Top command: `https://linux.die.net/man/1/top`) Unix command and the monitor per process network bandwidth usage

Table 6.4: Blockchain nodes evaluation on Raspberry Pi v3 (Synchronising (s), after synchronisation (as))

	avg en- ergy (mA)	avg %CPU	avg %Mem	avg VIRT (MB)	Total sent MB	Total received MB
Bitcoin light node	283.45	2.19	9.6	288	0.074	0.068
Litecoin light node	275.53	2.05	7.6	209	0.081	0.104
Ethereum full node (s)	599.72	256.82	85.96	1490	78.9	490
Ethereum full node (as)	371.26	47.39	29.36	1280	15.3	80.3
Bitcoin full node (s)	429.05	55.47	27.55	405	48	912
Litecoin full node (s)	437.62	117.80	19.36	341	17.1	546
Litecoin full node (as)	280.04	7.01	51.7	679	1.22	2.06

tool nethogs (Nethogs command: <https://github.com/raboof/nethogs>). Table 6.4 summarises the results. CPU measurement is 100% for each core, so the maximum for our quad core device is 400%.

Note that the different types of nodes implement different functionalities, and the same types of nodes in different blockchains also perform different tasks. Nodes were connected to public blockchain networks such as Bitcoin, Ethereum and Litecoin to test the device in real working conditions. In addition, when connected to the public network we had no control of the status of the network, i.e., the number of transactions published during each experiment were not the same. Therefore, this evaluation is not intended to be an exhaustive comparison of the performance of the different implementations, but rather to provide an idea of the viability of using blockchain nodes on IoT and to state technical difficulties.

The nodes chosen for the experiment were those with Raspberry Pi compatibility that run in public networks, so the nodes can take part of the routing protocol. To be able to compare the results we also measured the energy consumption of the Raspberry running Raspbian and without any node installed, obtaining 272.09 mA. Raspberry Pi is powered with 5V.

First, we measured the performance of light nodes. The Bitcoin light node (Electrum implementation), for instance, increased the energy consumption by less than 4%, with respect to the Raspberry base measurement (272.09 mA). Very similar to the values obtained for the Litecoin light node (Electrum implementation) which reached a 5% increase. The Ethereum light node was not tested because it was under development. Regarding the task performance, the CPU usage and virtual memory were also quite similar for both light nodes. There was more difference in the use of memory, where Bitcoin consumed 2% additional memory. Bandwidth consumption mainly differed in received bytes, where Litecoin downloaded almost 36Kb more. Results show that these light nodes run comfortably on the Raspberry Pi, they provide the functionality to make secure transactions, which is enough to benefit from blockchain features, and do not require an excessive amount memory, nor high energy consumption.

Next the experiments were performed with full nodes: the Ethereum full node (Go implementation), the

Bitcoin full node (Bitcoin core implementation) and the Litecoin full node (Litecoin core implementation). As stated, the functionality of the Ethereum full node is not the same as the Bitcoin one. First of all, the different consensus protocols, PoS and PoW, have radically different computational requirements. In addition, full nodes require a synchronisation process to download the full chain, during this process the computational requirements are expected to be higher. To evaluate the different consumption requirements, measurements were taken during the synchronisation and after it. At the moment of testing Litecoin's blockchain size was 14.03 GB, 49.47 GB for the Ethereum chain, and 155.8 GB for the Bitcoin chain. Note that the full Bitcoin node after the synchronisation could not be tested since the current Bitcoin chain size exceeds the memory of our test scenario. The Ethereum synchronisation took about 5 days (with several reboots, due to system failures), and Litecoin about 2 days. Moreover, after a couple of days of being close to a full synchronisation (fewer than 100 blocks to reach it), we decided to evaluate the Ethereum full node in its current state, 99.99% for a full synchronisation. We considered that 99.99% is a good approximation for a full synchronisation. The number of new blocks generated and the limited capabilities of the Raspberry Pi for that full node made it difficult to reach a full synchronisation.

When compared with light nodes, the additional functionality that full nodes implement, which is mainly the storage, is observed in higher consumption values, both in memory and CPU. Most values obtained after the synchronisation were lower than during it, as expected. In the Ethereum full node after synchronising, the average CPU used was 5 times lower, and average memory use 2.8 times lower, with respect to the values during the synchronisation.

The energy consumption, when compared with the Raspberry base measurement (272.09 mA), increased by 120% in the Ethereum node during synchronisation, whereas Bitcoin full node does it with 57% and Litecoin with 60%. These values after the synchronisation were 36% increase for Ethereum and less than 3% for Litecoin. The use of CPU and memory also rose in full nodes, especially during the synchronisation, the highest consumption values were the Ethereum node that reached an average of 256% of CPU usage and 86% of memory. However, after the synchronisation those values decreased to 47% and 29% respectively. For Litecoin, CPU usage was 117% during synchronisation and only 7% after synchronisation. Block validation is CPU intensive, so the CPU usage drop is normal after the chain synchronisation. In contrast memory usage in Litecoin increased from 19% to 51% after synchronisation. This is probably due to its memory intensive consensus protocol. As expected bandwidth values soared, when compared with light nodes, since they stored the full chain. Whereas for light nodes bandwidth values were in the order of KB, for full nodes were in the order of Mb. Bitcoin full node received 13 thousand times more bytes than its light node, and the Litecoin full node during synchronisation, 5 thousand times more than its light version, and barely 20 times more after synchronisation.

There are very few scenarios where it makes sense to use a full node IoT. We haven't found any application that uses it in IoT. It could be used in private deployments and for test purposes, but we understand that in final applications there would be other more suitable devices to run these nodes. The evaluation results show that the feasibility of using these nodes on the IoT is very limited. As stated, light nodes are a far better fit for limited resource devices such as those in the IoT.

The most noteworthy problems during the tests were the difficulties in finding up-to-date information about the different node implementations on the pages found for IoT devices, along with the high installation times in some cases and very high synchronisation times for full nodes. Moreover, the correct configuration for IoT devices has been necessary in some cases, such as Ethereum, otherwise the full node would collapse the Raspberry. The official documentation and implementations have been very useful for the deployment of the aforementioned nodes.

7

Conclusions and Future Work

This chapter concludes this thesis summarising the conclusions and future work suggested by the contributions that support it. The organisation of this last chapter is as follows. Section 7.1 presents the conclusions. Future work is highlighted in Section 7.2. Then, the publications that support this thesis and have been produced during the course of this research are shown in Section 7.3. Finally, the funding that has made this research possible is detailed in Section 7.4.

7.1 Conclusions

The IoT is a ubiquitous technology that is present in many situations of our everyday life. From smart homes to manufacturing, the IoT is improving the quality of life and optimising many services through its capabilities to sense and actuate over the physical world. Several research reports prognosticate a huge evolution of this paradigm in coming years, so its adoption in our society will be more noticeable. However, the IoT has serious limitations due to the resource-constrained nature of its devices, which drags out that expansion. This thesis, *IntegraDos*, has aimed to provide different application enablers to facilitate the adoption of the IoT through its integration with different technologies and paradigms. In particular, we have researched how to facilitate the run-time management and deployment of physical components, the development of IoT applications, an integration of the IoT with cloud computing, and how the IoT can be integrated with Blockchain has been analysed and evaluated.

In Chapter 3, a system for run-time managing and deploying physical resources in microcontrollers is presented. The goal of this solution is to reduce the gap between end users and customisable environments. Hereafter, end users can install their own sensors and actuators on their microcontrollers through well-defined interfaces without rebooting and programming these systems. The Constrained Application Protocol (CoAP) and ZigBee have been used as the embedded web service and communication means allocated in the microcontrollers, thereby enabling a lightweight and low-power approach for interacting with resources. The system provides a proxy to interconnect HTTP with CoAP in a Smart Gateway, in such a way that any HTTP client can carry out the operations provided by the Smart Gateway. In fact, the latter also contributes to helping the IoT towards the Web of Things. In addition, a Web UI has been incorporated to provide multi-device access to end users. Storage and temporal limitations have been taken into account in the design, so that the system provides objective middlewares and temporal behaviours. The results show the feasibility of the system to incorporate new physical libraries to be managed at run-time.

Appdaptivity, a framework that enables the development of IoT applications abstracting application developers from the underlying physical infrastructure is presented in Chapter 4. The proposed solution reduces the gap between IoT application and the underlying physical infrastructure, which is actually highly coupled and carried to multiple vertical silos. The resulting applications are created intuitively, personalised, portable and automatically adapted to changing contexts without recompiling and programming device-coupled applications. The Appdaptivity programming model is based on data flow programming, where application developers can create IoT applications connecting components through a Web UI. Application developers need only to focus on the application logic, whereas the aspects of discovery and injecting a device's information into applications are provided by Appdaptivity. The solution is intended for heterogeneous scenarios, enabling the portability of IoT applications, various multi-deployment options and the integration with external user applications. The underlying physical domain is supported through CoAP. Lastly, the final applications that will be used by end users, known as user applications, are not targeted to a specific application, as Appdaptivity enables the integration of multiple applications and platforms through its well-defined API and standards protocols. The evaluation was performed in the different scenarios that can be part of Appdaptivity as secure and non-secure environments. The evaluation results confirm a good scalability of the system with

respect to the user applications and the underlying physical infrastructure. Moreover, this work has also taken into account an evaluation in an IoT area. A smart city use case has been modelled using Appdaptivity and has been deployed with IoT devices and resource-constrained networks. The data flow programming model used in Appdaptivity allows multiple configurations, or behaviours such as those discovered in this approach, with only a few components while at the same time it enables the application development in a easy and non-technical way.

The λ -CoAP architecture, an architecture to integrate the IoT and cloud computing with an edge computing layer is presented in Chapter 5. The λ -CoAP architecture contributes to solving the problems of processing, analysing and storing large amounts of IoT data through a paradigm known as the Lambda Architecture. CoAP is the protocol used in the things side, hence its name, λ -CoAP. Some components and the expertise from Chapters 3 and 4 have strongly contributed to the composition of this architecture. The architecture covers the entire scope of an edge computing deployment, from the IoT, going through the edge, to the cloud. The λ -CoAP architecture incorporates a CoAP middleware so as to abstract the low-level aspects of the IoT devices and provides a lightweight way to actuate with the IoT. Moreover, the IoT is abstracted through Smart Gateways, allowing the access to it through different high level protocols. A lightweight virtualisation technology has been adopted in the Smart Gateways to give a better control and isolation of the running processes in addition to a visual framework to develop the edge logic through data flow programming. Lastly, the Lambda architecture deployed in the cloud enables the processing and storing of large amounts of IoT data. IoT applications such as structural health monitoring could leverage this architecture to reduce their latency and alleviate the bandwidth to the cloud. The evaluation confirms a good scalability of the architecture for a large IoT deployment, and how the edge computing can reduce the bandwidth and latency in the IoT-cloud communications.

Blockchain was conceived as a revolutionary technology to keep data reliable. However, modifying this technology without adequately guaranteeing its operation or applying it to scenarios where the cost does not compensate the improvement are risks into which one can fall easily. Therefore, the benefits of applying Blockchain to the IoT should be analysed carefully and taken with caution. Finally, Chapter 6 has provided an analysis of the main challenges that Blockchain and IoT must address in order for them to successfully work together. The key points where blockchain technology can help improve IoT applications have been identified. An evaluation has also been provided to prove the feasibility of using blockchain nodes on IoT devices. Existing platforms have also been examined to complete the study, offering a complete overview of the interaction between blockchain technology and the IoT paradigm. It is expected that Blockchain will revolutionise the IoT. The integration of these two technologies should be addressed, taking into account the challenges identified. The adoption of regulations is key to the inclusion of Blockchain and the IoT as part of government infrastructures. This adoption would speed up the interaction between citizens, governments and companies. Consensus will also play a key role in the inclusion of the IoT as part of the mining processes and distributing even more blockchains. Nevertheless, a dualism between data confidence and facilitating the inclusion of embedded devices could arise. Lastly, beyond the scalability and storage capacity which affect both technologies, research efforts should also be made to ensure the security and privacy of critical

technologies that the IoT and Blockchain can become.

7.2 Future Work

This research has led us to define several open challenges that should be addressed. Firstly, the run-time management and deployment system could ensure the compatibility of the system with different wireless communication technologies for resource-constrained devices in order to evaluate them and establish the most suitable for each environment. A run-time deployment and management of them in the same way that until now has been done with physical resources would enable the use of multiple and configurable wireless technologies over CoAP. Wireless communication technologies, like the IoT, are in a continuous expansion. Noted examples are the new generation of wireless technologies Low-Power Wide-Area Networks (LPWAN), such as Lora and Sigfox. Nevertheless, each technology has its own unique characteristics in terms of coverage, connectivity, security and power consumption, so each is suitable for certain environments. This approach should also take into account the efforts being made in enabling IPv6 connectivity over LPWA networks. The adoption of an upper-layer protocol such as CoAP over these technologies would also ensure the interoperability of them in the IoT.

On the other hand, Appdaptivity can enable multi-profile support in secure communications with the user applications, thereby facilitating the global adoption of the system. Currently, the single-certificate support in secure communications limits the expansion of the system when there are multiple profiles. Moreover, the definition of the secure connection of each CoAP server in the framework used has to be done manually and could be automated. Appdaptivity can also integrate the management of RESTlet components. RESTlet components enable the definition of application logic directly in IoT devices through CoAP. This would optimise the logic positioning between Appdaptivity and the underlying physical infrastructure enabling a distributed intelligence. An integration between the λ -CoAP architecture and Appdaptivity could also be considered.

It is also necessary to evaluate the λ -CoAP architecture in a larger IoT use case. Currently, the evaluation does not cover the IoT scenario that could be. Nevertheless, results confirm the capabilities of the architecture for a larger use case. For that reason, we have recently applied for a research project in collaboration with another research group at the University of Malaga that aims to monitor university spaces, optimising their use and detecting anomalies.

Lastly, the analysis of the challenges and open issues in the integration of Blockchain and IoT has established the guidelines for our future work to overcome this challenge. In the near future, this topic will be addressed with the goal of keeping IoT data and communications reliable.

7.3 Publications

The following subsections list the publications that have been produced during the course of this research (listed by category and in reverse chronological order).

Publications Supporting this Thesis

Tables 7.1, 7.2, 7.3 7.4 shows respectively the ISI-JCR, international conferences, international workshops and national conferences that support this thesis:

ISI-JCR Journals papers

Table 7.1: ISI-JCR Journals publications that support this thesis

Title:	On blockchain and its integration with IoT. Challenges and opportunities
Authors:	Ana Reyna, Cristian Martín , Jaime Chen, Enrique Soler, Manuel Díaz
Journal:	Future Generation Computer Systems
Publication Date:	November 2018
JCR 2017 Impact Factor:	4.639 (Q1)
DOI:	https://doi.org/10.1016/j.future.2018.05.046
Cites on Google Scholar:	11
Title:	Appdaptivity: An Internet of Things Device-Decoupled System for Portable Applications in Changing Contexts
Authors:	Cristian Martín , Jeroen Hoebeke, Jen Rossey, Manuel Díaz, Bartolomé Rubio, Floris Van den Abeele
Journal:	Sensors
Publication Date:	April 2018
JCR 2017 Impact Factor:	2.475 (Q2)
DOI:	https://doi.org/10.3390/s18051345
Cites on Google Scholar:	0
Title:	Run-time deployment and management of CoAP resources for the Internet of Things.
Authors:	Cristian Martín , Manuel Díaz, Bartolomé Rubio
Journal:	International Journal of Distributed Sensor Networks
Publication Date:	March 2017
JCR 2017 Impact Factor:	1.787 (Q2)
DOI:	https://doi.org/10.1177/1550147717698969
Cites on Google Scholar:	3
Title:	State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing.
Authors:	Manuel Díaz, Cristian Martín , Bartolomé Rubio
Journal:	Journal of Network and Computer Applications (JNCA)
Publication Date:	May 2016
Award:	JNCA 2017 Best Survey Paper Award
JCR 2016 Impact Factor:	3.50 (Q1)
DOI:	https://doi.org/10.1016/j.jnca.2016.01.010
Cites on Google Scholar:	212

International Conferences

Table 7.2: International conference publications that support this thesis

Title:	An edge computing architecture in the Internet of Things
Authors:	Cristian Martín , Manuel Díaz, Bartolomé Rubio
Conference:	20th IEEE International Symposium on Real-Time Computing. IEEE ISORC 2018
Place:	June 29-31, 2018. Singapur
Conference Rankings:	Core C (ERA), B1 (Qualis)
DOI:	https://doi.org/10.1109/ISORC.2018.00021
Cites on Google Scholar:	0
Title:	λ -CoAP: An Internet of Things and Cloud Computing Integration Based on the Lambda Architecture and CoAP.
Authors:	Manuel Díaz, Cristian Martín , Bartolomé Rubio
Conference:	11th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2015).
Place:	November 10-11, 2015, Wuhan (China)
Conference Rankings:	Core C (ERA), B3 (Qualis)
DOI:	http://dx.doi.org/10.1007/978-3-319-28910-6_18
Cites on Google Scholar:	6

International Workshops

Table 7.3: International workshop publications that support this thesis

Title:	SocICoAP: Social Interaction with Supplementary Sensors and Actuators through CoAP in Smartphones.
Authors:	Cristian Martín , Jaime Chen, Manuel Díaz, Ana Reyna, Bartolomé Rubio
Conference:	2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)
Place:	4-8 July, 2017, Turin (Italy)
Conference Rankings:	Core B (ERA), A2 (Qualis)
DOI:	http://dx.doi.org/10.1109/COMPSAC.2017.217
Cites on Google Scholar:	0

National Conferences

Table 7.4: National conference publications that support this thesis

Title:	λ -CoAP: An Internet of Things and Cloud Computing Integration Based on the Lambda Architecture and CoAP.
Authors:	Cristian Martín , Manuel Díaz, Bartolomé Rubio
Conference:	XIX Jornadas de Tiempo Real.
Place:	4-8 February, 2016, Malaga (Spain)
Conference Rankings:	-
DOI:	-
Cites on Google Scholar:	-

Further Publications

The publication shown in Table 7.5 has been produced during the course of this dissertation but it is not related to this thesis.

ISI-JCR Journals

Table 7.5: ISI-JCR Journals publications that have been published during this research

Title:	Smart Winery: A Real-Time Monitoring System for Structural Health and Ul- lage in Fino Style Wine Casks
Authors:	Eduardo Cañete, Jaime Chen, Cristian Martín , Bartolomé Rubio
Journal:	Sensors
Publication Date:	March 2018
JCR 2017 Impact Factor:	2.475 (Q2)
DOI:	https://doi.org/10.3390/s18030803
Cites on Google Scholar:	1

7.4 Funding

This thesis has been funded by the Spanish project TIC-1572 (“MIS TICa: Critical Infrastructures Monitoring based on Wireless Technologies”) and a grant for international recognition of a PhD at Málaga University.



Resumen

A.1 Introducción

El Internet de las cosas (IoT, por sus siglas en inglés), fue un nuevo concepto introducido por K. Asthon en 1999 para referirse a un conjunto identificable de objetos conectados a través de RFID (siglas de Radio Frequency IDentification, en español identificación por radiofrecuencia) [3]. Actualmente, el IoT no comprende únicamente el campo de RFID, sino que engloba otras áreas, como redes de sensores inalámbricas (WSN, por sus siglas en inglés). El IoT se caracteriza por ser una tecnología ubicua que está presente en un gran número de áreas, como puede ser la monitorización de infraestructuras críticas, *smart home*, *smart city*, sistemas de trazabilidad o sistemas asistidos para el cuidado de la salud. En la Figura A.1 se muestra un ejemplo de áreas donde el IoT es aplicado. El IoT está cada vez más presente en nuestro día a día, cubriendo un gran abanico de posibilidades con el fin de optimizar los procesos y problemas a los que se enfrenta la sociedad. Es por ello por lo que el IoT es una tecnología prometedora que está continuamente evolucionando gracias a la continua investigación y el gran número de dispositivos, sistemas y componentes emergidos cada día.

Sin embargo, los dispositivos involucrados en el IoT se corresponden normalmente con dispositivos embebidos con limitaciones de almacenamiento y procesamiento, así como restricciones de memoria y potencia. Además, el número de objetos o dispositivos conectados a Internet contiene grandes previsiones de crecimiento para los próximos años, con unas expectativas de 500 miles de millones de objetos conectados para 2030 [4]. Por lo tanto, para dar cabida a despliegues globales del IoT, además de suplir las limitaciones que existen,

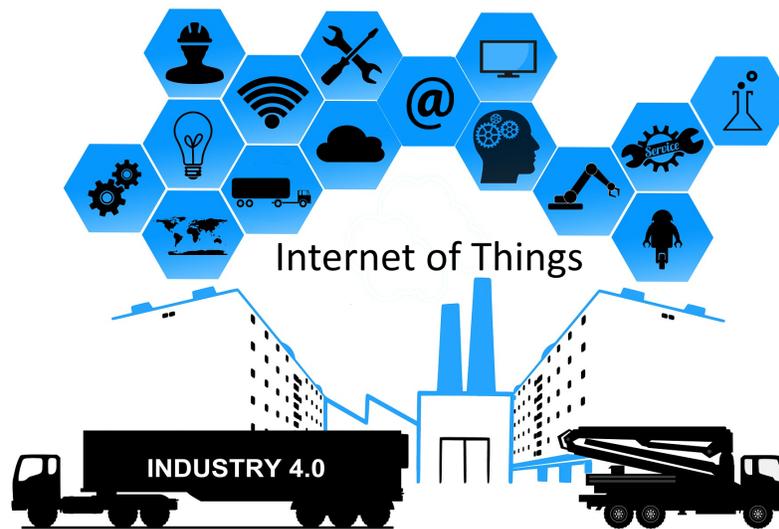


Figura A.1: El mundo conectado a través de Internet de las Cosas

es necesario involucrar nuevos sistemas y paradigmas que faciliten la adopción de este campo. Diferentes integraciones han sido investigadas durante esta tesis doctoral para suplir las limitaciones del IoT. Entre ellas, cabe destacar a cloud computing (computación en la nube), también conocido como el cloud de las cosas [8], que ha intentado solventar las limitaciones del IoT. Cloud computing permite un acceso bajo demanda a un conjunto configurable de recursos, proveyendo capacidades ilimitadas en términos de almacenamiento y procesamiento [9], las principales limitaciones en el IoT. Aunque la integración con cloud computing provee al IoT de las capacidades mencionadas, las necesidades actuales de extraer conocimiento de información en tiempo real implican pensar más allá. Recientes paradigmas como fog y edge computing intentan abordar este último problema, moviendo el procesamiento lo más cerca posible a dónde es generado, reduciendo por tanto la latencia y el ancho de banda en las comunicaciones entre el IoT y el cloud. Otro de los protocolos que ha tenido una gran repercusión en el IoT y que ha sido ampliamente utilizado en esta tesis, es el protocolo CoAP, que tiene como objetivo proveer servicios web RESTful en dispositivos con recursos limitados.

A.1.1 Constrained Application Protocol (CoAP)

Actualmente, el número de servicios web que intercambian información en Internet está en auge. Tales servicios proveen una gran cantidad de datos y servicios accesibles globalmente y de forma abierta a través de Internet. El movimiento del IoT a Internet proporcionaría un conjunto más amplio de servicios en Internet, incorporando servicios ciber-físicos y logrando la inclusión o integración del mundo físico en los servicios tradicionales de información. Este paradigma es también conocido como la web de las cosas. Además, los servicios web son accesibles en múltiples plataformas con conexión a Internet, como pueden ser ordenadores personales, tablets, smartphones o PDAs. Por lo que a partir de Internet, además de proveer información

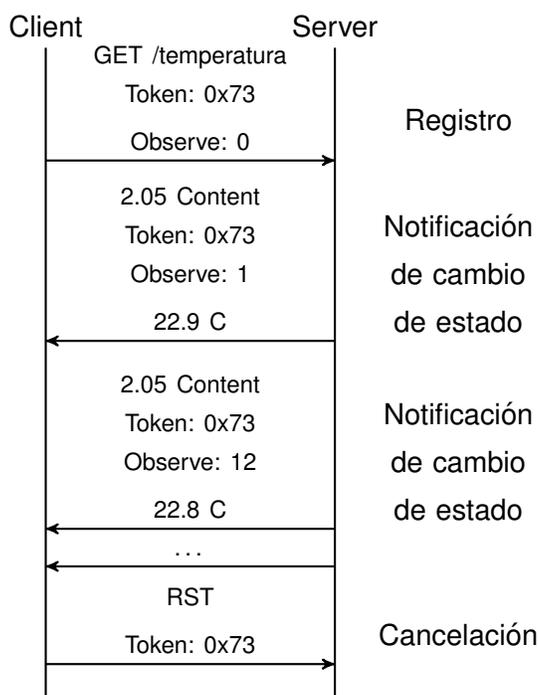


Figura A.2: Observando un recurso en CoAP

accesible de forma global, el IoT proveerá un acceso global a un gran número de dispositivos y sistemas de información.

A pesar de ello, los servicios web actuales suelen trabajar sobre protocolos como HTTP o TCP, que pueden ser demasiado pesados para dispositivos con limitaciones de recursos, como los involucrados en el IoT. CoAP [7] es un protocolo web diseñado por el Grupo de Trabajo de Ingeniería de Internet (IETF, por sus siglas en inglés) para dispositivos con recursos limitados. CoAP sigue el mismo estilo que los servicios web RESTful tradicionales, soportando la identificación de recursos mediante URIs y un modelo como HTTP. Pero CoAP hace uso del protocolo UDP, lo que conlleva a una reducción de la sobrecarga de transferencia y la cabecera de TCP. Unido a que CoAP contiene únicamente 4 bits de cabecera, lo convierten en un protocolo prominente para llevar a cabo la web de las cosas y poder instalar servidores de recursos en dispositivos embebidos.

Además, CoAP soporta una comunicación asíncrona y multicast, no disponible en TCP y deseable en dispositivos con limitaciones de recursos. A partir de la comunicación asíncrona, conocida como observe, se establece una suscripción entre los servidores CoAP y los clientes interesados en la información. Los datos son enviados cuando están disponibles en los servidores CoAP, evitándose la sobrecarga de peticiones por parte de los clientes. La Figura A.2 muestra un ejemplo de una comunicación entre un cliente y un servidor con una suscripción. El cliente interesado en observar un recurso inicia una petición para obtener sus datos asíncronamente. A partir de ese momento, el cliente recibirá los cambios de estado del recurso deseado. Por último, la observación termina cuando el cliente la cancela.

A.1.2 Motivación

A pesar de las mejoras introducidas por los anteriores protocolos y paradigmas en el IoT, la adopción del IoT todavía presenta un gran número de desafíos para llegar a ser una realidad. Entre los desafíos se encuentran la volatilidad y movilidad de sus aplicaciones, las limitaciones de sus dispositivos, y las necesidades de un procesamiento con baja latencia y datos confiables. Durante el desarrollo de esta tesis doctoral hemos identificado cuatro diferentes motivaciones que han inspirado nuestra investigación para facilitar la adopción del IoT.

Por una parte, las aplicaciones del IoT todavía siguen diseñándose acopladas a los dispositivos finales, lo que conlleva a que tales aplicaciones no se adapten a los continuos cambios que puedan producirse en la infraestructura subyacente. En el caso de la inclusión de nuevos dispositivos a estas aplicaciones, o incluso de su desconexión debido al consumo de su batería, aún es necesario reconfigurar las aplicaciones y dispositivos para adaptarse a los nuevos cambios del entorno. En el IoT, donde la cantidad de dispositivos crece sin precedentes y los dispositivos presentan una gran volatilidad, estas limitaciones dificultan con creces su expansión. Por ejemplo, en el caso de la monitorización medioambiental, las aplicaciones desacopladas de los dispositivos podrían mostrar mediciones sobre los sensores desplegados dinámicamente en función de las necesidades de los usuarios. Además, las aplicaciones del IoT son personalizadas, es decir, la infraestructura del IoT subyacente se puede diseñar para que funcione con ciertos perfiles de usuarios. Esto aumenta enormemente la complejidad en el desarrollo de aplicaciones. Por ejemplo, el desarrollo de toda la lógica personalizada para una aplicación personalizada para más de cien usuarios finales puede suponer un mayor esfuerzo de desarrollo que toda la arquitectura en sí misma. Por último, pero no menos importante, los usuarios finales pueden formar parte de diferentes aplicaciones del IoT en diferentes escenarios. En este caso, sería deseable tener un mejor control sobre estas aplicaciones con la mínima configuración posible y el uso de una interfaz común.

Aparte de la portabilidad software, las soluciones del IoT comprenden una multitud de componentes hardware que pueden ser intercambiados o requeridos en diferentes entornos dependiendo de las necesidades de los usuarios. En la mayoría de las situaciones, los dispositivos del IoT se pueden dividir en dos tipos de categorías. Por un lado, los dispositivos pueden constituir una caja negra con sensores específicos, lo que dificulta su configuración. Un ejemplo de ello son las pulseras inteligentes. Por otro lado, los dispositivos del IoT pueden ser compuestos a través de microcontroladores configurables, permitiendo la creación de entornos personalizados. Estos dispositivos están más cerca de la visión del IoT de tener dispositivos autónomos y configurables, también conocidos como dispositivos plug&play. Sin embargo, las herramientas y el conocimiento necesarios para programar y configurar tales sistemas no están actualmente accesibles a toda la población. Además, este proceso debería de poder ser realizado en tiempo de ejecución para evitar la interrupción del servicio de los dispositivos. Esto podría conllevar también a un ahorro energético (crítico en dispositivos del IoT), desactivando los componentes hardware no necesarios cuando sea requerido. Múltiples escenarios de aplicación hacen uso de sensores y actuadores para su operación, tales como *smart cities* y aplicaciones ambientales, por lo que este desafío podría tener un efecto positivo en múltiples escenarios del IoT.

Por otra parte, las limitaciones de los dispositivos del IoT dificultan la aplicación de paradigmas, como big data, que convierten datos en bruto en información útil, requiriendo grandes capacidades en términos de procesamiento y almacenamiento. Paradigmas como cloud computing han sido integrados con el IoT para superar sus limitaciones y proveer las capacidades necesarias. Aunque estas integraciones pueden superar tales limitaciones, las necesidades actuales de extraer conocimiento de datos en tiempo real implican pensar más allá. Aplicaciones en entornos críticos, como la monitorización de la salud estructural de infraestructuras, no sólo requieren de grandes capacidades de cómputo a través de aprendizaje profundo sino también un bajo tiempo de respuesta, en otro caso, la información podría dejar de ser útil. En tales escenarios, donde la latencia de propagación de la información sobre la red podría no cumplir con los requerimientos de las aplicaciones, conducen a que cloud computing pueda no ser adecuado debido a una alta latencia. Paradigmas actuales como fog y edge computing deberían de ser tenidos en cuenta para llevar a cabo la integración del IoT con cloud computing y cumplir con los requisitos de tiempo de las citadas aplicaciones, sin olvidar las limitaciones de los dispositivos del IoT.

Por último, esta gran expansión del IoT debe de ser soportada por mecanismos y protocolos estándares como los vistos anteriormente para reducir la existente heterogeneidad en este campo. Esta heterogeneidad conduce hasta soluciones verticales y acopladas, lo que dificulta la adopción del IoT. Sin embargo, aparte de la heterogeneidad presente en el IoT, la confiabilidad de su información es también un asunto importante a tener en cuenta. Hoy en día confiamos en la información de las entidades financieras y gubernamentales entre otras, pero, ¿podemos estar seguros de que la información proveída por otra entidad externa no ha sido alterada, falsificada o manipulada de cualquier manera? Esta es una pregunta difícil de responder en sistemas centralizados donde toda la información reside en un único lugar. Entidades no confiables pueden alterar la información acorde a sus propios intereses, por lo que la información que proveen puede no ser completamente confiable. Esto trae la necesidad de verificar que la información no ha sido modificada durante su ciclo de vida. Aunque el IoT puede facilitar la digitalización de la información en sí, la confiabilidad en la información sigue todavía siendo un desafío. En ese sentido, una nueva tecnología que nació con la primera criptomoneda descentralizada, tiene el potencial de ofrecer una solución al problema de confiabilidad de la información: Blockchain. La integración con el IoT permitiría la mejora de ambas tecnologías. Dominios de aplicación que gestionan datos sensibles, como atención médica y vigilancia policial podrían hacer uso de esta integración para adquirir más información de sus procesos y almacenarla de forma confiable.

Los desafíos anteriores han motivado el trabajo de esta tesis doctoral. Específicamente, esta tesis aborda las siguientes preguntas de investigación:

- ¿Cómo puede facilitarse la gestión y despliegue de recursos físicos en el IoT?
- ¿Cómo podemos definir intuitivamente aplicaciones del IoT portables, adaptables, personalizadas y desacopladas de los dispositivos?
- ¿Cómo puede ser el IoT integrado con cloud computing teniendo en cuenta la gran cantidad de datos generados, una baja latencia y las limitaciones de sus dispositivos?
- ¿Cómo puede ser el IoT integrado con Blockchain?, y ¿cuáles son los potenciales beneficios y desafíos

de tal integración?

A.1.3 Contribuciones

Esta tesis doctoral pretende investigar los anteriormente mencionados desafíos de investigación. El principal objetivo de esta tesis doctoral (IntegraDos) es facilitar la adopción del IoT a través de los desafíos de investigación e integraciones identificados. La Figura A.3 muestra una visión general de las contribuciones de esta tesis. En ella se pueden diferenciar cuatro componentes principales: el IoT, cloud, blockchain y edge. Todos ellos han sido tenidos en cuenta para superar las limitaciones del IoT. Por un lado, ha sido abordado cómo puede ser facilitado la gestión de sensores y actuadores en dispositivos físicos sin programar y resetear sus sistemas (1). Por otro lado, un sistema para programar aplicaciones del IoT portables, adaptables, personalizadas y desacopladas de los dispositivos ha sido definido (2). También, han sido analizados los componentes para una integración del IoT y cloud computing, concluyendo en la arquitectura λ -CoAP (3). Por último, los desafíos para una integración del IoT y Blockchain han sido analizados junto con una evaluación de las posibilidades de los dispositivos del IoT para incorporar nodos de Blockchain (4).

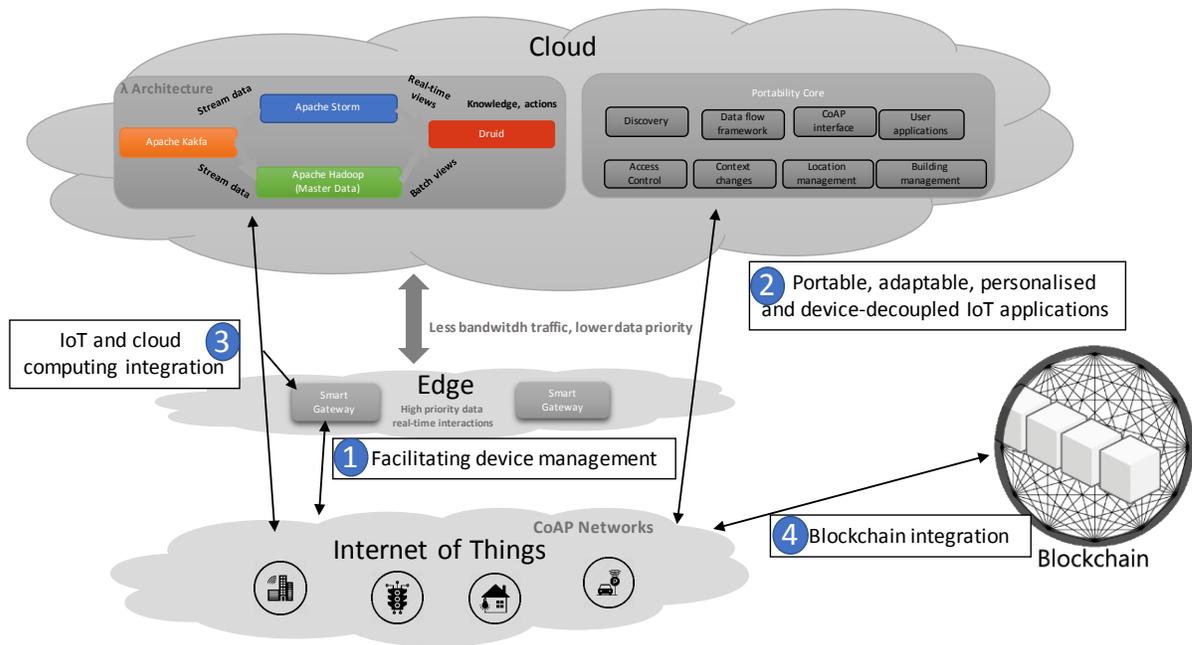


Figura A.3: Visión general de las contribuciones de la tesis

En particular, las siguientes contribuciones han sido proporcionadas en esta tesis doctoral para responder a las preguntas de investigación formuladas:

- Un sistema para el despliegue y gestión de sensores y actuadores que reduce la brecha originada entre los usuarios finales y los entornos personalizados. A través de este sistema, los usuarios finales

pueden incorporar nuevos sensores y actuadores en sus microcontroladores sin tener que acceder y programar las placas de desarrollo. Dominios de aplicación con una alta densidad de componentes hardware serán beneficiados de este sistema en la gestión de sus componentes. Las limitaciones de los dispositivos para soportar librerías han sido tenidas en cuenta con la definición de un middleware objetivo para determinados entornos.

- Un marco de desarrollo que permite el desarrollo de aplicaciones del IoT portables y desacopladas de los dispositivos finales, que a su vez pueden ser adaptables en contextos cambiantes, conocido como Appdaptivity. A través de Appdaptivity, los desarrolladores de aplicaciones pueden crear intuitivamente aplicaciones portables y personalizadas, con una completa abstracción de la infraestructura subyacente. Esto permite el desarrollo de aplicaciones que se adaptan a los continuos cambios en la infraestructura subyacente. Los desarrolladores de aplicaciones sólo necesitan concentrarse en la lógica de la aplicación, mientras que los detalles de bajo nivel de la infraestructura subyacente son gestionados por Appdaptivity. Esta contribución cumple con los requerimientos del desarrollo de aplicaciones del IoT en general, y específicamente de aquellas que tienen comportamientos dinámicos y componentes plug&play.
- Un amplio análisis de los componentes para una integración del IoT y cloud computing, incluyendo integraciones y propuestas existentes. El estudio comprende desde componentes de bajo nivel para el IoT, como IoT middlewares, hasta plataformas multidisciplinares de cloud computing. El análisis condujo hasta la definición de la arquitectura λ -CoAP, que presenta una integración de cloud computing con el IoT. La arquitectura λ -CoAP tiene como objetivo procesar, analizar y consultar grandes cantidades de información del IoT con una baja latencia. Esta baja latencia es lograda en parte gracias a una capa edge que ha sido incorporada tanto para reducir la latencia como el ancho de banda en las comunicaciones entre el IoT y el cloud. Dominios de aplicación que requieren de un procesamiento de datos a gran escala con una baja latencia, como la monitorización de la salud estructural, podrían hacer uso de esta arquitectura para superar sus desafíos.
- Un exhaustivo estudio de los desafíos y los problemas existentes en la nueva disruptiva tecnología para la confiabilidad de información, Blockchain, y su integración con el IoT. El estudio también analiza plataformas actuales de Blockchain y aplicaciones, y provee una evaluación y comparación del rendimiento de diferentes nodos de Blockchain en dispositivos del IoT. Por último, esta contribución pretende analizar los desafíos y los problemas abiertos para una integración del IoT con Blockchain en un futuro próximo con el fin de asegurar la confiabilidad de la información en aplicaciones del IoT.

Por tanto, el objetivo general de esta tesis doctoral, facilitar la adopción del IoT, ha sido abordado a través de las contribuciones anteriores. Tales contribuciones no están estrictamente relacionadas entre sí, sino que contribuyen al objetivo general de este tesis como vamos a ver a continuación. En primer lugar, las contribuciones del sistema de gestión y despliegue de recursos físicos en tiempo de ejecución y la arquitectura λ -CoAP han sido diferentes iteraciones del mismo proyecto y están perfectamente integradas. Esto puede abrir más oportunidades, como aplicaciones que requieran de un gran procesamiento de datos y la gestión de

componentes hardware. Por ejemplo, un caso de uso de una aplicación podría procesar los datos recibidos y decidir qué componentes físicos no son necesarios para su desactivación, ahorrando energía en los dispositivos del IoT. Por otra parte, Appdaptivity fue desarrollado durante una estancia de investigación en el grupo de investigación IDLab de la Universidad de Gante. Este proyecto contribuye al objetivo general definido en esta tesis doctoral y es parte del grupo de investigación IDLab. Por tanto, este proyecto no ha sido inicialmente integrado con el resto de contribuciones, sin embargo, esto podría ser logrado gracias a su arquitectura basada en componentes. Esta integración debería de tener en cuenta la distribución de la lógica de las aplicaciones en la arquitectura λ -CoAP, permitiendo el desarrollo de aplicaciones con baja latencia, portables, desacopladas de los dispositivos y que requieran de grandes capacidades de cómputo, lo que representaría un gran desafío de investigación. Por último, la contribución de la integración del IoT y Blockchain es un estudio exhaustivo de los problemas abiertos y los desafíos en un tema innovador y actual con el fin de ser abordado e integrado con el resto de contribuciones en un futuro próximo.

A.2 Despliegue y gestión de recursos físicos CoAP en tiempo de ejecución

Los recursos en CoAP representan una operación en un servicio web embebido. Las operaciones en los recursos pueden representar tanto a sensores (por ejemplo, un sensor de temperatura) como actuadores (por ejemplo, un actuador de luz). Normalmente, los recursos se definen en tiempo de compilación, lo que dificulta la gestión de los recursos cuando el sistema está ejecutándose. Aunque CoAP define las directrices para crear recursos en tiempo de ejecución a través de peticiones de tipo POST, no establece las reglas para administrar y desplegar recursos con componentes físicos como sensores y actuadores. El objetivo de este sistema es aprovechar las capacidades de CoAP para proveer un mecanismo de gestión y despliegue de recursos con componentes físicos en tiempo de ejecución. Esto último no requiere la programación o la instalación de componentes externos, por lo que los usuarios finales podrán incorporar nuevos sensores o actuadores en sus microcontroladores sin tener que acceder y programar la placa de desarrollo.

El sistema de gestión y despliegue de recursos físicos está compuesto por tres componentes principales. Por un lado, una interfaz web provee una forma accesible y amigable de gestionar recursos pertenecientes a los CoM (CoAP Middleware) asociados. A través de la interfaz web los usuarios finales pueden tanto crear un nuevo recurso en un CoM, como editar y eliminar los recursos previamente creados. Por otro lado, una pasarela inteligente provee un proxy para traducir de HTTP a CoAP, que no es únicamente utilizado por la interfaz web sino que también puede ser usado por clientes HTTP externos a través de su interfaz REST. La pasarela inteligente también comprende una base de datos/caché que almacena información sobre los recursos de los CoM conectados y sus datos con el fin de proteger los CoM de altas tasas de acceso. Por último, los CoMs ofrecen un servidor CoAP y un conjunto de recursos que puede ser gestionados por las pasarelas inteligentes y la interfaz web.

Conectar sensores o actuadores en microcontroladores requiere de la conexión de ellos en determinados conectores o pines. Este es el principal enfoque por el cual este sistema ha sido desarrollado. Los usuarios

pueden conectar un sensor o un actuador en una placa de desarrollo. Una vez que este ha sido conectado correctamente, los usuarios pueden registrar el sensor/actuador instalado junto con sus conectores con el fin de establecer el recurso disponible para poder interactuar con él en el CoM correspondiente. Para ello, se ha modificado el protocolo CoAP de tal forma que permita la gestión de este nuevo tipo de recursos. Por lo tanto, los usuarios pueden crear un nuevo recurso conectándolo en las placas de desarrollo y registrándolo en la interfaz web.

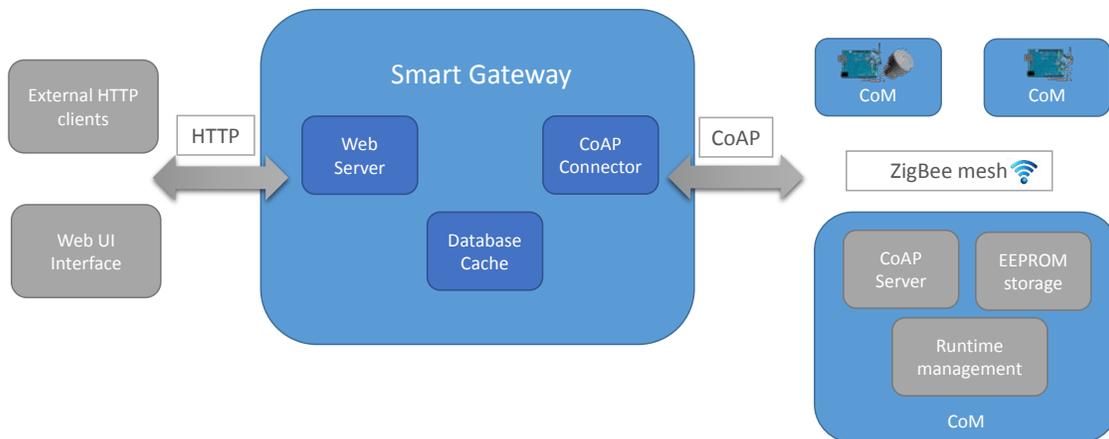


Figura A.4: Visión general del sistema de gestión y despliegue de recursos físicos en tiempo de ejecución

La Figura A.4 muestra una visión general de la arquitectura con los componentes mencionados anteriormente y el flujo de datos entre ellos. El flujo de datos comienza con una petición HTTP para realizar una operación o interactuar con un recurso desde la interfaz web o un cliente HTTP externo. A continuación, la pasarela inteligente traduce la petición HTTP a CoAP y la transmite al middleware CoM. Finalmente, la pasarela inteligente responde a través de HTTP cuando obtiene del CoM el resultado de la operación. A partir de este flujo se permite la gestión y despliegue de sensores y actuadores en los diferentes CoM en tiempo de ejecución.

Aunque algunos sensores y actuadores no requieren de librerías externas para interactuar con ellos, como algunos sensores analógicos, muchos otros requieren de librerías externas para su funcionamiento. Las librerías necesarias para el funcionamiento de tales sensores y actuadores pueden requerir de una importante cantidad de memoria. Por tanto, los dispositivos del IoT con ciertas limitaciones están lejos de ser soportados. Para aliviar esta problemática, cada servidor CoAP define un middleware objetivo en el cual se dará soporte para un determinado número de librerías. También, hemos modificado el protocolo CoAP para soportar la comunicación en dispositivos embebidos, en este caso a través de ZigBee. ZigBee es un protocolo de comunicación basado en el estándar IEEE 802.15.4. ZigBee ha sido elegido porque proporciona un modo de

funcionamiento de baja potencia y una comunicación segura y global para el IoT.

A.3 Appdaptivity: un sistema para el desarrollo de aplicaciones portables y desacoplados de los dispositivos del IoT

En aplicaciones desacopladas de los dispositivos, el desarrollo se centra en la lógica de la aplicación, en vez de diseñar aplicaciones para dispositivos específicos. Esto permite un gran desacoplamiento de las aplicaciones de los dispositivos, y una perfecta adaptación de las aplicaciones a los cambios en el entorno. Esta segunda contribución, conocida como Appdaptivity, permite el desarrollo de aplicaciones independientemente de la infraestructura física subyacente. A través de un proceso de descubrimiento, la infraestructura se adhiere al sistema cuando está disponible. El descubrimiento puede realizarse en múltiples escenarios gracias a la involucración de los usuarios finales. En cualquier caso, el descubrimiento se realiza dinámicamente en segundo plano y la infraestructura subyacente puede ser descubierta en múltiples contextos. Esto conlleva a que la lógica de la aplicación sólo tenga que definirse una vez y este proceso de descubrimiento permita la portabilidad de aplicaciones a múltiples y heterogéneos entornos.

Una abstracción ha sido adoptada para categorizar la infraestructura subyacente y los cambios de contexto para los usuarios finales: la localización. La localización hace referencia a una localización lógica donde una infraestructura física ha sido desplegada, por ejemplo, una habitación. La localización permite modelar el mundo físico tal y como ha sido concebido. Todos los cambios de contexto que hayan ocurrido en las localizaciones se verán reflejados en la lógica de aplicación definida. Esto requiere que la infraestructura subyacente tenga que soportar la localización para correlacionar sus dispositivos con las localizaciones lógicas definidas. Habilitar la localización en dispositivos embebidos puede incurrir en una pérdida de rendimiento. Por tanto, trabajos anteriores han sido tenidos en cuenta para no afectar en el rendimiento de los dispositivos [129]. Lo mismo ocurre con el desarrollo de aplicaciones personalizadas, donde se ha tenido en cuenta trabajos anteriores para gestionar el acceso en dispositivos embebidos [130], de tal forma que sólo las personas autorizadas tenga acceso a los dispositivos. En el proceso de descubrimiento, la infraestructura subyacente es descubierta para el usuario que tenga permisos en ella, y será actualizada con cualquier cambio. Si una determinada lógica depende de una situación (por ejemplo, un valor de temperatura), la lógica de la aplicación no será activada hasta que todos los correspondientes componentes han sido recibidos. Por lo tanto, el desarrollo de aplicaciones puede realizarse de forma global para todas las aplicaciones personalizadas, ya que la lógica de aplicación será activada dependiendo de los permisos de cada usuario.

Appdaptivity ha adoptado un modelo de programación basado en flujo de datos [39] para definir aplicaciones de usuario intuitivamente. En el desarrollo basado en flujo de datos los usuarios definen las aplicaciones a través de un grafo conectado de nodos, modelando un flujo de acciones que los datos deberían de tomar. Los nodos tienen una función que no depende de otros nodos, por lo que forman componentes altamente reusables y portables para el desarrollo de la lógica de aplicaciones. Además, Appdaptivity provee un gran conjunto de nodos para definir aplicaciones, que pueden ser definidas sin una línea de código fuente. Este mecanismo permite que los usuarios no técnicos encuentren más fácil utilizar el sistema, lo que a su vez facilita

la expansión de la solución propuesta. Otros marcos de desarrollo proveen algunas de las capacidades para desarrollar aplicaciones personalizadas, desacopladas de los dispositivos, y que se adapten a contextos cambiantes. Hasta donde tenemos conocimiento, este enfoque es el primero en proporcionar un marco de desarrollo para la creación intuitiva de aplicaciones del IoT que pueden ser portables, personalizadas y adaptadas a contextos cambiantes.

Appdaptivity comprende el Portability Core (PoCo), que es el componente encargado de habilitar la programación basada en flujo de datos a través de una interfaz web accesible y transparente. El PoCo permite el desarrollo de aplicaciones con localizaciones para los usuarios finales. Las aplicaciones resultantes pueden ser dinámicamente portables a otros entornos. Una adaptación para diversos escenarios de despliegue ha sido tomada en cuenta, y el PoCo permite una variedad de despliegues para diferentes casos de uso. La lógica de la aplicación comprende diferentes comportamientos en el desarrollo basado en flujo de datos, mientras que las aplicaciones de usuario son un conjunto de clientes que reciben y envían datos al PoCo con la información originada en el sistema y la información requerida para activar los flujos de datos. Los comportamientos comprenden la lógica de las aplicaciones resultantes y serán siempre actualizados con los cambios en la infraestructura. Las aplicaciones de usuario son normalmente aplicaciones para smartphones que muestran una representación actualizada de los comportamientos con la infraestructura subyacente proveída por el PoCo para los usuarios finales.

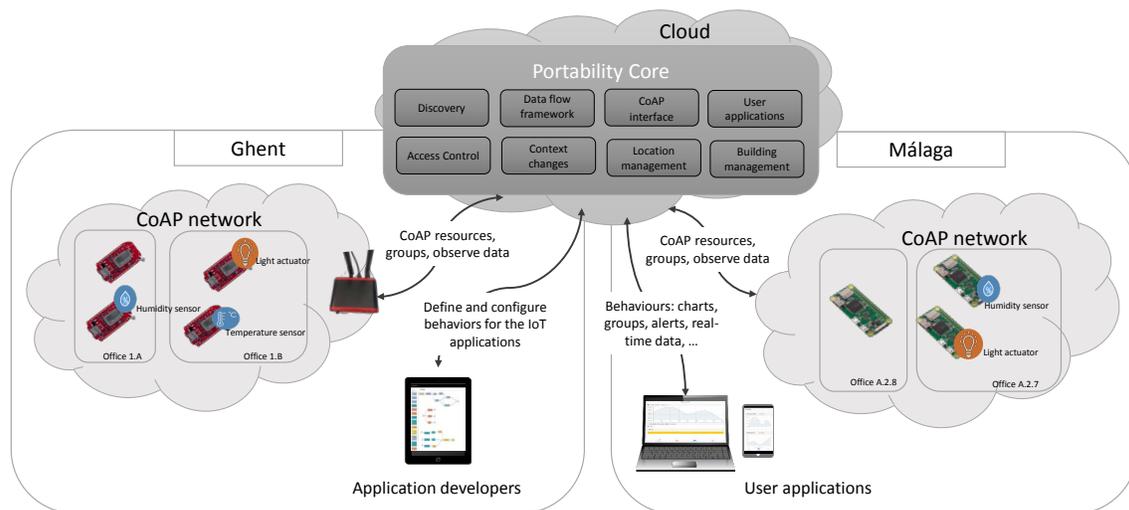


Figura A.5: Arquitectura de Appdaptivity

Los dispositivos del IoT son otra parte importante en el sistema, y la más limitada, ya que se corresponden con dispositivos con recursos limitados. Por esta razón, Appdaptivity ha adoptado CoAP como el protocolo de aplicación para interactuar con los dispositivos finales. En el ámbito de esta contribución, habrá sensores que podrán ser monitorizados y actuadores con los que se podrá interactuar, por lo tanto, un estándar ligero como CoAP con recursos accesibles representa un protocolo adecuado. Todas las redes de sensores inalámbricas que soportan CoAP son conocidas como redes de CoAP en esta contribución. Las redes de CoAP proveen

los sensores, actuadores y la información de los dispositivos que son accesibles a través de CoAP para crear aplicaciones del IoT. El PoCo es el encargado de interactuar con las redes de CoAP y permite la definición de flujo de datos con los recursos de CoAP obtenidos, que compondrán los diferentes comportamientos en las aplicaciones de usuario. Una visión general del sistema es presentada en la Figura A.5. En ella se muestra el conjunto de componentes que permite las funcionalidades de Appdaptivity. En este caso, el PoCo ha sido desplegado en el cloud, pero también puede ser desplegado de forma local y embebida. En aplicaciones del IoT globales, las redes de CoAP son desplegadas en diferentes localizaciones físicas, como por ejemplo, Málaga y Gante. Appdaptivity gestiona el proceso de portabilidad de las aplicaciones en tales entornos, al mismo tiempo que permite la definición de comportamientos personalizados con una interfaz intuitiva.

A.4 λ -CoAP: una Integración de Internet de las Cosas y Cloud Computing

El IoT genera grandes cantidades de datos, lo cual se espera que continúe en crecimiento para los próximos años [143]. Las limitaciones de los dispositivos del IoT hacen que abordar paradigmas actuales como big data o aprendizaje profundo, sea una tarea muy complicada, ya que requieren de grandes capacidades de procesamiento y almacenamiento. Tales paradigmas intentan extraer conocimiento y generar acciones sobre información del IoT. En los últimos años, la integración del IoT con tecnologías disruptivas como cloud computing [2] han proveído las capacidades necesarias en el IoT para abordar los anteriores paradigmas. Sin embargo, esto ha resultado en un incremento de latencia en las comunicaciones entre el IoT y el cloud, y una dependencia en factores externos como disponibilidad, enrutamiento y redes, lo que implica el uso de servicios externos. En determinadas ocasiones es necesario actuar con una latencia muy reducida, como el caso de la detección anticipada de un terremoto o una explosión. Un nuevo paradigma conocido como edge computing [10] tiene como objetivo reducir el tiempo de respuesta y el ancho de banda en comunicaciones entre el IoT y el cloud, moviendo la computación al borde (edge) de la red. Las nuevas generaciones de coches autónomos pretenden generar hasta 1GB de información por segundo, por lo que edge computing jugará un papel clave para reducir la latencia y el ancho de banda. En otro caso, supondría un cuello de botella para las redes de comunicaciones.

La integración del IoT con cloud computing empezó con un análisis de los componentes requeridos para tal integración. Los componentes requeridos fueron clasificados en dos categorías teniendo en cuenta las necesidades tanto en el IoT como en el cloud. Por un lado, el análisis abordó plataformas cloud multidisciplinares para satisfacer las limitaciones del IoT, ofreciendo nuevas oportunidades de negocio y una mayor escalabilidad. Por otro lado, el análisis abordó la comparación de diferentes middleware del IoT para abstraer las limitaciones de los dispositivos del IoT. A través de los middleware IoT, los dispositivos tendrán un mecanismo ligero e interoperable de comunicación entre ellos mismos y con los sistemas cloud.

El análisis facilitó la definición de la arquitectura λ -CoAP, que provee una integración de cloud computing con el IoT con una capa de edge computing. Por un lado, la heterogeneidad dentro de los dispositivos

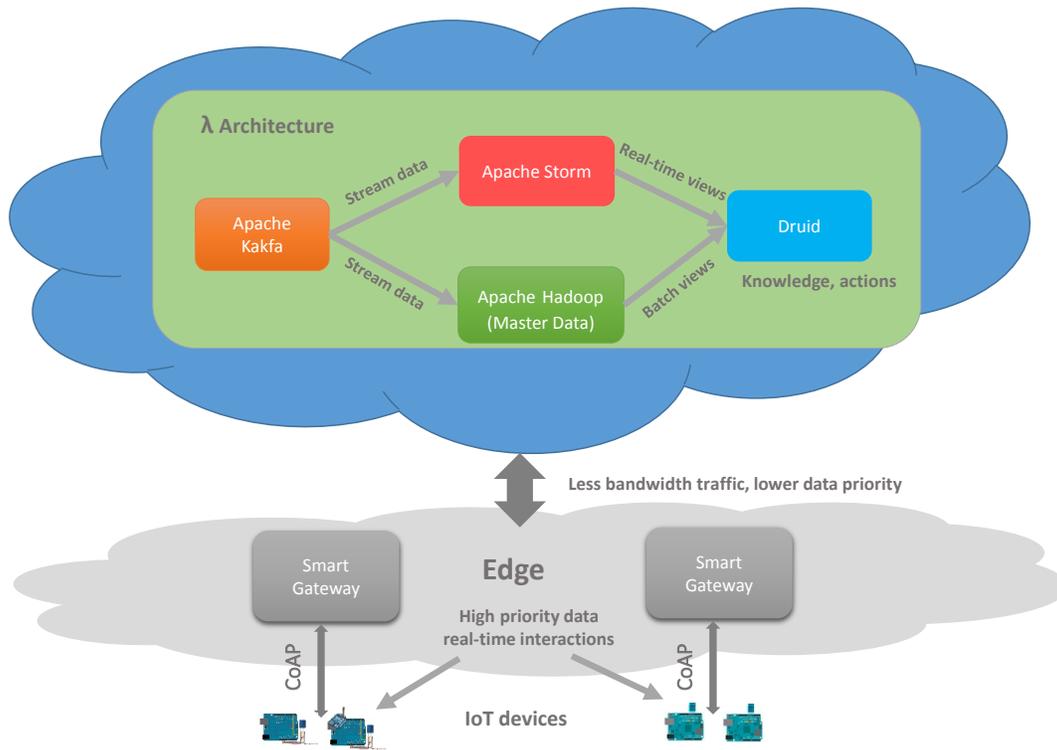


Figura A.6: Una visión general de la arquitectura λ-CoAP

del IoT involucrados es abstraída al mismo tiempo que se permite el acceso a servicios web ligeros a través de CoAP. Por otra parte, la arquitectura lambda permite el procesamiento, actuación y análisis en tiempo real de las grandes cantidades de datos generados por el IoT. Una capa edge que reside entre el IoT y la arquitectura lambda tiene como objetivo reducir la latencia y el ancho de banda entre las comunicaciones del IoT y el cloud. La arquitectura λ-CoAP se complementa con una pasarela inteligente que permite interconectar la arquitectura lambda, HTTP y CoAP; un middleware para instalar CoAP en dispositivos embebidos; y componentes externos, como una interfaz web de usuario y un componente para gestionar y actuar sobre el IoT subyacente, respectivamente.

La Figura A.6 muestra una visión general de la arquitectura λ-CoAP. En la parte inferior, los dispositivos finales incorporan un middleware basado en CoAP cuyas funciones principales son abstraer la comunicación con las capas superiores para consultar información y actuar sobre los dispositivos, así como abstraer aspectos de descubrimiento y bajo nivel. Este middleware también ha sido utilizado en la contribución de gestión y despliegue de recursos físicos, Sección A.2. En la parte central residen las pasarelas inteligentes y la capa edge. Al principio (Sección A.2), las pasarelas inteligentes actuaban como un puente de interconexión entre la arquitectura lambda, los dispositivos subyacentes y los usuarios finales. Sin embargo, las pasarelas inteligentes han sido adaptadas para soportar una arquitectura de edge computing, además de ser dispositivos autónomos con capacidades propias de procesamiento y con una tecnología de virtualización ligera. Por lo

que en ellas se permite definir la lógica de la aplicación de la capa edge a través de una interfaz web intuitiva, de forma similar que en la Sección A.3. De hecho, los buenos resultados y la experiencia adquirida en Appdaptivity ayudaron a la adopción de esta interfaz en la capa edge. Con su procesamiento se puede reducir la latencia de comunicación y aligerar el ancho de banda de las redes. Por lo tanto, la capa edge comprende las pasarelas inteligentes desplegadas en la arquitectura. Por último, en la parte posterior, el cloud es responsable de distribuir los componentes de la arquitectura lambda.

A.5 La Integración de Blockchain e Internet de las Cosas

En muchas áreas donde una exhaustiva trazabilidad de activos durante su ciclo de vida es requerida por regulaciones, la inmutabilidad de la información se convierte en un desafío. Concretamente, regulaciones de la Unión Europea requieren que los productores alimentarios realicen una trazabilidad de todas las materias primas utilizadas en la elaboración de sus productos alimentarios, además del destino final de cada uno de ellos. En el caso de una gran compañía alimentaria con miles de proveedores y millones de clientes, la información necesita de estar digitalizada y su procesamiento automatizado con el fin de cumplir con tales regulaciones. Un ejemplo de una gran regulación en trazabilidad es la industria porcina. En este sector, además de una trazabilidad de las materias primas, los tratamientos y el destino de cada pieza, también debe de ser registrado el traslado de animales entre factorías.

Estos escenarios involucran muchos participantes, muchos de ellos todavía confiando en procesos de información no automatizados. En el caso de una intoxicación alimentaria, que ha sido uno de los mayores problemas para la salud de la población mundial a lo largo de la historia, la información que se pierde o es difícil de encontrar implica demoras en la ubicación del foco del problema. Esto puede provocar la desconfianza de la población hacia los productos contaminados, y por ende, una gran disminución de su demanda. De acuerdo con la Organización Mundial de la Salud, se estima que cada año 600 millones de personas en el mundo sufren una enfermedad a causa de una intoxicación alimentaria, de las cuales 420.000 padecen por la misma causa [187]. Por lo tanto, tener información inaccesible o incluso su pérdida puede afectar gravemente a la seguridad alimentaria y a la salud de los consumidores. En este tipo de escenarios el IoT tiene el potencial de transformar y revolucionar la industria y la sociedad, digitalizado el conocimiento de tal forma que pueda ser consultado y controlado en tiempo real.

Aparte de tener la información disponible, la información también necesita de ser confiable. Mantener la información confiable e inmutable durante su ciclo de vida es crítico en estos entornos. Una forma de proveer integridad en información es a través de un servicio distribuido confiable por todos sus participantes que garantice que toda la información permanece inmutable. Si todos los participantes tienen la información y tienen la forma de verificar que esa información no ha sido alterada desde su primera creación, la integridad puede ser lograda. Tener un sistema que garantice la confiabilidad de la información permitiría entre otras cosas a los gobiernos compartir información de forma segura con sus ciudadanos. Esto es exactamente lo que hace Blockchain. Esta nueva tecnología pretende revolucionar la forma en la que intercámbiamos y accedemos a la información. Sin embargo, debido a su novedad, su integración con el IoT todavía no ha sido bien

abordada.

Esta última contribución tiene como objetivo analizar los desafíos en la integración del IoT y Blockchain, y los potenciales beneficios que la integración puede proveer. Específicamente, se han analizado los desafíos en Blockchain y su integración con el IoT. Este análisis ha mostrado como el almacenamiento, la escalabilidad o los aspectos legales son unos de los problemas a tener en cuenta a la hora de abordar la integración de ambas tecnologías. También, se han comparado diferentes plataformas de Blockchain que podrían componer tal integración. Por último, una evaluación de nodos de Blockchain en dispositivos del IoT muestra las capacidades de esta tecnología en este tipo de dispositivos

A.6 Conclusiones y Trabajos Futuros

A.6.1 Conclusiones

El IoT es una tecnología ubicua que está presente en muchas situaciones de nuestro día a día. Desde casas inteligentes hasta procesos de fabricación, el IoT está mejorando la calidad de vida y optimizando muchos servicios a través de sus capacidades de monitorizar y actuar sobre el mundo físico. Varios informes de investigación pronostican una gran evolución de este paradigma para los próximos años, por lo que su adopción en nuestra sociedad será aún más notable. Sin embargo, el IoT presenta severas limitaciones debido a los recursos limitados de sus dispositivos, lo que su condiciona notablemente su expansión. Esta tesis doctoral, *IntegraDos*, tiene como objetivo proveer diferentes sistemas que permitan facilitar la adopción del IoT a través de la integración con diferentes tecnologías y paradigmas. Específicamente, ha sido investigado cómo facilitar la gestión y despliegue de componentes físicos, el desarrollo de aplicaciones del IoT, una integración del IoT con cloud computing, y se han analizado y evaluado cómo puede ser integrado el IoT con Blockchain.

La primera contribución de esta tesis doctoral se corresponde con un sistema para gestionar y desplegar recursos físicos presentado en la Sección A.2. El objetivo de esta solución es reducir la brecha entre los usuarios finales y los entornos personalizados. De aquí en adelante, los usuarios finales podrán instalar sus propios sensores y actuadores en sus microcontroladores a través de interfaces bien definidas y sin tener que programar ni resetear sus sistemas. CoAP y ZigBee han sido usados como servicio web embebido y el medio de comunicación en los microcontroladores respectivamente, por lo tanto permitiendo un enfoque ligero y de bajo consumo para interactuar con recursos. El sistema provee un proxy para interconectar HTTP con CoAP en una pasarela inteligente, de tal forma que cualquier cliente HTTP puede llevar a cabo las operaciones proveídas por la pasarela. De hecho, esto último contribuye a realizar aún más la idea de la web de las cosas. Además, una interfaz web ha sido incorporada para proveer un acceso multidispositivo a los usuarios finales.

Appdaptivity, un marco de desarrollo que permite el desarrollo de aplicaciones del IoT abstrayendo a los desarrolladores de aplicaciones de la subyacente infraestructura, se corresponde con la segunda aportación, presentada en la Sección A.3. La solución propuesta pretende reducir la brecha entre las aplicaciones del IoT y la subyacente infraestructura, que está normalmente demasiado acoplada, conllevando a soluciones verticales. Las aplicaciones resultantes son creadas intuitivamente, personalizadas, portables y automáticamente adaptadas a los cambios de contexto, sin necesidad de recompilar y programar aplicaciones acopladas

a los dispositivos. El modelo de programación de Appdaptivity está basado en programación de flujo de datos, donde los desarrolladores de aplicaciones pueden crear aplicaciones del IoT conectando componentes a través de una interfaz web. Los desarrolladores de aplicaciones sólo necesitan centrarse en la lógica de la aplicación, mientras que otros aspectos de descubrimiento y la inyección de la información de los dispositivos son gestionados por Appdaptivity. La solución está pensada para escenarios heterogéneos, permitiendo la portabilidad de las aplicaciones del IoT, varias opciones de despliegue y la integración con aplicaciones de usuario externas. La infraestructura subyacente está soportada a través de CoAP. Por último, las aplicaciones finales que serán usadas por los usuarios, no están enfocadas en una única aplicación, sino que Appdaptivity permite la integración de múltiples aplicaciones y plataformas a través de sus interfaces bien definidas y sus protocolos estándares.

La arquitectura λ -CoAP, una arquitectura para integrar el IoT con cloud computing con una capa de edge computing ha sido presentada en la Sección A.4. La arquitectura λ -CoAP contribuye a resolver los problemas de procesamiento, análisis y almacenamiento las grandes cantidades de información provenientes del IoT, a través de la arquitectura lambda. CoAP es el protocolo usado en el lado de los dispositivos, de ahí su nombre λ -CoAP. Algunos componentes y la experiencia obtenida de las contribuciones de las secciones A.2 y A.3 han contribuido fuertemente para conformar esta arquitectura. La arquitectura cubre una visión completa de un despliegue de edge computing, desde el IoT, pasando por el edge, hasta el cloud. La arquitectura λ -CoAP incorpora a middleware CoAP que abstrae los aspectos de bajo nivel de los dispositivos del IoT y provee una forma ligera de interactuar con el IoT. Además, el IoT es abstraído con las pasarelas inteligentes, permitiendo el acceso a través de diferentes protocolos de alto nivel. Una tecnología de virtualización ligera ha sido adoptada en las pasarelas inteligentes para tener un mejor control y aislamiento de los procesos en ejecución además de un marco de desarrollo visual para la programación de la lógica en el edge. Por último, la arquitectura lambda desplegada en el cloud permite el procesamiento y almacenamiento de grandes cantidades de datos procedentes del IoT. Aplicaciones del IoT como monitorización de la salud estructural podrían aprovechar esta arquitectura para reducir su latencia y aliviar el ancho de banda con el cloud.

Blockchain es concebida como una tecnología revolucionaria que mantiene la información confiable. Sin embargo, modificar la tecnología sin garantizar adecuadamente su operación o aplicarla a escenarios donde los costes no compensan, son riesgos en los que se puede caer fácilmente. Por tanto, los beneficios de aplicar Blockchain en el IoT deben ser analizados cuidadosamente y tomados con precaución. Por último, la Sección A.5 ha proveído de un análisis de los principales desafíos que Blockchain y el IoT deben abordar para poder trabajar juntos exitosamente. Se han identificado los puntos clave donde Blockchain puede ayudar a las aplicaciones del IoT. Además, una evaluación de diferentes nodos de Blockchain en dispositivos del IoT y una comparación de varias plataformas analizan la factibilidad de la tecnología. Es esperado que Blockchain revolucione el IoT. Por lo tanto, la integración de estas dos tecnologías debería ser abordada teniendo en cuenta los desafíos identificados en esta última contribución.

A.6.2 Trabajos Futuros

Esta investigación nos ha conducido a definir varios desafíos abiertos que pueden ser abordados. Por un lado, el sistema de gestión y despliegue podría garantizar la compatibilidad con diferentes tecnologías de comunicación inalámbrica para dispositivos con recursos limitados, con el fin de evaluarlas y establecer la más adecuada para cada entorno. Un despliegue y gestión de ellas en tiempo de ejecución de la misma forma que se realiza en el sistema de gestión y despliegue, permitiría el uso de múltiples y configurables tecnologías inalámbricas sobre CoAP. Las tecnologías de comunicación inalámbricas, como el IoT, están en una continua expansión. Ejemplos destacados son la nueva generación de redes de gran alcance LPWAN, como Lora y Sigfox. Sin embargo, cada tecnología tiene sus características únicas en términos de cobertura, conectividad, seguridad y consumo, por lo que cada una es adecuada para determinados entornos. Este enfoque también tendría que tener en cuenta los esfuerzos para habilitar IPv6 en redes LPWAN. La adopción de una capa superior como CoAP sobre estas tecnologías aseguraría también la interoperabilidad de ellas en el IoT.

Por otro lado, Appdaptivity podría habilitar un soporte multi-perfil en comunicaciones seguras con las aplicaciones de usuario, facilitando por tanto la adopción global del sistema. Actualmente, el soporte de un único certificado en comportamientos personalizados limita la expansión del sistema. Además, la definición de la conexión segura de cada servidor CoAP es realizada manualmente en el framework usado y podría ser automatizada. Appdaptivity también podría integrar los componentes RESTlet. Los componentes RESTlet permiten la definición de la lógica de la aplicación directamente en los dispositivos del IoT. Eso optimizaría el posicionamiento de la lógica entre Appdaptivity y la subyacente infraestructura, permitiendo una inteligencia distribuida. Una integración entre la arquitectura λ -CoAP y Appdaptivity también podría considerarse.

También, se podría llevar a cabo una evaluación de la arquitectura λ -CoAP en un caso de uso de una mayor envergadura. Actualmente, la evaluación realizada no cubre un escenario del IoT como podría serlo. Sin embargo, los resultados confirman las capacidades de la arquitectura para un caso de uso de mayor envergadura. En este sentido, hemos solicitado recientemente un proyecto de investigación en colaboración con otro grupo de investigación de la Universidad de Málaga que tiene como objetivo monitorizar los espacios universitarios, optimizando su uso y detectando anomalías.

Por último, el análisis de los desafíos y los problemas abiertos en la integración de Blockchain y el IoT establecieron las pautas de nuestro trabajo futuro para superar este desafío. En un futuro cercano, este tema se abordará con el objetivo de mantener la confiabilidad de los datos y las comunicaciones del IoT.

Bibliography

- [1] “C.h.i.p.” Available online: <https://getchip.com/>. (accessed on 22 May 2017).
- [2] M. Díaz, C. Martín, and B. Rubio, “State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing,” *Journal of Network and Computer Applications*, vol. 67, pp. 99–117, 2016.
- [3] K. Ashton, “That ‘internet of things’ thing,” *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009.
- [4] “Internet of things at a glance.” Available online: <https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf>. (accessed on 28 October 2018).
- [5] “Transmission of ipv6 packets over ieee 802.15.4 networks.” Available online: <https://tools.ietf.org/html/rfc4944>. (accessed on 22 May 2017).
- [6] “Rpl: Ipv6 routing protocol for low-power and lossy networks.” Available online: <https://tools.ietf.org/html/rfc6550>. (accessed on 23 May 2017).
- [7] K. H. Shelby, Zach and C. Bormann, “The constrained application protocol (coap).” Available online: <https://tools.ietf.org/html/rfc7252/>. (accessed on 13 May 2016).
- [8] M. Aazam, I. Khan, A. A. Alsaffar, and E.-N. Huh, “Cloud of things: Integrating internet of things and cloud computing and the issues involved,” in *In Applied Sciences and Technology (IBCAST), 2014 11th International Bhurban Conference on, Anchorage, Alaska, USA*, pp. 414–419, IEEE, June 2014.
- [9] A. Botta, W. de Donato, V. Persico, and A. Pescapé, “On the integration of cloud computing and internet of things,” in *In Proceedings of the 2nd International Conference on Future Internet of Things and Cloud (FiCloud-2014), Barcelona, Spain*, pp. 27–29, Aug 2014.
- [10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
- [11] M. García-Valls, A. Dubey, and V. Botti, “Introducing the new paradigm of social dispersed computing: Applications, technologies and challenges,” *Journal of Systems Architecture*, p. In press, 2018.
- [12] L. Alonso, J. Barbarán, J. Chen, M. Díaz, L. Llopis, and B. Rubio, “Middleware and communication technologies for structural health monitoring of critical infrastructures: A survey,” *Computer Standards & Interfaces*, vol. 56, pp. 83–100, 2018.



- [13] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, “On blockchain and its integration with iot. challenges and opportunities,” *Future Generation Computer Systems*, 2018.
- [14] A. Botta, W. De Donato, V. Persico, and A. Pescapé, “Integration of cloud computing and internet of things: a survey,” *Future Generation Computer Systems*, vol. 56, pp. 684–700, 2016.
- [15] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of things: A survey on enabling technologies, protocols, and applications,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.
- [16] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, “Blockchain and iot integration: A systematic survey,” *Sensors*, vol. 18, no. 8, p. 2575, 2018.
- [17] E. F. Jesus, V. R. Chicarino, C. V. de Albuquerque, and A. A. d. A. Rocha, “A survey of how to use blockchain to secure internet of things and the stalker attack,” *Security and Communication Networks*, vol. 2018, 2018.
- [18] “Observing resources in the constrained application protocol (coap).” Available online: <https://tools.ietf.org/html/rfc7641>. (accessed on 23 May 2017).
- [19] G. K. Teklemariam, J. Hoebeke, I. Moerman, and P. Demeester, “Facilitating the creation of iot applications through conditional observations in coap,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 177, 2013.
- [20] M. Weiser, “Connecting the physical world with pervasive networks,” 2002.
- [21] V. Tsiatsis, A. Gluhak, T. Bauge, F. Montagut, J. Bernat, M. Bauer, C. Villalonga, P. M. Barnaghi, and S. Krco, “The sensei real world internet architecture.,” in *Future Internet Assembly, Ghent, Belgium*, pp. 247–256, Dec 2010.
- [22] “Openiot.” Available online: <https://github.com/OpenIoTOrg/openiot>. (accessed on 17 May 2015).
- [23] H. Aloulou, M. Mokhtari, T. Tiberghien, J. Biswas, and L. J. H. Kenneth, “A semantic plug&play based framework for ambient assisted living,” in *Impact Analysis of solutions for chronic disease Prevention and Management*, pp. 165–172, Springer, 2012.
- [24] A. Bröring, P. Maué, K. Janowicz, D. Nüst, and C. Malewski, “Semantically-enabled sensor plug & play for the sensor web,” *Sensors*, vol. 11, no. 8, pp. 7568–7605, 2011.
- [25] F. Yang, N. Matthys, R. Bachiller, S. Michiels, W. Joosen, and D. Hughes, “ μ pnp: plug and play peripherals for the internet of things,” in *in proceedings of the Tenth European Conference on Computer Systems, Bordeaux, France*, p. 25, ACM, 2015.
- [26] “Global sensor network.” Available online: <https://github.com/LSIR/gsn/wiki>. (accessed on 20 December 2015).

- [27] C. Perera, A. Zaslavsky, P. Christen, A. Salehi, and D. Georgakopoulos, "Connecting mobile things to global sensor network middleware using system-generated wrappers," in *In Proceedings of the Eleventh ACM International Workshop on Data Engineering for Wireless and Mobile Access, Scottsdale, AZ, USA*, pp. 23–30, ACM, 2012.
- [28] A. Sleman and R. Moeller, "Integration of wireless sensor network services into other home and industrial networks; using device profile for web services (dpws)," in *In Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on, Damascus, Syria*, pp. 1–5, IEEE, April 2008.
- [29] A. Dunkels *et al.*, "Efficient application integration in ip-based sensor networks," in *In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, Berkeley, CA, USA*, pp. 43–48, ACM, 2009.
- [30] T. Levä, O. Mazhelis, and H. Suomi, "Comparing the cost-efficiency of coap and http in web of things applications," *Decision Support Systems*, vol. 63, pp. 23–38, 2014.
- [31] T. Riedel, N. Fantana, A. Genaid, D. Yordanov, H. R. Schmidtke, and M. Beigl, "Using web service gateways and code generation for sustainable iot system development," in *Internet of Things (IOT), 2010*, pp. 1–8, IEEE, 2010.
- [32] P. Ruckebusch, E. De 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*, vol. 36, pp. 127–151, 2016.
- [33] A. Taherkordi, F. Loiret, R. Rouvoy, and F. Eliassen, "Optimizing sensor network reprogramming via in situ reconfigurable components," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 2, p. 14, 2013.
- [34] R. Gummadi, O. Gnawali, and R. Govindan, "Macro-programming wireless sensor networks using kairós," in *In International Conference on Distributed Computing in Sensor Systems, Marina del Rey, CA, USA*, pp. 126–140, Springer, 2005.
- [35] R. Newton and M. Welsh, "Region streams: Functional macroprogramming for sensor networks," in *In Proceedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004, Toronto, Canada*, pp. 78–87, ACM, 2004.
- [36] Y.-H. Tu, Y.-C. Li, T.-C. Chien, and P. H. Chou, "Ecocast: interactive, object-oriented macroprogramming for networks of ultra-compact wireless sensor nodes," in *In Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on, Chicago, IL, USA*, pp. 366–377, IEEE, 2011.
- [37] G. K. Teklemariam, F. Van Den Abeele, I. Moerman, P. Demeester, and J. Hoebeke, "Bindings and restlets: A novel set of coap-based application enablers to build iot applications," *Sensors*, vol. 16, no. 8, p. 1217, 2016.
- [38] D. Alessandrelli, M. Petraccay, and P. Pagano, "T-res: Enabling reconfigurable in-network processing in iot-based wsns," in *In Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on, Cambridge, MA, USA*, pp. 337–344, IEEE, 2013.

- [39] N. K. Giang, M. Blackstock, R. Lea, and V. C. Leung, “Developing iot applications in the fog: A distributed dataflow approach,” in *In Internet of Things (IOT), 2015 5th International Conference on the, Seoul, South Korea*, pp. 155–162, IEEE, 2015.
- [40] R. Kleinfeld, S. Steglich, L. Radziwonowicz, and C. Doukas, “glue. things: a mashup platform for wiring the internet of things with the internet of services,” in *In Proceedings of the 5th International Workshop on Web of Things, Cambridge, MA, USA*, pp. 16–21, ACM, 2014.
- [41] “Node-red: A visual tool for wiring the internet of things.” Available online: <https://nodered.org/>. (accessed on 26 January 2017).
- [42] S. Nastic, S. Sehic, M. Vogler, H.-L. Truong, and S. Dustdar, “Patricia—a novel programming model for iot applications on cloud platforms,” in *In Service-Oriented Computing and Applications (SOCA), 2013 IEEE 6th International Conference on, Koloa, HI, USA*, pp. 53–60, IEEE, 2013.
- [43] T. Booth and S. Stumpf, “End-user experiences of visual and textual programming environments for arduino,” in *In International Symposium on End User Development, Copenhagen, Denmark*, pp. 25–39, Springer, 2013.
- [44] G. Celani and C. E. V. Vaz, “Cad scripting and visual programming languages for implementing computational design concepts: a comparison from a pedagogical point of view,” *International Journal of Architectural Computing*, vol. 10, no. 1, pp. 121–137, 2012.
- [45] “Coap implementations.” Available online: <http://coap.technology/impls.html>. (accessed on 20 December 2015).
- [46] A. Azzara, D. Alessandrelli, S. Bocchino, M. Petracca, and P. Pagano, “Pyot, a macroprogramming framework for the internet of things,” in *In Industrial Embedded Systems (SIES), 2014 9th IEEE International Symposium on, Pisa, Italy*, pp. 96–103, IEEE, 2014.
- [47] M. Kovatsch, S. Mayer, and B. Ostermaier, “Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things,” in *In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, Palermo, Italy*, pp. 751–756, IEEE, 2012.
- [48] M. Kovatsch, M. Lanter, and S. Duquennoy, “Actinium: A restful runtime container for scriptable internet of things applications,” in *In Internet of Things (IOT), 2012 3rd International Conference on the, Wuxi, China*, pp. 135–142, IEEE, 2012.
- [49] “Mqtt.” Available online: <http://mqtt.org/>. (accessed on 12 December 2017).
- [50] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, “Performance evaluation of mqtt and coap via a common middleware,” in *In Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on, TBD, Singapore*, pp. 1–6, IEEE, 2014.
- [51] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, “Comparison of two lightweight protocols for smartphone-based sensing,” in *in Communications and Vehicular*

- Technology in the Benelux (SCVT), 2013 IEEE 20th Symposium on, Namur, Belgium*, pp. 1–6, IEEE, 2013.
- [52] “Iotivity.” Available online: <https://www.iotivity.org/>. (accessed on 12 December 2017).
- [53] M. B. Alaya, Y. Banouar, T. Monteil, C. Chassot, and K. Drira, “Om2m: Extensible etsi-compliant m2m service platform with self-configuration capability,” *Procedia Computer Science*, vol. 32, pp. 1079–1086, 2014.
- [54] “Eclipse kura.” Available online: <http://www.eclipse.org/kura/>. (accessed on 13 December 2017).
- [55] J.-P. Calbimonte, S. Sarni, J. Eberle, and K. Aberer, “Xgsn: An open-source semantic sensing middleware for the web of things,” in *In 7th International Workshop on Semantic Sensor Networks, Riva del Garda, Trentino, Italy*, Oct 2014.
- [56] B. Anggorjati, K. Çetin, A. Mihovska, and N. R. Prasad, “Rfid added value sensing capabilities: European advances in integrated rfid-wsn middleware,” in *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on, Boston, MA, USA*, pp. 1–3, IEEE, 2010.
- [57] “Sensei.” Available online: <http://www.ict-sensei.org/>. (accessed on 28 May 2015).
- [58] “Nimbits.” Available online: <http://www.nimbits.com/>. (accessed on 20 December 2015).
- [59] “Xively-public cloud for internet of things.” Available online: <https://xively.com/>. (accessed on 20 December 2015).
- [60] A. Pintus, D. Carboni, and A. Piras, “Paraimpu: a platform for a social web of things,” in *In Proceedings of the 21st international conference companion on World Wide Web, Lyon, France*, pp. 401–404, ACM, April 2012.
- [61] “Particle.” Available online: <https://www.particle.io/>. (accessed on 20 December 2015).
- [62] “Thinking things.” Available online: <http://www.thinkingthings.telefonica.com/>. (accessed on 20 December 2015).
- [63] “Sensorcloud.” Available online: <http://www.sensorcloud.com/>. (accessed on 20 December 2015).
- [64] “Cloudplugs.” Available online: <http://cloudplugs.com/>. (accessed on 20 December 2015).
- [65] G. Merlino, D. Bruneo, S. Distefano, F. Longo, A. Puliafito, and A. Al-Anbuky, “A smart city lighting case study on an openstack-powered infrastructure,” *Sensors*, vol. 15, no. 7, pp. 16314–16335, 2015.

- [66] “Ripple.” <https://ripple.com/>, 2017. (accessed on 20 October 2017).
- [67] “Litecoin.” <https://litecoin.org/>, 2011. (accessed on 4 March 2018).
- [68] “Nxt white paper.” Available online: <https://bravenewcoin.com/assets/Whitepapers/NxtWhitepaper-v122-rev4.pdf>, 2014. (accessed on 3 March 2018).
- [69] S. King and S. Nadal, “Peercoin—secure & sustainable cryptocoin.” <https://peercoin.net/whitepaper>, 2012. (accessed on 20 October 2017).
- [70] F. Schuh and D. Larimer, “Bitshares 2.0: General overview.” Available online: <https://bravenewcoin.com/assets/Whitepapers/bitshares-general.pdf>, 2017. (accessed on 3 March 2018).
- [71] B. Markus, “Dogecoin.” <http://dogecoin.com/>, 2013. (accessed on 20 October 2017).
- [72] “Namecoin.” <https://namecoin.org/>, 2014. (accessed on 20 October 2017).
- [73] “Dash.” <https://www.dash.org/es/>, 2017. (accessed on 20 October 2017).
- [74] “Monero.” <https://getmonero.org/>, 2017. (accessed on 20 October 2017).
- [75] I. A. Naidu R, “Nestle, unilever, tyson and others team with ibm on blockchain.” <http://www.reuters.com/article/us-ibm-retailers-blockchain/nestle-unilever-tyson-and-others-team-with-ibm-on-blockchain-idUSKCN1B21B1>, 2017. (accessed on 20 October 2017).
- [76] V. Pradeep, “Renault partners with microsoft for blockchain-based digital car maintenance book.” <https://mspoweruser.com/renault-partners-microsoft-blockchain-based-digital-car-maintenance-book/>, 2017. (accessed on 20 October 2017).
- [77] M. C, “The end of passport gates? dubai to test ‘invisible’ airport checks using facial recognition.” <http://www.telegraph.co.uk/technology/2017/06/13/end-passport-gates-dubai-test-invisible-airport-checks-using/>, 2017. (accessed on 20 October 2017).
- [78] “e-identity.” <https://e-estonia.com/solutions/e-identity/e-residency/>, 2017. (accessed on 20 October 2017).
- [79] “How to get an illinois birth certificate online.” <https://vital-records.us/order-an-illinois-birth-certificate/>, 2017. (accessed on 20 October 2017).
- [80] C. R, “Indian states look to digitize land deals with blockchain.” <https://www.reuters.com/article/us-india-landrights-tech/indian-states-look-to-digitize-land-deals-with-blockchain-idUSKBN1AQ1T3>, 2017. (accessed on 20 October 2017).

- [81] “Bitpay.” <https://bitpay.com/>, 2017. (accessed on 20 October 2017).
- [82] “Abra.” <https://www.abra.com/>, 2017. (accessed on 20 October 2017).
- [83] “Bitnation.” <https://bitnation.co/>, 2017. (accessed on 20 October 2017).
- [84] “Onename.” <https://onename.com/>, 2017. (accessed on 20 October 2017).
- [85] “Keybase.” <https://keybase.io/>, 2017. (accessed on 20 October 2017).
- [86] “Shocard.” <https://shocard.com/>, 2017. (accessed on 20 October 2017).
- [87] “Follow my vote.” Available online: <https://followmyvote.com/>, 2017. (accessed on 1 February 2018).
- [88] “Tierion.” <https://tierion.com/>, 2017. (accessed on 20 October 2017).
- [89] “Proof of existence.” <https://poex.io/>, 2017. (accessed on 20 October 2017).
- [90] “Factom.” <https://www.factom.com/>, 2017. (accessed on 20 October 2017).
- [91] “Everledger.” <https://www.everledger.io/>, 2017. (accessed on 20 October 2017).
- [92] “Mit digital diploma pilot program.” <http://web.mit.edu/registrar/records/certs/digital.html>, 2017. (accessed on 20 October 2017).
- [93] “Provenance.” <https://www.provenance.org/>, 2017. (accessed on 20 October 2017).
- [94] “Skuchain.” <http://www.skuchain.com/>, 2017. (accessed on 20 October 2017).
- [95] “Robomed network.” <https://robomed.io/>, 2017. (accessed on 20 October 2017).
- [96] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, “Medrec: Using blockchain for medical data access and permission management,” in *Open and Big Data (OBD), International Conference on, Vienna, Austria*, pp. 25–30, IEEE, 2016.
- [97] “Synechron.” <https://www.synechron.com/finlabs/mortgage-lending>, 2017. (accessed on 20 October 2017).
- [98] “Ubitquity.” <https://www.ubitquity.io/>, 2017. (accessed on 20 October 2017).
- [99] “Atlant.” <https://atlant.io/>, 2017. (accessed on 20 October 2017).
- [100] G. Prisco, “Slock. it to introduce smart locks linked to smart ethereum contracts, decentralize the sharing economy.” Available online: <https://bitcoinmagazine.com/articles/slock-it-to-introduce-smart-locks-linked-to-smart-ethereum-contracts-decentralize-the-sharing-economy-1446746719/>, 2016. (accessed on 1 February 2018).
- [101] “Daocasino.” <https://dao.casino/>, 2017. (accessed on 20 October 2017).
- [102] “Peerplays.” <https://www.peerplays.com/>, 2017. (accessed on 20 October 2017).

- [103] “Wagerr.” <https://wagerr.com/>, 2017. (accessed on 20 October 2017).
- [104] “Storj.io.” <https://storj.io/>, 2017. (accessed on 20 October 2017).
- [105] R. J, “Sony wants to digitize education records using the blockchain.” <https://techcrunch.com/2017/08/09/sony-education-blockchain/>, 2017. (accessed on 20 October 2017).
- [106] “ujo.” <https://ujomusic.com/>, 2017. (accessed on 20 October 2017).
- [107] “resonate.” <https://resonate.is/>, 2017. (accessed on 20 October 2017).
- [108] “Lo3energy.” Available online: <https://lo3energy.com/>, 2017. (accessed on 1 February 2018).
- [109] “Aigang.” Available online: <https://aigang.network/>, 2017. (accessed on 1 February 2018).
- [110] “My bit.” Available online: <https://mybit.io/>, 2017. (accessed on 1 February 2018).
- [111] V. Buterin, “Ethereum white paper.” Available online: <https://github.com/ethereum/wiki/wiki/White-Paper>, 2013. (accessed on 4 February 2018).
- [112] P. Veena, S. Panikkar, S. Nair, and P. Brody, “Empowering the edge-practical insights on a decentralized internet of things,” *Empowering the Edge-Practical Insights on a Decentralized Internet of Things*. IBM Institute for Business Value, vol. 17, 2015.
- [113] “Aerotoken.” Available online: <https://aerotoken.com>, 2017. (accessed on 1 February 2018).
- [114] “Chain of things.” Available online: <https://www.chainofthings.com/>, 2017. (accessed on 1 February 2018).
- [115] “Chronicled.” Available online: <https://chronicled.com/>, 2017. (accessed on 1 February 2018).
- [116] “modum.” Available online: <https://modum.io/>, 2017. (accessed on 1 February 2018).
- [117] “Riddle and code.” Available online: <https://www.riddleandcode.com>, 2017. (accessed on 1 February 2018).
- [118] “Chain of things.” Available online: <https://www.blockchainofthings.com/>, 2017. (accessed on 1 February 2018).
- [119] M. Fowler, *Patterns of enterprise application architecture*. Addison-Wesley Longman Publishing Co., Inc., 2002.
- [120] “Constrained restful environments (core) link format.” Available online: <https://tools.ietf.org/html/rfc6690>. (accessed on 09 May 2016).
- [121] A. Ludovici and A. Calveras, “A proxy design to leverage the interconnection of coap wireless sensor networks with web applications,” *Sensors*, vol. 15, no. 1, pp. 1217–1244, 2015.

- [122] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota, "Rest enabled wireless sensor networks for seamless integration with web applications," in *In Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on, Valencia, Spain*, pp. 867–872, IEEE, 2011.
- [123] "Basic coap library for arduino." Available online: <https://github.com/dgiannakop/Arduino-CoAP>. (accessed on 12 May 2016).
- [124] "The 10 most popular internet of things applications right now." Available online: <http://iot-analytics.com/10-internet-of-things-applications/>. (accessed on 20 Dec 2015).
- [125] "Libelium." Available online: <http://www.libelium.com/>. (accessed on 16 November 2016).
- [126] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, "Iot-oas: An oauth-based authorization service architecture for secure services in iot scenarios," *Sensors Journal, IEEE*, vol. 15, no. 2, pp. 1224–1234, 2015.
- [127] "50 sensor applications for a smarter world." Available online: http://www.libelium.com/resources/top_50_iiot_sensor_applications_ranking/#show_infographic. (accessed on 12 May 2017).
- [128] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.
- [129] E. Dalipi, F. Van den Abeele, I. Ishaq, I. Moerman, and J. Hoebeke, "Ec-iiot: An easy configuration framework for constrained iiot devices," in *Internet of Things (WF-IIoT), 2016 IEEE 3rd World Forum on, Reston, VA, USA*, pp. 159–164, IEEE, 2016.
- [130] F. Van den Abeele, I. Moerman, P. Demeester, and J. Hoebeke, "Secure service proxy: A coap (s) intermediary for a securer and smarter web of things," *Sensors*, vol. 17, no. 7, p. 1609, 2017.
- [131] Z. Shelby, C. Bormann, and S. Krco, "Core resource directory." Available online: <https://tools.ietf.org/html/draft-ietf-core-resource-directory-09>. (accessed on 1 February 2017).
- [132] F. Van den Abeele, J. Hoebeke, G. K. Teklemariam, I. Moerman, and P. Demeester, "Sensor function virtualization to support distributed intelligence in the internet of things," *Wireless Personal Communications*, vol. 81, no. 4, pp. 1415–1436, 2015.
- [133] "Oma lightweightm2m (lwm2m)." Available online: <http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html>. (accessed on 2 February 2017).
- [134] M. Ruta, F. Scioscia, A. Pinto, E. Di Sciascio, F. Gramegna, S. Ieva, and G. Loseto, "Resource annotation, dissemination and discovery in the semantic web of things: a coap-based framework," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of*

Things (iThings/CPSCoM), *IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pp. 527–534, IEEE, 2013.

- [135] “The websocket protocol.” Available online: <https://tools.ietf.org/html/rfc6455>. (accessed on 16 February 2017).
- [136] “The ionic framework.” Available online: <https://ionicframework.com/>. (accessed on 20 January 2017).
- [137] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester, “Facilitating sensor deployment, discovery and resource access using embedded web services,” in *In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, Palermo, Italy*, pp. 717–724, IEEE, 2012.
- [138] F. V. den Abeele, T. Vandewinckele, J. Hoebeke, I. Moerman, and P. Demeester, “Secure communication in ip-based wireless sensor networks via a trusted gateway,” in *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore*, pp. 1–6, April 2015.
- [139] “Group communication for the constrained application protocol (coap).” Available online: <https://www.ietf.org/rfc/rfc7390.txt>. (accessed on 14 February 2017).
- [140] I. Ishaq, J. Hoebeke, I. Moerman, and P. Demeester, “Observing coap groups efficiently,” *Ad Hoc Networks*, vol. 37, pp. 368–388, 2016.
- [141] “Homelab demo video.” Available online: <https://www.youtube.com/watch?v=i0B-rYuKJfY>. (accessed on 25 January 2018).
- [142] “Homelab ghent university.” Available online: <https://www.ugent.be/ea/idlab/en/research/research-infrastructure/homelab.htm>. (accessed on 25 January 2018).
- [143] A. B. Zaslavsky, C. Perera, and D. Georgakopoulos, “Sensing as a service and big data,” in *In International Conference on Advances in Cloud Computing (ACC-2012), Bangalore, India*, pp. 21–29, July 2012.
- [144] D. Balageas, C.-P. Fritzen, and A. Güemes, *Structural health monitoring*, vol. 90. John Wiley & Sons, 2010.
- [145] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.
- [146] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The hadoop distributed file system,” in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, Incline Villiage, Nevada, USA*, pp. 1–10, May May 2010.
- [147] “Mapreduce tutorial.” Available online: http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html. (accessed on 20 December 2015).

- [148] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, *et al.*, “Apache hadoop yarn: Yet another resource negotiator,” in *In Proceedings of the 4th annual Symposium on Cloud Computing, Santa Clara, CA, USA*, p. 5, ACM, Oct 2013.
- [149] “Apache hive.” Available online: <https://hive.apache.org/>. (accessed on 20 December 2015).
- [150] “Apache pig.” Available online: <https://pig.apache.org/>. (accessed on 20 December 2015).
- [151] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: cluster computing with working sets,” in *In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, Boston, MA, USA*, pp. 10–10, June 2010.
- [152] “Apache spark.” Available online: <https://spark.apache.org/>. (accessed on 20 December 2015).
- [153] F. Yang, E. Tschetter, X. Léauté, N. Ray, G. Merlino, and D. Ganguli, “Druid: a real-time analytical data store,” in *In Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 157–168, ACM, 2014.
- [154] “Druid.” Available online: <http://druid.io/>. (accessed on 20 December 2015).
- [155] “Apache hbase.” Available online: <http://hbase.apache.org/>. (accessed on 20 December 2015).
- [156] “Apache phoenix.” Available online: <http://phoenix.apache.org/>. (accessed on 20 December 2015).
- [157] “Cloudera impala.” Available online: <http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html>. (accessed on 20 December 2015).
- [158] “Opentsdb.” Available online: <http://opentsdb.net/>. (accessed on 20 December 2015).
- [159] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, *et al.*, “Storm@ twitter,” in *In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, Snowbird, Utah, USA*, pp. 147–156, ACM, 2014.
- [160] M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica, “Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters,” in *In Proceedings of the 4th USENIX conference on Hot Topics in Cloud Computing, Boston, MA, USA*, pp. 10–10, USENIX Association, June 2012.
- [161] “Apache spark streaming.” Available online: <https://spark.apache.org/streaming/>. (accessed on 20 December 2015).

- [162] J. Kreps, N. Narkhede, J. Rao, *et al.*, “Kafka: A distributed messaging system for log processing,” in *In Proceedings of 6th International Workshop on Networking Meets Databases (NetDB), Athens, Greece, 2011*.
- [163] “Apache kafka.” Available online: <http://kafka.apache.org/>. (accessed on 20 December 2015).
- [164] D. Dossot, *RabbitMQ Essentials*. Packt Publishing Ltd, 2014.
- [165] “Rabbitmq.” Available online: <https://www.rabbitmq.com/>. (accessed on 20 December 2015).
- [166] “Apache ambari.” Available online: <http://ambari.apache.org/>. (accessed on 20 December 2015).
- [167] K. Aberer, M. Hauswirth, and A. Salehi, “The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks,” tech. rep., 2006. Submitted to ACM/IFIP/USENIX 7th International Middleware Conference.
- [168] D. Le-Phuoc, H. Q. Nguyen-Mau, J. X. Parreira, and M. Hauswirth, “A middleware framework for scalable management of linked streams,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 16, pp. 42–51, 2012.
- [169] F. Jammes, A. Mensch, and H. Smit, “Service-oriented device communications using the devices profile for web services,” in *In Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, Grenoble, France*, pp. 1–8, ACM, 2005.
- [170] “Web services for devices.” Available online: <http://ws4d.org/>. (accessed on 20 December 2015).
- [171] “Microsoft web services on devices.” Available online: [https://msdn.microsoft.com/en-us/library/windows/desktop/aa826001\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa826001(v=vs.85).aspx). (accessed on 20 December 2015).
- [172] J. Cubo, A. Nieto, and E. Pimentel, “A cloud-based internet of things platform for ambient assisted living,” *Sensors*, vol. 14, no. 8, pp. 14070–14105, 2014.
- [173] “The omg data-distribution service for real-time systems (dds).” Available online: <http://portals.omg.org/dds/>. (accessed on 20 December 2015).
- [174] C. Bormann, A. P. Castellani, and Z. Shelby, “Coap: An application protocol for billions of tiny internet nodes,” *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.
- [175] M. Castro, A. J. Jara, and A. F. Skarmeta, “Enabling end-to-end coap-based communications for the web of things,” *Journal of Network and Computer Applications*, pp. –, 2014.
- [176] “Linksmart.” Available online: <https://linksmart.eu/>. (accessed on 20 December 2015).

- [177] D. Hughes, K. L. Man, Z. Shen, and K. K. Kim, "A loosely-coupled binding model for wireless sensor networks," in *In SoC Design Conference (ISOCC), 2012 International, Jeju Island, Korea (South)*, pp. 273–276, IEEE, Nov 2012.
- [178] J. Maerien, S. Michiels, D. Hughes, C. Huygens, and W. Joosen, "Seclooci: A comprehensive security middleware architecture for shared wireless sensor networks," *Ad Hoc Networks*, vol. 25, pp. 141–169, 2015.
- [179] N. Marz and J. Warren, *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.
- [180] M. Villari, A. Celesti, M. Fazio, and A. Puliafito, "Alljoyn lambda: An architecture for the management of smart environments in iot," in *In Smart Computing Workshops (SMART-COMP Workshops), 2014 International Conference on, Hong Kong, China*, pp. 9–14, IEEE, 2014.
- [181] R. Morabito, R. Petrolo, V. Loscri, and N. Mitton, "Legiot: a lightweight edge gateway for the internet of things," *Future Generation Computer Systems*, vol. 81, pp. 1–15, 2018.
- [182] R. Morabito, J. Kjällman, and M. Komu, "Hypervisors vs. lightweight virtualization: a performance comparison," in *Cloud Engineering (IC2E), 2015 IEEE International Conference on, Tempe, AZ, USA*, pp. 386–393, IEEE, 2015.
- [183] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, 2013.
- [184] S. Ziegler, C. Crettaz, and I. Thomas, "Ipv6 as a global addressing scheme and integrator for the internet of things and the cloud," in *In Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on, Victoria, Canada*, pp. 797–802, IEEE, May 2014.
- [185] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 414–454, 2014.
- [186] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 4, pp. 2233–2243, 2014.
- [187] "World health organization food safety fact sheet." Available online: <http://www.who.int/mediacentre/factsheets/fs399/en/>, 2017. (accessed on 1 February 2018).
- [188] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." Available online: <https://bitcoin.org/bitcoin.pdf>, 2008. (accessed on 1 February 2018).
- [189] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies*. O'Reilly Media, Inc., 2014.
- [190] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, 2017.

- [191] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, *et al.*, “Hyperledger fabric: A distributed operating system for permissioned blockchains,” *arXiv preprint arXiv:1801.10228*, 2018.
- [192] J. Kennedy, “\$1.4bn investment in blockchain start-ups in last 9 months, says pwc expert.” Available online: <http://linkis.com/Ayjzj>, 2016. (accessed on 1 February 2018).
- [193] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, “Bitcoin-ng: A scalable blockchain protocol,” in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16)*, Santa Clara, CA, USA, pp. 45–59, 2016.
- [194] Y. Sompolinsky and A. Zohar, “Accelerating bitcoin’s transaction processing,” *Fast Money Grows on Trees, Not Chains. IACR Cryptology ePrint Archive*, vol. 881, 2013.
- [195] C. Decker and R. Wattenhofer, “A fast and scalable payment network with bitcoin duplex micropayment channels,” in *Symposium on Self-Stabilizing Systems, Edmonton, AB, Canada*, pp. 3–18, Springer, 2015.
- [196] C. Stathakopoulou, C. Decker, and R. Wattenhofer, *A faster Bitcoin network*. Tech. rep., ETH, Zurich, Semester Thesis, 2015.
- [197] “Bigchaindb: The scalable blockchain database powering ipdb.” Available online: <https://www.bigchaindb.com/>, 2017. (accessed on 1 February 2018).
- [198] “Ipfs is the distributed web.” Available online: <https://ipfs.io/>, 2017. (accessed on 1 February 2018).
- [199] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, “A survey on the security of blockchain systems,” *Future Generation Computer Systems*, p. In press, 2017.
- [200] I. Eyal and E. G. Sirer, “Majority is not enough: Bitcoin mining is vulnerable,” in *International conference on financial cryptography and data security, San Juan, Puerto Rico*, pp. 436–454, Springer, 2014.
- [201] J. Bonneau, E. W. Felten, S. Goldfeder, J. A. Kroll, and A. Narayanan, “Why buy when you can rent? bribery attacks on bitcoin consensus,” 2016.
- [202] G. Karame, E. Androulaki, and S. Capkun, “Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin,” *IACR Cryptology ePrint Archive*, vol. 2012, no. 248, 2012.
- [203] “Bitcoin average transaction confirmation time.” Available online: <https://blockchain.info/es/charts/avg-confirmation-time>, 2017. (accessed on 1 February 2018).
- [204] H. Finney, “The finney attack(the bitcoin talk forum).” Available online: <https://bitcointalk.org/index.php?topic=3441.msg48384>, 2011. (accessed on 1 February 2018).

- [205] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, “Eclipse attacks on bitcoin’s peer-to-peer network,” in *USENIX Security Symposium, Washington, D.C., USA*, pp. 129–144, USENIX Association, 2015.
- [206] “Segwit2x backers cancel plans for bitcoin hard fork.” Available online: <https://techcrunch.com/2017/11/08/segwit2x-backers-cancel-plans-for-bitcoin-hard-fork/>, 2017. (accessed on 1 February 2018).
- [207] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, “Zerocash: Decentralized anonymous payments from bitcoin,” in *Security and Privacy (SP), 2014 IEEE Symposium on, San Jose, CA, USA*, pp. 459–474, IEEE, 2014.
- [208] I. Miers, C. Garman, M. Green, and A. D. Rubin, “Zerocoin: Anonymous distributed e-cash from bitcoin,” in *Security and Privacy (SP), 2013 IEEE Symposium on, Berkeley, CA, USA*, pp. 397–411, IEEE, 2013.
- [209] “Bitcoin fog.” Available online: <http://bitcoinfofog.info/>, 2017. (accessed on 1 February 2018).
- [210] G. Maxwell, “Coinjoin: Bitcoin privacy for the real world.” Available online: <https://bitcointalk.org/index.php?topic=279249.msg2983902#msg2983902>, 2013. (accessed on 1 February 2018).
- [211] A. Greenberg, “‘dark wallet’ is about to make bitcoin money laundering easier than ever,” *URL* <http://www.wired.com/2014/04/dark-wallet>, 2014.
- [212] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, “Mixcoin: Anonymity for bitcoin with accountable mixes,” in *International Conference on Financial Cryptography and Data Security, San Juan, Puerto Rico*, pp. 486–504, Springer, 2014.
- [213] T. Ruffing, P. Moreno-Sanchez, and A. Kate, “Coinshuffle: Practical decentralized coin mixing for bitcoin,” in *European Symposium on Research in Computer Security, Heraklion, Crete, Greece*, pp. 345–364, Springer, 2014.
- [214] G. Maxwell, “Coinswap: Transaction graph disjoint trustless trading,” *CoinSwap: Transactiongraphdisjointrustlesstrading (October 2013)*, 2013.
- [215] L. Valenta and B. Rowan, “Blindcoin: Blinded, accountable mixes for bitcoin,” in *International Conference on Financial Cryptography and Data Security, San Juan, Puerto Rico*, pp. 112–126, Springer, 2015.
- [216] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, “Hawk: The blockchain model of cryptography and privacy-preserving smart contracts,” in *Security and Privacy (SP), 2016 IEEE Symposium on, San Jose, CA, USA*, pp. 839–858, IEEE, 2016.
- [217] G. Zyskind, O. Nathan, and A. Pentland, “Enigma: Decentralized computation platform with guaranteed privacy,” *arXiv preprint arXiv:1506.03471*, 2015.
- [218] “Quorum whitepaper.” Available online: <https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum%20Whitepaper%20v0.1.pdf>, 2016. (accessed on 1 February 2018).

- [219] G. Greenspan, “Multichain private blockchain—white paper.” Available online: <https://www.multichain.com/download/MultiChain-White-Paper.pdf>, 2015. (accessed on 1 February 2018).
- [220] S. Jehan, “Rockchain a distributed data intelligence platform.” <https://icobazaar.com/static/4dd610d6601de7fe70eb5590b78ed7cd/RockchainWhitePaper.pdf>, 2017. (accessed on 20 October 2017).
- [221] A. Lazarovich, *Invisible Ink: blockchain for data privacy*. PhD thesis, Massachusetts Institute of Technology, 2015.
- [222] G. Zyskind, O. Nathan, *et al.*, “Decentralizing privacy: Using blockchain to protect personal data,” in *Security and Privacy Workshops (SPW), 2015 IEEE, San Jose, CA, USA*, pp. 180–184, IEEE, 2015.
- [223] D. Houlding, “Healthcare blockchain: What goes on chain stays on chain.” Available online: <https://itpeernetwork.intel.com/healthcare-blockchain-goes-chain-stays-chain/>, 2017. (accessed on 1 February 2018).
- [224] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, “Town crier: An authenticated data feed for smart contracts,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria*, pp. 270–282, ACM, 2016.
- [225] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, “Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab,” in *International Conference on Financial Cryptography and Data Security, Christ Church, Barbados*, pp. 79–94, Springer, 2016.
- [226] N. Atzei, M. Bartoletti, and T. Cimoli, “A survey of attacks on ethereum smart contracts (sok),” in *International Conference on Principles of Security and Trust, Uppsala, Sweden*, pp. 164–186, Springer, 2017.
- [227] K. Christidis and M. Devetsikiotis, “Blockchains and smart contracts for the internet of things,” *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
- [228] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, “Making smart contracts smarter,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria*, pp. 254–269, ACM, 2016.
- [229] C. K. Frantz and M. Nowostawski, “From institutions to code: towards automated generation of smart contracts,” in *Foundations and Applications of Self* Systems, IEEE International Workshops on, Augsburg, Germany*, pp. 210–215, IEEE, 2016.
- [230] C. K. Elwell, M. M. Murphy, and M. V. Seitzinger, “Bitcoin: questions, answers, and analysis of legal issues.” Available online: <https://fas.org/sgp/crs/misc/R43339.pdf>, 2013. (accessed on 1 February 2018).
- [231] “Bitcoin is a fraud that will blow up, says jp morgan boss.” Available online: <https://www.theguardian.com/technology/2017/sep/13/bitcoin->

- fraud-jp-morgan-cryptocurrency-drug-dealers, 2017. (accessed on 1 February 2018).
- [232] “Bitcoin could be here for 100 years but it’s more likely to ‘totally collapse,’ nobel laureate says.” Available online: <https://www.cnbc.com/2018/01/19/bitcoin-likely-to-totally-collapse-nobel-laureate-robert-shiller-says.html>, 2018. (accessed on 1 February 2018).
- [233] “Bitcoin could hit \$100,000 in 10 years, says the analyst who correctly called its \$2,000 price.” Available online: <https://www.cnbc.com/2017/05/31/bitcoin-price-forecast-hit-100000-in-10-years.html>, 2017. (accessed on 1 February 2018).
- [234] E. B. Centralny, “Virtual currency schemes—a further analysis.” Available online: <https://www.ecb.europa.eu/pub/pdf/other/virtualcurrencyschemesen.pdf>, 2015. (accessed on 1 February 2018).
- [235] “Bitlegal.” Available online: <http://bitlegal.io/>, 2017. (accessed on 1 February 2018).
- [236] “Regulatory fears hammer bitcoin below \$10,000, half its peak.” Available online: <https://www.reuters.com/article/uk-global-bitcoin/regulatory-fears-hammer-bitcoin-below-10000-half-its-peak-idUSKBN1F60CG>, 2017. (accessed on 1 February 2018).
- [237] “R3.” Available online: <https://www.r3.com/>, 2017. (accessed on 1 February 2018).
- [238] “Trusted iot alliance.” Available online: <https://www.trusted-iot.org/>, 2017. (accessed on 1 February 2018).
- [239] “Alastria: National blockchain ecosystem.” Available online: <https://alastria.io/>, 2017. (accessed on 1 February 2018).
- [240] C. Cachin and M. Vukolić, “Blockchains consensus protocols in the wild,” *arXiv preprint arXiv:1707.01873*, 2017.
- [241] A. Baliga, “Understanding blockchain consensus models.” Available online: <https://www.persistent.com/wp-content/uploads/2017/04/WP-Understanding-Blockchain-Consensus-Models.pdf>, 2017. (accessed on 3 March 2018).
- [242] F. Tschorsch and B. Scheuermann, “Bitcoin and beyond: A technical survey on decentralized digital currencies,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.
- [243] N. T. Courtois, “On the longest chain rule and programmed self-destruction of crypto currencies,” *arXiv preprint arXiv:1405.0534*, 2014.
- [244] I. Stewart, “Proof of burn. bitcoin. it.” Available online: https://en.bitcoin.it/wiki/Proof_of_burn, 2012. (accessed on 4 March 2018).

- [245] A. Nember, “Nem technical reference.” Available online: https://nem.io/wp-content/themes/nem/files/NEM_techRef.pdf, 2018. (accessed on 4 March 2018).
- [246] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, “Permacoin: Repurposing bitcoin work for data preservation,” in *Security and Privacy (SP), 2014 IEEE Symposium on, San Jose, CA, USA*, pp. 475–490, IEEE, 2014.
- [247] L. Lamport *et al.*, “Paxos made simple,” *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.
- [248] M. Burrows, “The chubby lock service for loosely-coupled distributed systems,” in *Proceedings of the 7th symposium on Operating systems design and implementation, Seattle, WA, USA*, pp. 335–350, USENIX Association, 2006.
- [249] D. Ongaro and J. K. Ousterhout, “In search of an understandable consensus algorithm.” in *USENIX Annual Technical Conference, Philadelphia, PA, USA*, pp. 305–319, USENIX Association, 2014.
- [250] M. Nabi-Abdolyousefi and M. Mesbahi, “Sieve method for consensus-type network tomography,” *IET Control Theory & Applications*, vol. 6, no. 12, pp. 1926–1932, 2012.
- [251] D. Mazieres, “The stellar consensus protocol: A federated model for internet-level consensus,” *Stellar Development Foundation*, 2015.
- [252] “Hdac.” Available online: <https://hdac.io/>, 2017. (accessed on 1 February 2018).
- [253] V. Gramoli, “From blockchain consensus back to byzantine consensus,” *Future Generation Computer Systems*, 2017.
- [254] J. C. Buzby and T. Roberts, “The economics of enteric infections: human foodborne disease costs,” *Gastroenterology*, vol. 136, no. 6, pp. 1851–1862, 2009.
- [255] H. Malviya, “How blockchain will defend iot.” Available online: <https://ssrn.com/abstract=2883711>, 2016. (accessed on 1 February 2018).
- [256] S. Gan, *An IoT simulator in NS3 and a key-based authentication architecture for IoT devices using blockchain*. PhD thesis, Indian Institute of Technology Kanpur, 2017.
- [257] “Filament.” Available online: <https://filament.com/>, 2017. (accessed on 1 February 2018).
- [258] M. A. Khan and K. Salah, “Iot security: Review, blockchain solutions, and open challenges,” *Future Generation Computer Systems*, 2017.
- [259] M. Samaniego and R. Deters, “Hosting virtual iot resources on edge-hosts with blockchain,” in *Computer and Information Technology (CIT), 2016 IEEE International Conference on, Yanuca Island, Fiji*, pp. 116–119, IEEE, 2016.
- [260] M. Aazam and E.-N. Huh, “Fog computing and smart gateway based communication for cloud of things,” in *In Proceedings of the 2nd International Conference on Future Internet of Things and Cloud (FiCloud-2014), Barcelona, Spain*, pp. 27–29, Aug 2014.

- [261] “Ethembedded.” Available online: <http://ethembedded.com/>, 2017. (accessed on 1 February 2018).
- [262] “Raspnode.” Available online: <http://raspnode.com/>, 2017. (accessed on 1 February 2018).
- [263] K. Wüst and A. Gervais, “Do you need a blockchain?,” *IACR Cryptology ePrint Archive*, vol. 2017, p. 375, 2017.
- [264] “Ant router rl-ltc the wifi router that mines litecoin.” Available online: https://shop.bitmain.com/antrouter_rl_ltc_wireless_router_and_asic_litecoin_miner.htm, 2017. (accessed on 1 February 2018).
- [265] “Ethraspbian.” Available online: <http://ethraspbian.com/>, 2017. (accessed on 1 February 2018).
- [266] R. Roman, J. Lopez, and M. Mambo, “Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges,” *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.
- [267] J. Lopez, R. Rios, F. Bao, and G. Wang, “Evolving privacy: From sensors to the internet of things,” *Future Generation Computer Systems*, vol. 75, pp. 46–57, 2017.
- [268] M. Banerjee, J. Lee, and K.-K. R. Choo, “A blockchain future to internet of things security: A position paper,” *Digital Communications and Networks*, 2017.
- [269] C. Fernandez-Gago, F. Moyano, and J. Lopez, “Modelling trust dynamics in the internet of things,” *Information Sciences*, vol. 396, pp. 72–82, 2017.
- [270] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, “Mur-dpa: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud,” *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2609–2622, 2015.
- [271] C. Wang, Q. Wang, K. Ren, and W. Lou, “Privacy-preserving public auditing for data storage security in cloud computing,” in *INFOCOM, 2010 Proceedings IEEE, San Diego, California, USA*, pp. 1–9, Ieee, 2010.
- [272] C. Liu, C. Yang, X. Zhang, and J. Chen, “External integrity verification for outsourced big data in cloud and iot: A big picture,” *Future Generation Computer Systems*, vol. 49, pp. 58–67, 2015.
- [273] “Bitcoin fog.” Available online: <http://www.the-blockchain.com/2016/05/01/babelchain-machine-communication-proof-understanding-new-paper/>, 2016. (accessed on 1 February 2018).
- [274] M. Samaniego and R. Deters, “Internet of smart things-iiot: Using blockchain and clips to make things autonomous,” in *Cognitive Computing (ICCC), 2017 IEEE International Conference on, Honolulu, Hawaii, USA*, pp. 9–16, IEEE, 2017.
- [275] “The lisk protocol.” Available online: <https://docs.lisk.io/docs/the-lisk-protocol>, 2017. (accessed on 1 February 2018).



(Dalías, Almería, Spain). “*Buscando norte y guía, siguiendo sendas de virtud.*”