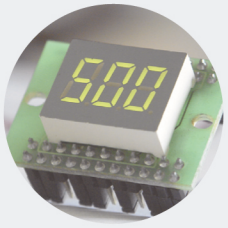
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Resilient IoT Network for short-range data transmission

GONÇALO LOPES BAPTISTA SALAZAR
novembro de 2018

POLITÉCNICO
DO PORTO

Resilient IoT Network for short-range data transmission

Gonçalo Lopes Baptista Salazar

Department of Electrical Engineering

Instituto Superior de Engenharia do Porto

2018

This report fulfills the requirements contained in TEDI course form, of the 2nd year of the Masters in Electrical and Computer Engineering, Autonomous Systems

E-mail:

1160015@isep.ipp.pt

Advisor: Dr. Lino Figueiredo, lbf@isep.ipp.pt

Company: CEiiA

Supervisor: Eng. Nuno Ferreira, nuno.ferreira@ceiia.com

Department of Electrical Engineering

Instituto Superior de Engenharia do Porto

November 15, 2018

Ackowledgements

Firstly, I would like to thank my advisor Dr. Lino Figueiredo for his support, expertise and guidance throughout all the project.

I would also like to thank my supervisor Eng. Nuno Ferreira for his continued help and motivation, ensuring that this project reached its goal and all the members of the EES team, both current and past, for all the help, knowledge and environment that helped me see the end of the work.

Finally, I would like to thank my parents, my brother, Amigão, Catarina and the rest of my family for all their support and encouragement during this return to school.

This page intentionally left blank.

Abstract

The current trends predict an increase of Internet of Things (IoT) devices to the billionths, changing the everyday life of the human race everywhere around the world. However, in order to work reliably these devices will require a means of communication that leverages current technologies allowing them to transmit data at short distances.

The ubiquity of Bluetooth Low Energy (BLE) alongside the solid foundations of IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) will enable the creation of vast networks of interconnected devices in a reliable and transparent way. Nevertheless the validation of such technologies combination is key before the large deployment of such networks can be started.

This work validated such a setup, proving that the reliable communication between devices using 6LoWPAN over BLE can be achieved. The project obtained promising results in terms of transparency between this stack and more traditional Internet Protocol (IP) stacks for multiple distances.

The conclusions obtained open the possibility for real world scenario testing and small scale deployment for further validation of 6LoWPAN over BLE.

Keywords

Internet of things, Bluetooth Low Energy, 6LoWPAN, IPv6

This page intentionally left blank.

Resumo

As tendências actuais prevêm um aumento de dispositivos de IoT para a ordem dos milhares de milhão, provocando mudanças no dia a dia de pessoas por todo o mundo. Contudo, para garantir que estes dispositivos funcionam de forma fiável, irão necessitar de um meio de comunicação que alavanque as technologias actuais de forma a permitir a comunicação a curtas distâncias.

A ubiquidade do BLE combinado com as bases sólidas do 6LoWPAN irão possibilitar a criação de vastas redes de dispositivos ligados entre si de forma confiável e transparente. Contudo a validação desta combinação de tecnologias é essencial antes da implantação em larga escala destes sistemas.

Este trabalho validou esta combinação de tecnologias, provando que comunicação entre dispositivos utilizando 6LoWPAN sobre BLE é possível. Este projecto obteve resultados promissores em termos de transparências entre esta e *stacks* mais tradicionais de protocolos de internet.

As conclusões obtidas abrem a possibilidade de testes em cenários reais e de instalação de pequenas redes para maior validação de 6LoWPAN sobre BLE.

Palavras-chave

Internet das coisas, Bluetooth Low-Energy, 6LoWPAN, IPv6

This page intentionally left blank.

Contents

List of Figures

This page intentionally left blank.

List of Tables

This page intentionally left blank.

Acronyms

3G 3rd Generation

3GPP 3rd Generation Partnership Project

4G 4th Generation

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks

ACL Asynchronous Connection-Less

AMQP Advanced Message Queueing Protocol

API Application Programming Interface

ASCII American Standard Code for Information Interchange

ATT Attribute Protocol

BIND Berkeley Internet Name Domain

BLE Bluetooth Low Energy

CAN Controller Area Network

CAT-M1 Category M1

CBOR Concise Binary Object Representation

CEiiA *Centro de Engenharia e Desenvolvimento*

CID Channel Identifier

CID Context Identifier Extension

CoAP Constrained Application Protocol

CoRE Constrained RESTful Environments

CTS Current Time Service

CRC Cyclic Redundancy Check

DAC Destination Address Compression

DAD Duplicate Address Detection

DAM Destination Address Mode

DCI Destination Context Identifier

dI2C differential Inter-Integrated Circuit (I^2C)

DDS Data Distribution Service

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

DSR Dynamic Source Routing

DTLS Datagram Transport Layer Security

EID IPv6 Extension Header Identifier

EIGRP Enhanced Interior Gateway Routing Protocol

FDMA Frequency Division Modulation Access

GAP Generic Access Profile

GATT Generic Attribute Profile

GPS Global Positioning System

GSM Global System for Mobile Communications

HCI Host Controller Interface

HIDS Human Interface Device Service

HLIM Hop Limit

HSPA High Speed Packet Access

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

I^2C Inter-Integrated Circuit

ICMP Internet Control Message Protocol

ICMPv6 Internet Control Message Protocol version 6

ID Identifier

IID Interface Identifier

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IoT Internet of Things

ICTs Information and Communication Technologies

IP Internet Protocol

IPSP Internet Protocol Support Profile

IPSS Internet Protocol Support Services

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ISEP *Instituto Superior de Engenharia do Porto*

IS-IS Intermediate System to Intermediate System

ISM Industrial, Scientific and Medical

JSON JavaScript Object Notation

L2CAP Logical Link Control and Adaptation Protocol

LE Low Energy

LoRaWAN Long Range Wide Area Network

LoRa Long Range

LPWAN Low Power Wide Area Network

LR-WPANs Low-Rate Wireless Personal Area Networks

LED Light-Emitting Diode

M Multicast Compression

MAC Media Access Control

MCU Microcontroller

MEEC *Mestrado em Engenharia Eletrotécnica e de Computadores*

MIC Message Integrity Check

MPS Maximum Payload Size

MQTT Message Queue Telemetry Transport

MTU Maximum Transmission Unit

NA Neighbour Advertisement

NAT Network Address Translation

NAT64 Network Address Translation IPv6 to IPv4

NB-IoT Narrow-Band IoT

NH Next Header

NS Neighbour Solicitation

OASIS Organization for the Advancement of Structured Information Standards

OLSR Optimized Link State Routing Protocol

OMG Object Management Group

OS Operating System

OSPF Open Shortest Path First

PASP Phone Alert Service Profile

PDU Protocol Data Unit

PHY Physical Layer

POI Points of Interest

QoS Quality of Service

RA Router Advertisement

REST REpresentational State Transfer

RFC Request For Comments

RIP Routing Information Protocol

RPL Routing Protocol for Low Power and Lossy Networks

RS Router Solicitation

RTT Round-Trip Time

SAC Source Address Compression

SAM Source Address Mode

SCI Source Context Identifier

SDIO Secure Digital Input Output

SDU Service Data Unit

SDK Software Development Kit

SIG Special Interest Group

SLAAC Stateless Address Autoconfiguration

SoB System on Board

SSH Secure Shell

SSL Secure Sockets Layer

SOAP Simple Object Access Protocol

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol/ Internet Protocol

TDMA Time Division Multiple Access

TF Traffic Class Flow Label

TLS Transport Layer Security

UART Universal Asynchronous Receiver/Transmitter

UDP User Datagram Protocol

URL Uniform Resource Locator

USB Universal Serial Bus

UUID Universally Unique Identifier

VAC Vehicle Addon Controller

VPN Virtual Private Network

WG Work Group

XML Extensible Markup Language

XMPP Extensible Messaging and Presence Protocol

This page intentionally left blank.

Chapter 1

Introduction

The current trends and predictions all point to an even bigger increase of the connected devices, increasing the number of IoT nodes to the billionths of nodes. This effect is already provoking many changes to the human lifestyle with the always on, always connected devices such as smartphones, but this will increase when it reaches all other devices that interact with people.

The interface and limits between the analog and the digital world will be fuzzed and the everyday objects will make their status available to users in real-time.

To enable these changes the communication methods used will be of utmost importance and will define if the users are able to communicate with all devices or if they will be locked in to a certain vendor or platform. The communication stack will divide itself into two categories; wide range and short range, with the first being able to communicate up to some kilometres and the second to some hundreds of metres.

1.1 Context

This project emerged from the need of *Centro de Engenharia e Desenvolvimento* (CEiiA) to find a solution to reliably send data from the existing devices with lower power and cost at a short range, since the existing solutions communicate using Global System for Mobile Communications (GSM).

This is interesting since for most application the devices accumulate data and are almost always stationary, which leads to the possibility to transmit information not using GSM but some form of proximity communication method. Another important limitation is that sometimes the devices are in low GSM coverage areas and could require the transmission of data to other devices that could send in via other mechanisms.

This context presents therefore an opportunity to evaluate novel communication methods for short-range data transmission that could be implemented in the already existing solutions and complement their functionalities.

1.2 Goals

The main goal of this thesis is to evaluate the possibilities for the setup of a short range IoT communication network.

To perform such task the state of the art must be evaluated to determine what technologies are most common for this application. Since the work is perform within a company some considerations regarding the already adopted technologies and the problems that the company already faces need to be taken into account. Ideally existing products would be used to increase their value.

To validate the chosen technologies some tests need to be perform to evaluate

the usability of the proposed solution and the possibility of integration within the existing solutions.

1.3 Timeline

The project was planned with the different developments required. The timeline is available in Appendice A. Due to the environment in which the thesis was developed some changes occurred however all of the stages were accomplished.

1.4 Structure

This section presents the structure of this thesis, which is divided in the following nine chapters:

After this introductory chapter, the second chapter analyses the state of the art, describing what is the IoT and evaluating some of the network and communication protocols used.

The third chapter presents the BLE technology describing the different functional blocks that are used.

In the fourth chapter, the 6LoWPAN protocol is explained in greater detail to have a better overview of its inner workings.

In the fifth chapter, the problems that will be approached and the system architecture are described considering the current implemented solutions and the proposed ones.

The sixth chapter describes all the steps performed during the development of the system.

In chapter seven the tests and results of this thesis are presented, including the methodology used and the discussed of the obtained results.

The eighth chapter summarises all the conclusions of this work and proposes some future work to further improve on the work presented.

The end of this thesis is reserved for the appendices which contain some more detailed results.

1.5 CEiiA

CEiiA was created in 1999 with the goal of supporting competitiveness of the Portuguese automotive industry. Since then, CEiiA enlarged the activity, and is now focused on aeronautics, mobility, naval/offshore and automotive, always pushing the industry! [1]

Since 2015, CEiiA's headquarters are located in Matosinhos, having offices in Lisbon and Évora, it has a subsidiary in Brazil, engineering teams deployed in France, United Kingdom, Italy and Switzerland and participates in multiple projects around the world.

Figure 1.1: CEiiA's Headquarters [2].

All of the projects described are built around the mobi.me system which

works as a platform to provide mobility services. The goal when developing mobi.me was to connect vehicles and infrastructures, to integrate different information systems and to promote sustainability, offering a comprehensive answer to the needs of users, operators and city authorities. In short, providing mobility as a service, so that users can have access to different services in the required location and the required time.

This page intentionally left blank.

Chapter 2

State of the Art

This chapter tries to define some of the most important concepts regarding networking, gateways and IoT. Some of the most common low-power wireless communication technologies used in IoT networks will be presented and compared regarding their performance and application scenarios.

This chapter will also present a market research to find similar systems to the one presented in this project and the solutions used by those systems.

Based on the findings presented in this chapter and considering CEiiA's projects some considerations will be presented to ensure an easier understanding of the choices made and the problem approach taken.

2.1 Internet of Things

The increase in electronic devices that is currently happening with no signs of slowdown in the near future leads to the ubiquity of such devices, creating what is called the Internet of Things.

Ericsson predicts that in 2022, 29 billion devices will be connected, with 18 billion being IoT devices. Of these IoT devices 16 billion will be short-range IoT devices [3].

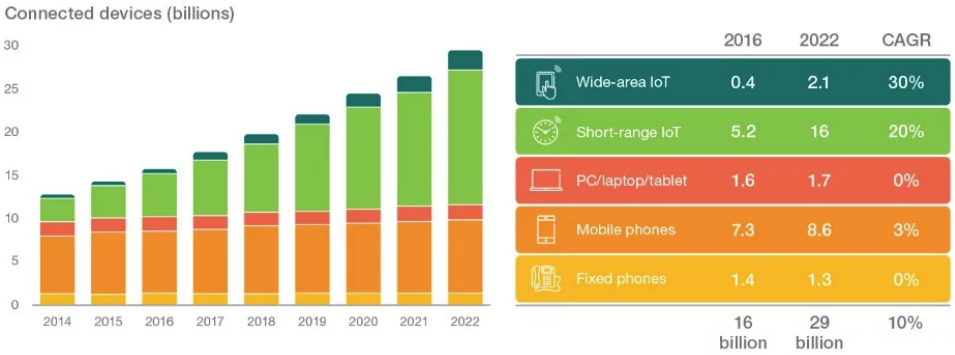


Figure 2.1: IoT growth predictions [3].

"The Internet of Things can be perceived as a far-reaching vision with technological and societal implications." [4]; this implies that the problems and issues that arise from this increase of connected systems and objects are not only a technical problem that relates to the network configurations, power consumption, data volumes and security but also the more human aspects of the increase of connectivity, such as tracking, privacy protection, human-machine interfaces.

The IoT adds a new dimension when compared to traditional Information and Communication Technologies (ICTs) that provides the **"Any Place"** and **"Any Time"**, the **"Any Thing"** dimension as can be seen in figure

2.2 [4].

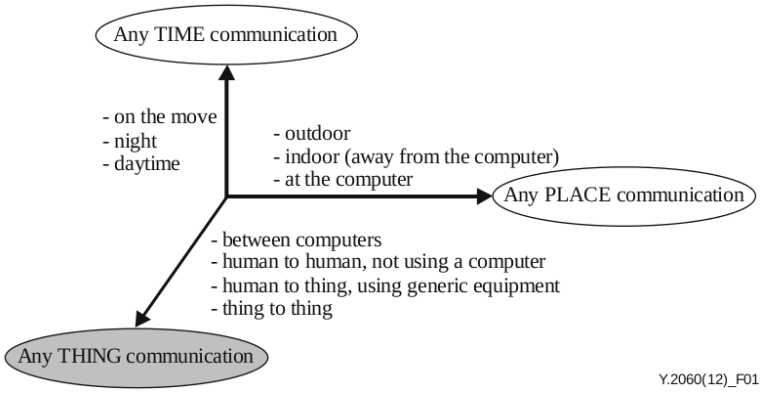


Figure 2.2: The new dimension introduced in the Internet of things [4].

The *things* connected by the IoT can be both physical and virtual and can interact or not with humans. The networks that they form can go from more common already used TCP/IP based networks or to more ad-hoc distributed mesh networks that do not have a standard structure. Figure 2.3 represents the mapping between the physical and the information worlds in the IoT, showing both direct communication between devices and communication with and without gateways.

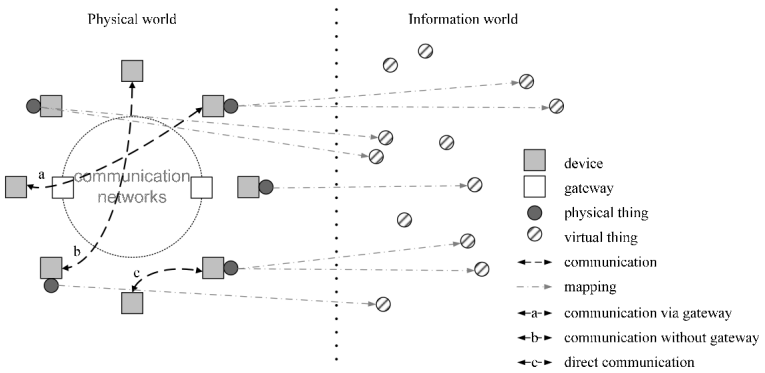


Figure 2.3: IoT networks and interfaces [4].

However to provide some useful information these networks must have a Cloud or a Human interface and a connecting point, whether by using a gateway or via a direct connection to the network.

"There are two important aspects to scaling within the Internet of Things:

- scaling up Internet technologies to a large number of inexpensive nodes, while
- scaling down the characteristics of each of these nodes and of the networks being built out of them, to make this scaling up economically and physically viable."

, according to the *"Terminology for Constrained-Node Networks"* [19].

The applications of such networks have multiple possible applications ranging from industrial control and sensing to mobility industry and smart cities. Allowing the increase of sensors and actuators to automate most of the processes. Nevertheless the decision making cannot be completely offloaded to most of those devices due to the constraints on processing power, data storage and power consumption.

2.1.1 Devices

In such a distributed network not all devices are equal due to the different constraints on the deployment or the application. These devices can have power constraints or not, can be fixed or mobile, can have high or low processing power, etc. Therefore it is required to distinguish between the different types of devices in the networks.

All the devices can be divided in two categories: edge nodes and gateways, however the distinction between the two might sometimes be blurry when

the capabilities of the devices are similar and overlap.

Edge Nodes

Edge nodes are usually low-power devices that have constraints regarding cost and power consumption. These devices are the majority of the IoT devices. Commonly they have one or two Microcontrollers (MCUs) that are connected to both sensors and actuators to gather information and act upon the environment. The devices communicate using low-power network protocols between them, in more ad-hoc closed networks, or with a gateway to ensure the information reaches the Internet.

Figure 2.4 represents a possible architecture for an edge node with two MCUs, where one of them controls the sensors and the actuators and the other is responsible for the communication.

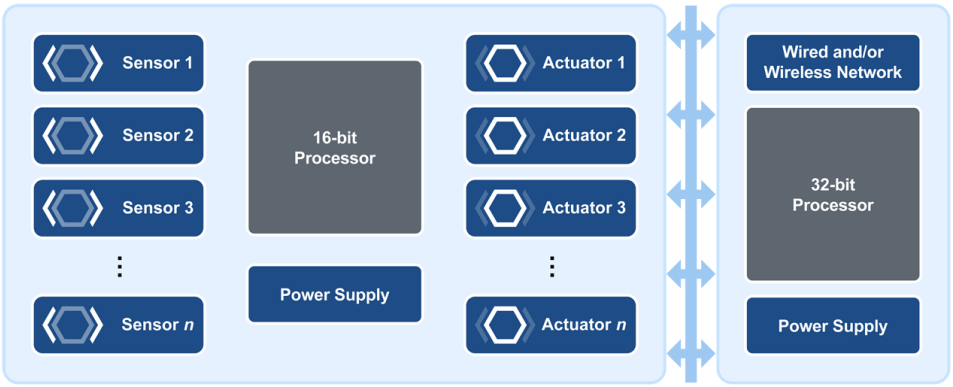


Figure 2.4: Edge Node example architecture [7].

Gateways

Gateways are responsible for translating the information between two different networks guaranteeing the security and separation between both net-

works. Usually gateways are used to connect two different low power networks or a low power network and the internet.

The gateways are usually not as constrained as the edge devices since they have different requirements and the networks would have less of them than edge nodes. The gateways however can also have sensors and actuators of their own.

Figure 2.5 represents a gateway architecture along with some of the protocols it could translate between as well as the security and routing blocks that exist to ensure every message is correctly delivered.

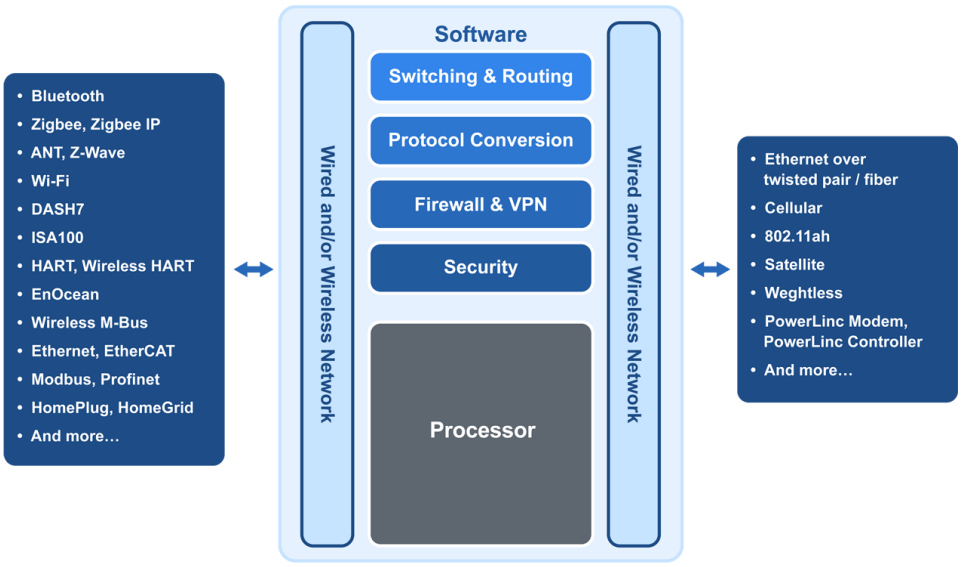


Figure 2.5: Gateway example architecture [7].

2.2 IoT Network Protocols

Multiple wireless network protocols exist, defined by multiple standards and operating in multiple frequencies. Those protocols have their own advantages and disadvantages and different goals and applications. Most of the protocols

presented are based on radio technologies defined by the IEEE organization in the 802.11 and 802.15 standards.

The selection of the most suitable network protocol is dependent of the application and is tied to what the goals, range and data rates requires are.

2.2.1 Wi-Fi

Wi-Fi is a wireless networking technology based on the **IEEE 802.11** standards. It is one of the most pervasive wireless technologies used, being present in most consumer and industrial electronic devices. The large deployment of this technology makes it easy to use and to develop applications based on it due the existing knowledge.

Wi-Fi works on both the 2.4 GHz and 5 GHz frequencies and has data rates between 2 Mbps and 1.73 Gbps [9] depending on the frequency and modulation used. However the power consumption of Wi-Fi setups is quite high making sometimes unsuitable for low-power applications that characterize most IoT applications.

2.2.2 LoRaWAN

Long Range Wide Area Network (LoRaWAN) is a Low Power Wide Area Network (LPWAN) technology targeting the IoT requirements of security, bi-directional communication and long-range communications. The network is laid out in a star of stars topology where gateways act as transparent bridges between edge nodes and a central server. The communication between edge nodes and the gateways is performed at multiple frequencies and data rates, the selection of the data rate is a trade-off between communication range and message duration. [10].

LoRaWAN is based on the LoRa modulation, a proprietary modulation system, which operates on the unlicensed spectrum, which means that anyone can setup a network. It can operate with data rates between 0.3 kbps and 50 kbps, with ranges 2 to 5 km in urban environment and 15 km in suburban environment [8] and low power consumption.

2.2.3 BLE

BLE is a wireless low-power technology that operates in the 2.4 GHz frequency, the same as the Classic Bluetooth, but is optimized for battery consumption. It is capable of communicating with data rates between 125 kbps and 2 Mbps and can operate in three different topologies: point-to-point, broadcast and mesh. [15], [16] BLE has a maximum line of sight range of 300 m.

BLE is currently one of the most widely distributed technologies present in devices, from smartphones to IoT application in automation. BLE was designed to ensure that it does not suffer from interference from Wi-Fi and other technologies operating in that frequency range. It was also designed with power efficiency in mind, ensuring that low-power consumption is obtained both during communication and also while sleeping. [17]

The three BLE topologies have different applications in mind. Point-to-point is used in one to one applications similar to Bluetooth Classic, for instance pairing smartphones with speakers or headsets. Broadcast is a one-to-many connection used for sharing Points of Interest (POI) or location information, usually using a BLE beacon. Mesh is used in many to many scenarios, to establish ad-hoc networks used for large scale deployment of networks for applications such as building automation, sensor networks or asset tracking. [16]

2.2.4 ZigBee

ZigBee is a low-power wireless technology that operates on the 2.4 GHz frequency with data rates of 250 kbps and ranges of up to 300 meters in line of sight. It is a technology currently deploy in many industrial and home automation scenarios where power consumption is an issue.[14]

The networks can have multiple structures from star to mesh. Multiple ZigBee protocols exist and not all of them are compatible, making some specifications not able to talk with each other.

Standard networks usually have a coordinator responsible for gathering and relaying messages between nodes, when configured in a mesh network the coordinator responsibility is to accept new nodes onto the network, however the network can live even if the coordinator is no longer present.

2.2.5 Cellular

Multiple cellular based communication solutions exist, however, all require an operator to setup the infrastructure and therefore fees are always required. They are characterized by long range (ie: 35 km for GSM and 200 km for HSPA) and operate in licensed radio bands (900/1800/1900/2100 MHz). Multiple standards and technologies exist for cellular based communication, defined by 3rd Generation Partnership Project (3GPP), that can be divided into two groups: standard and low-power.

Standard cellular network include GSM/3G/4G which are usually used for communication due to the high availability of the service, the big coverage, high data rates and long range. However these technologies have a high power consumption not making them suitable for low-power applications.

Low-power standards, Category M1 (CAT-M1) and Narrow-Band IoT (NB-IoT)

are more recent standards presented by 3GPP that are design for different types of applications. Both these technologies operate in the same licensed spectrum that the standard cellular network technologies but provide lower power consumption at the expense of bandwidth and latency. [12]

2.2.6 SigFox

SigFox is a proprietary low power, low data rate wireless technology operating on the unlicensed Industrial, Scientific and Medical (ISM) radio bands. The network operates on a one-hop star topology network that requires an operator, usually SigFox, to setup the antennas. Due to this fact the number of messages that can be sent daily is limited, the coverage is conditioned by the infrastructure deployment by the operators and a fee is required to use the network.

SigFox networks are mostly uplink only. The data rates are between 10 bps and 1 kbps with ranges between 30 and 50 km in rural environments and 3 and 10 km in urban environments [8]. The very low power consumption makes is suitable for applications where small amounts of data need to be sent and long range is required.

2.2.7 6LoWPAN

6LoWPAN is an open standard defined by the Internet Engineering Task Force (IETF) to enable IPv6 for small embedded devices. "The concept was born from the idea that the Internet Protocol could and should be applied to even the smallest of devices." [18] The standard was developed to be used on top of existing **IEEE 802.15.4** based networks such as BLE and ZigBee enabling IPv6 addressing for such low-power devices.

The 6LoWPAN standard allows multiple low-powered devices to communicate seamlessly with the Internet, differing only in the header compression used in the 6LoWPAN which is optimized for the **IEEE 802.15.4** networks, allowing interoperability between them, apart from a gateway that does not need to translate packages.

Since 6LoWPAN is an IP it also allows for the leveraging of previous knowledge and tools used in IP networks.

2.2.8 Summary

Table 2.1 presents a summary of the different evaluated network protocols. The different characteristics are present for each, the 6LoWPAN protocol does not have fixed values since it is dependent on the physical layer on top of which it is used. It can be used on top of any IEEE 802.15.4 based network.

	Range	Data rate	Frequency	Power Consumption
Wi-Fi	50-100 m	2 - 1730 Mbps	2.4 or 5 GHz	High
LoRaWAN	2 -15 km	0.3 - 50 kbps	868/902/920 MHz	Low
BLE	100 m	125 - 2000 Mbps	2.4 GHz	Low
ZigBee	300 m	250 kbps	2.4 GHz	Low
Cellular	35 - 200 km	50 - 300 Mbps	900/1800/1900/2100 MHz	High
SigFox	3 - 50 km	10 - 1000 bps	868/902/920 MHz	Low
6LoWPAN	PHY dependent	PHY dependent	PHY dependent	PHY dependent

Table 2.1: IoT Network Protocols Summary

2.3 IoT Communication Protocols

On top of the Network Protocols presented in subsection 2.2, data needs to be transmitted between multiple devices that must communicate using the same *"language"*.

In the IoT, the communication can be divided in two types: device to device and device to Internet. The communication between the devices, whether edge nodes or gateways, needs to be coherent and have low overhead to ensure the limited resources are correctly used; between the gateways and the Internet this limitation usually does not exist, therefore more common protocols can be used.

Due to the recent growth in the IoT world a standard protocol still does not exist, nonetheless multiple contenders are available each with their own strengths and weaknesses. All the protocols were designed with certain set of applications in mind but all try to reduce the bandwidth used and the non payload bytes transmitted.

The following sections will attempt to present some of the more common ones along with the advantages and disadvantages of each one.

2.3.1 HTTP/2

Hypertext Transfer Protocol (HTTP)/2 is a major revision of the HTTP that tries to address some of the issues present in the HTTP/1.1 while ensuring backwards compatibility with the previous revisions. It is based partially in the experimental "SPDY" protocol developed by Google.

The goals of the protocol was to ensure the core features of the HTTP/1.1 while improving its efficiency, allow for multiplexing of requests via streams, add flow control and server to client push. [24]

HTTP is built on top of TCP and can make use of other security features such as SSL/TLS .

Although some IoT applications might use HTTP, the overhead imposed by the protocol headers makes it unusable for constraint devices.

HTTP requests and responses usually follow a REpresentational State Transfer (REST) model, to guarantee a certain degree of uniformity between the different applications, where each method (POST, PUT, GET) represents a certain action on the server.

2.3.2 MQTT

Message Queue Telemetry Transport (MQTT) is a communication protocol, initially developed by IBM and currently maintained by Organization for the Advancement of Structured Information Standards (OASIS) that works over Transmission Control Protocol (TCP). The protocol works with a publisher/subscriber logic and was projected with reliability and low-power in mind. [25]

The use of a publish/subscribe model has several implications in the network structure. A broker is required to relay the messages between the publishers and the subscribers, however the publishers do not need to be aware of the subscribers and the performance of the publisher is independent of the number of subscribers.

The protocol also allows for very small message headers making it good for low-bandwidth networks. The payload does not have a standard format, therefore the application layer needs to ensure this is coherent between all the nodes. The protocol has Quality of Service (QoS) measures built-in, supporting 3 levels: at most once, at least once and exactly once, determining how the message is sent to the broker. The publisher never knows if the message is delivered to any subscriber.

The MQTT protocol has some limitations since it does not support message queueing nor time to live, the broker just stores one message and delivers it to the subscribers that are awake. It is not possible to tell if a device that is

in low-power mode will ever receive the message.

2.3.3 CoAP

Constrained Application Protocol (CoAP) was created by the IETF and is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation. [26]

It is based on the REST model like many HTTP applications, where servers make resources available under a URL that can be accessed via methods such as POST, PUT, GET, etc. It has the advantage of allowing skill transfer from HTTP. It supports asynchronous message exchange, low header overhead and parsing complexity and security capabilities using Datagram Transport Layer Security (DTLS). [27] The data payloads can use multiple encodings such as JSON, CBOR, XML, etc. However due to the model used it does not support one to many communication as other protocols.

2.3.4 AMQP

Advanced Message Queueing Protocol (AMQP) is a protocol standard for message-queueing communications, maintained by OASIS. It works on top of TCP or User Datagram Protocol (UDP) and is focused on low-power devices supporting one-to-one and one-to-many communication.

It intends to define a low-barrier of entry protocol, with built-in safety and reliability features, guaranteeing interoperability of implementations and stability of operations.[26]

The devices communicate with a broker that supports message queuing and

ensures that the messages are correctly delivered to the destination. Unlike MQTT it is not a pure publisher/subscriber communication although it can work as one. Compared to MQTT it has a bigger overhead, due to the increase of features it supports.[29]

2.3.5 WebSockets

WebSockets is a communication protocol different from, but compatible with, HTTP that operates on top of the TCP layer. It is defined in the RFC 6455 [31] as "a two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code." To allow two-way communication without requiring multiple HTTP connections to be made, reducing the overall overhead.

Although some applications using websockets for the IoT exist, this protocol was designed with web browsers in mind, hence the concerns with overhead reduction and low-power were not taken into account, despite it using less bandwidth than HTTP.

2.3.6 XMPP

Extensible Messaging and Presence Protocol (XMPP) is a set of open technologies for instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data. [33]

It was originally developed by the Jabber open-source community to provide an open and decentralized alternative to closed communication protocols.

The standard for XMPP was defined by the IETF and it is a proven standard with security features (based on TLS) and flexible applications.

It is useful in IoT contexts since it already provides a tested protocol that is extensible and scalable with support for one-to-one and one-to-many communication patterns that can be used depending on the application. [34]

2.3.7 DDS

Data Distribution Service (DDS) is a middleware standard from the Object Management Group (OMG) that provides a broker-less, fully distributed communication platform. It intends to provide an interoperable, low-latency, reliable and scalable data transmission protocol.[35]

DDS provides a data centric layer to communicate from device-to-device or device-to-server, with multiple QoS and priority definitions per topic and device.[36] Although the communication is broker-less, a transparent broker can be used to relay messages.

DDS uses a data-centric architecture meaning that all the messages include the contextual information needed to be interpreted, meaning that the applications are aware of data but not necessarily of its encoding. [35]

2.3.8 Summary

Table 2.2 presents a summary of the different evaluated communication protocols. The different network protocols are qualitatively analysed from the perspective of an IoT application considering the constraints and requirements of such applications.

	Overhead	Topology	Designed for IoT
HTTP/2	High	One-to-One	No
MQTT	Low	One-to-Many	Yes
CoAP	Low	One-to-One	Yes
AMQP	Medium	One-to-Many	Yes
WebSockets	High	One-to-One	No
XMPP	Medium	One-to-Many	No
DDS	Medium	One-to-Many	Yes

Table 2.2: IoT Communication Protocols Summary

2.4 Security

One of the most important aspects to consider when developing or analysing IoT systems is security. The need to ensure that the system cannot be compromised nor leveraged with malicious intents.

According to ABI Research, it is expected that the IoT market will grow up to 64 million devices in 2021 [45]. This means that the attack surface available will increase largely in the coming years, therefore it is important to ensure that security considerations are taken into account in IoT systems. One of the main issues is that malicious attackers could infiltrate the edge nodes and that they could leverage them to infiltrate the main network or perform attacks on other networks, hence nodes are one of the most important defence layers since they can limit the reach of attackers.

Some network protocols already have support for security measures either built-in or added on top of the protocol. For instance, SSL/TLS can be added on top of TCP/IP to add a security layer, while BLE already has

built-in security measures.

Gateways could be used to compartmentalize networks and reduce the types and number of message that the nodes can send to other networks and even signal and ban nodes that are not behaving as expected. This ensures a greater level of security for both the nodes and the network.

2.5 Routing and network redundancy

Several protocols exist in networks to ensure both redundancy and to ensure that packets are correctly delivered. Not all network protocols require routing nor redundancy, some protocols are bus and broadcast based and therefore all the packets are delivered to all nodes. However this solutions are not always feasible mainly in large networks.

Most routing protocols are built for internet like solutions such as Open Shortest Path First (OSPF) [37], Routing Information Protocol (RIP) [38], Intermediate System to Intermediate System (IS-IS) [39] and Enhanced Interior Gateway Routing Protocol (EIGRP) [40]. These protocols also put solutions in place to deal with redundancy, however this is not always possible since it relies on the network to have more than one route between two nodes.
Newer protocols such as Routing Protocol for Low Power and Lossy Networks (RPL), Optimized Link State Routing Protocol (OLSR), Dynamic Source Routing (DSR) were created to address the requirements of redundancy and routing in lossy networks, which is usually the case in IoT scenarios.

Routing protocols can be either hierarchical or flat based, both with their advantages and disadvantages. Hierarchical based protocols use a topology

similar to the one used in the internet where nodes are divided in subgroups with one node responsible to relay to information to the wider network. This reduces the complexity in each sub network but requires all the information to pass through one node, leading to a possible single point of failure if no explicit redundancy measures are put into place. Flat protocols consider all nodes as equals and work in a more flooding methodology considering all nodes as possible receivers and broadcasting the information with a certain time to live until it reaches its destination.

2.5.1 OLSR

OLSR ([41], [42]) is a proactive routing protocol that targets mobile wireless LANs. In OLSR, nodes select a multipoint relay (MPR) among its' one hop nodes such as it can reach all of its 2 hop neighbours via a MPR, all of this information is stored in a neighbour table. The topology table is constructed by messages broadcasted by nodes with information regarding the nodes that have selected the sender as a MPR. Each of the nodes maintains a routing table with information from the neighbour table and the topology table. This information allows each node to know the routes to each other node. Being a proactive protocol it allows the routing information to be available at every instant for every node, however it has a greater overhead since the network is periodically flooded with messages with network information updates.

2.5.2 DSR

DSR [43] is a on-demand routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR allows for a fully self-organizing and self-configuring network without the need for infrastructure. It operates mainly via two mechanisms: route discovery and route

maintenance, both completely on-demand. Route discovery is used when a node wants to send a message to another node and does not yet know the route to the destination. Route maintenance is used when a node wants to ensure the route to another node still exists. During route discovery the nodes might discover other routes besides the one it requires which can be used in the future. Being on demand, DSR imposes a smaller overhead on the network, however the routing information might not always be available when required. The protocol was designed to work in highly mobile networks with up to 200 nodes but no more than 5 to 10 hops between nodes.

2.5.3 RPL

RPL [44] is a hierarchical routing protocol built on top of Internet Protocol version 6 (IPv6) supporting point-to-point, point-to-multipoint and multipoint-to-point communication between nodes. RPL constructs Destination Oriented Directed Acyclic Graph (DODAG) where the communication flow towards the central node. DODAGs are constructed according to network defined metrics trying to determine the best path to the destination. When constructing the network each node will select a parent node in a tree like fashion up to the head of the network which is parentless. The nature of RPL allows it to repair networks, detect and avoid loops and maintain smaller networks if the connection to the main DAG is lost. It supports multiple DAGs within the same DODAG. The protocol also allows for rule creation when constructing the network (e.g. battery operated nodes can advertise a less reliable connection to avoid being chosen as a parent node)

2.6 IoT Vendors

Currently a number of vendors of IoT devices exist, both with industrial and commercial applications. Although the full details of the implementations are not known due to trade secrets some information can be extracted from the functional descriptions. This section presents some of the existing IoT device vendors.

2.6.1 Rigado

Rigado provides IoT solutions for building management, smart retail and asset tracking by offering solutions with modules and gateways. [47]

The Rigado gateway has BLE, Wi-Fi, IEEE 802.15.4, Ethernet and Cellular. The gateway has a ARM Cortex-M7 that runs Linux allowing for the creation of custom applications using standard programming languages and interfaces.

Figure 2.6: Rigado Gateway [48].

Rigado also develops wireless modules that can be integrated in custom hardware to accelerate time to market. The modules interface via BLE and IEEE 802.15.4 with the gateway which then relays all of the information to a central server.

2.6.2 Libelium

Libelium is a IoT device company that focus on creating nodes and gateways with support for 120 different types of sensors with 10 node models. [49] The nodes can be deployed in different scenarios from cities, agriculture to security and water sensing.

The devices support Wi-Fi, Lorawan, 4G, SigFox, IEEE 802.15.4 and 868 and 900 MHz wireless communications and ModBus, CANBus, RS-232 and RS-485 wired communication. The devices can integrate with any cloud platform.

Figure 2.7: Libelium Node [49].

The gateways in the Libelium ecosystem support BLE, 4G and Ethernet and allow for the use of lower consumption network protocols and pass the storage and server communication to the gateway instead of the nodes. The gateways also support device tracking via BLE and Wi-Fi MAC address identification and distance calculation.

2.6.3 Nest

Nest is a company owned by Google that focus on consumer market IoT solutions such as cameras, thermostats, security systems and doorbells. Nest products integrate with the Google ecosystem and communicate via Wi-Fi, IEEE 802.15.4 and BLE using Thread, a protocol stack based on 6LoWPAN. [51]

The solution was designed with a focus on usability and integration with the other Google and Nest ecosystem products. The devices can communicate directly with a central server or via a gateway that can be one of Nest's or Google's product.

Figure 2.8: Nest thermostat [51].

The devices aim to control and sense the home environment allowing to "learn" the user habits and control lighting, temperature, CO_2 levels and movement generating alarms when required.

The Nest ecosystem also sets up an interface so that other vendors can integrate with it and use the Nest gateways. Systems like the Phillips Hue lightbulbs and the TP-Link Smart Plugs integrate with Nest [52]

This page intentionally left blank.

Chapter 3

Bluetooth Low Energy Technology

This chapter will explain the functioning of BLE describing all the layers and different blocks that are used according to the specification.

BLE is a proprietary wireless low-energy personal area network technology developed by the Bluetooth Special Interest Group (SIG). It was a development posterior to the Bluetooth Classic standard that aimed at reducing the power consumption while maintaining a similar communication range.

Similar to the specification for Bluetooth Classic, BLE defines multiple profiles with different purposes that manufacturers are expected to comply with to ensure compatibility between the different implementations. All the specifications are overseen by the Bluetooth SIG. [56].

The Bluetooth Core Specification is currently in version 5 [57], it defines both the Classic and Low Energy implementations considering all the required building blocks from the Physical Layer (PHY) to the Profiles that can be used for communication.

The BLE standard defines 2 layers, controller and host and also establishes a clear separation between what is implemented and what are the responsibilities of each layer. The following sections will detail the most important components of each and how they interact.

The BLE layer structure is presented in figure 3.1, the application layer is not the subject of this work but is the layer that defines the logic to be implemented in specific applications and that interacts with the host layer via a defined Application Programming Interface (API).

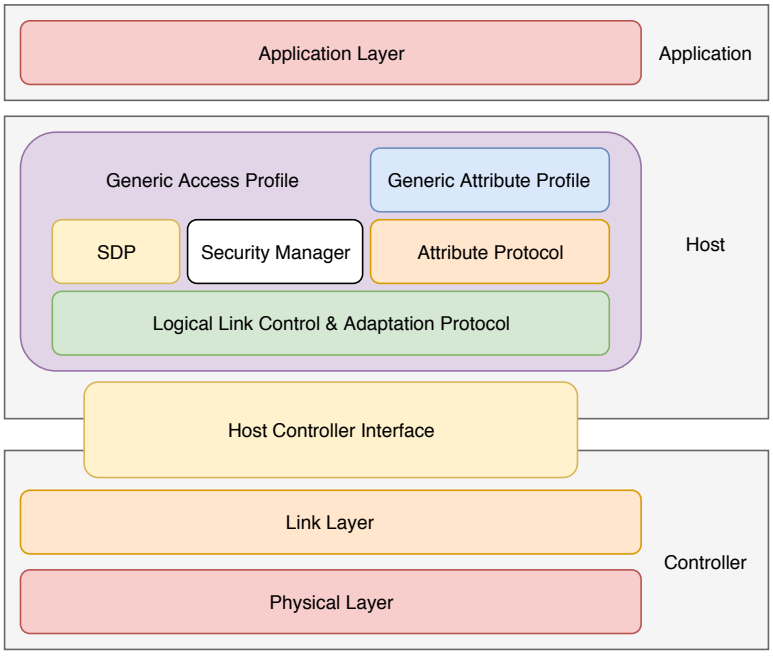


Figure 3.1: BLE Layers.

3.1 HCI

The Host Controller Interface (HCI) layer, as shown in figure 3.1, sits between the host and the controller layers. This block defines how the two layers interact and transmits the messages sent between the two.

The Bluetooth core system consists of a Host and one or more Controllers. A Host is a logical entity defined as all of the layers below the non-core profiles and above the HCI. A Controller is a logical entity defined as all of the layers below the HCI.

Both the controller and the host implement the respective parts of the HCI.

The host and the controller might be implemented in the same or in different hardware platforms that communicate using the defined HCI commands. These commands are defined in the Bluetooth Specification[57]. The commands can be transmitted over any physical bus but are usually transmitted over Universal Serial Bus (USB), Universal Asynchronous Receiver/Transmitter (UART) or Secure Digital Input Output (SDIO).

Over UART, the Bluetooth specification defines 3 types of HCI packets used in BLE communications:

- Command packets
  - Sent by the Host to control the Controller by setting configuration parameters, starting advertisements, scannings and connections, etc.
- Event packets
  - Sent by the controller to inform the Host of occurrences, for instance, scanning results, connection requests, connection completion, etc.
- Asynchronous Connection-Less (ACL) Data packets
  - Used by both the Host and the Controller to exchange data in a asynchronous manner, this data is usually forwarded to the device connected via BLE.

The specification also defines the settings for the connection. It should have 8 data bits, no parity and 1 stop bit. The use of flow control is permitted but optional.

3.2 Controller layer

The controller layer implements and defines the lower layers of BLE: the PHY layer and the link layer.

3.2.1 PHY

The PHY contains all the required implementation to ensure that the correct information is sent over the air.

BLE operates in the unlicensed 2.4 GHz ISM band and employs two access schemes Frequency Division Modulation Access (FDMA) and Time Division Multiple Access (TDMA).

FDMA consists of dividing the frequency space in smaller bands with different purposes. This is achieved by operating in 40 channels, each separated by 2 MHz, along the allocated frequency range. Within these channels, 3 are used as primary advertising channels and 37 are used as secondary advertising channels and as data channels.

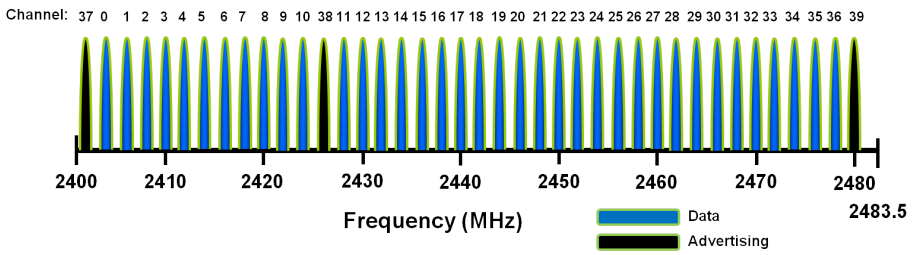


Figure 3.2: BLE channels [59].

TDMA consists in the allocation of time slots for transmission inside each of the channels. A device transmits data at a predetermined interval and another responds after another interval. Therefore each channel is divided into different time slots, each time slot is referred as an event. This also allows for multiple connections to the same device, each communicating in its own time slot.

The PHY layer defines four types of events [57]:

- Advertising
  - Standard advertising event compatible with the older Bluetooth version
- Extended Advertising
  - Advertising event that allows for more data to be transmitted and with increased features
- Periodic Advertising
  - Event used for broadcasting packets between two devices at a fixed interval to communicate without a connection, must be initiated by an extended advertisement event.
- Connection
  - Event that occurs when two devices establish a connection to exchange data

According to the specification not all devices are required to have a transmitter and a receiver and can have only one of those depending on the supported capabilities.

3.2.2 Link layer

The link-layer sits on top of the PHY layer and defines the state of the BLE devices.

Devices can be in one of 5 link layer states: standby, advertising, scanning, initating, and connecting as can be seen in figure 3.3.

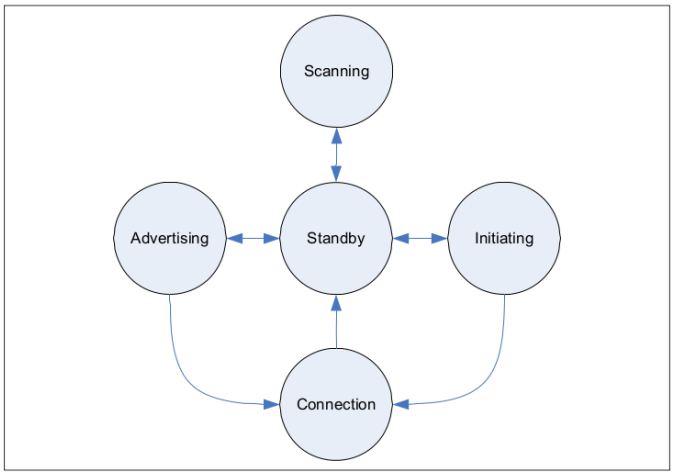


Figure 3.3: Bluetooth device state diagram [57].

When in standby, the device is idling and taking no action. In scanning, it is operating in the BLE advertising channels looking for devices that are advertising. Before moving to the connected state, one of the devices must initiate the connection.

These 5 states are related to the link layer roles: scanner, advertiser, master and slave. The scanner and advertiser are the roles the devices take when scanning and advertising respectively. The master and slave roles are roles taken after 2 devices are connected. The device that initiated the connection becomes a master and the other device becomes the slave, forming a piconet[1].

[1] multiple devices sharing the same physical channel and with a synchronized clock and hopping frequency

The BLE standard poses no restriction on what might be supported by each device nor the roles that can be taken at any given moment. If the link layer supports it, a device can be a slave, master, scanner and advertiser simultaneously.

Frequency Hopping

Devices inside a piconet use frequency hopping to periodically change the communication frequency channel to avoid interference. This achieved by using a pseudo-random ordering of the 37 data channels determined by the initiator device. The pattern can be adapted to exclude portions of the frequency that are used by interfering devices.

The adaptive hopping technique improves Bluetooth co-existence with static (non-hopping) ISM systems when these are co-located and have access to information about the local radio environment, or detected by other means. [57]

Figure 3.4 depicts a diagram with 3 active BLE links with the frequency hopping sequence, each of them in a different color. Each of the links has a a different hopping sequence in order to more easily avoid collisions. Link 3, depicted in orange, has the hopping sequence outlined for clarity.

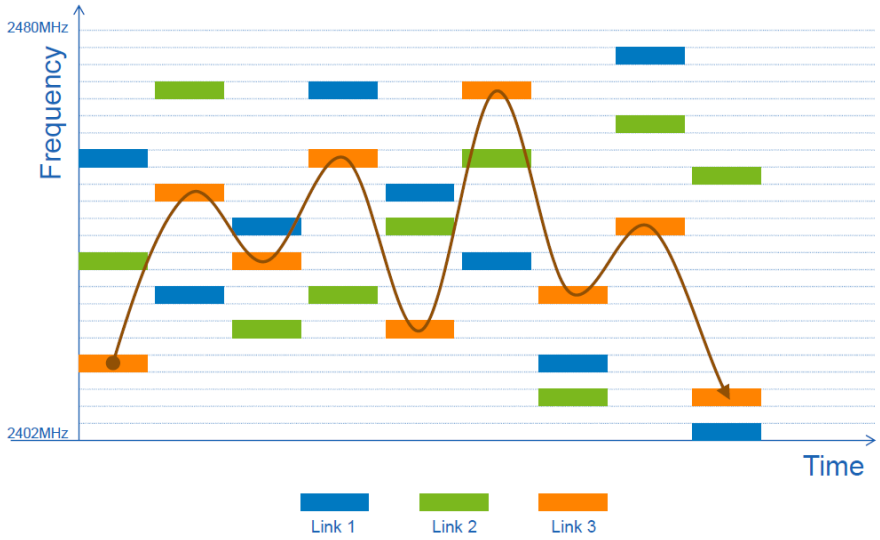


Figure 3.4: Bluetooth frequency hopping [59].

Topologies

Devices inside a piconet have restrictions in the communication with other devices. Slaves cannot communicate with slaves but devices can be slave and master simultaneously. The standard also states that a master can connect to multiple slaves and that a slave can connect to multiple masters.

Figure 3.5 represents the different possible BLE topologies. In this figure stars represent devices advertising, circles represent devices scanning and arrows point from master to slave. In BLE, each slave communicates with a master using a different channel therefore in the piconet composed by devices A, B and C there are two channels (A-B and A-C), device D is advertising and device A can connect to it to include it in its piconet.

The piconet composed by devices K, L and M is similar to the previous piconet but device K is a slave to M and a master to L. Device K advertises and N can initiate a connection with it becoming its master.

Piconet H is purely a advertising piconet and in this case there is only one physical channel, the advertising one, with device H as a broadcaster and devices I and J as scanners.

The last piconet, composed by devices O, P, Q and R is composed by variations of the connection types present in the other piconets.

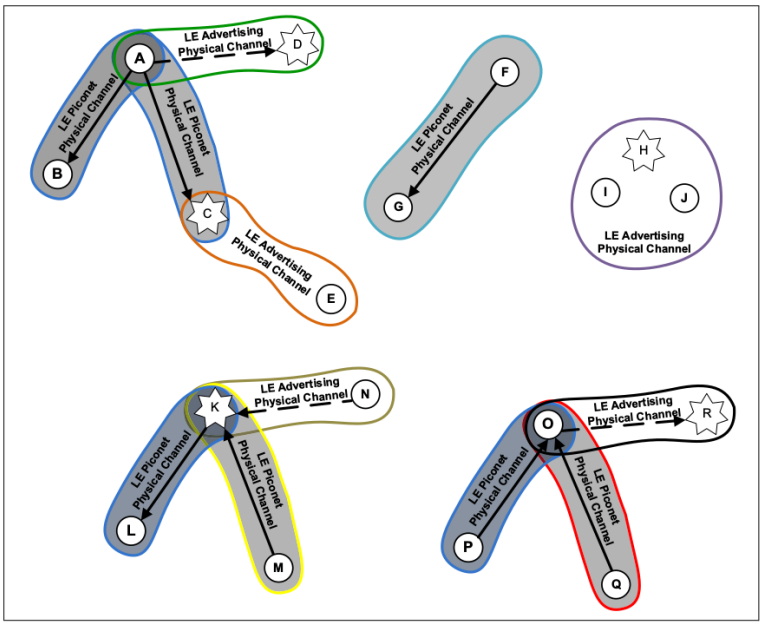


Figure 3.5: BLE topology [57].

Packets

The link layer defines one type of packets for all the messages sent over the air, whether when advertising, connecting or sending data. These BLE packets contain 4 fields the preamble, the access address, the Protocol Data Unit (PDU) and the Cyclic Redundancy Check (CRC) as shown in figure 3.6.

Figure 3.6: BLE packet structure.

The base packet will then include inside the PDU the specific payloads for the two types of channel transmissions, advertising channel and data channel. The base for both the advertising and the data channel packets is quite similar.

Advertising channel packets include a header that indicates the type of the packet and a payload that depends on the type of the advertising (*cf.* figure 3.7).

Figure 3.7: BLE advertising channel packet structure.

The BLE specification [57] defines 7 advertising channel packets, each with a different purpose:

- Connectable Undirected Advertising (ADV_IND)
- Connectable Directed Advertising (ADV_DIRECT_IND)
- Non-Connectable Undirected Advertising (ADV_NONCONN_IND)
- Scannable Undirected Advertising (ADV_SCAN_IND)
- Scan Request (SCAN_REQ)
- Scan Response (SCAN_RSP)

- Connection Request (CONNECT_REQ)

The Connection Request packet which is sent by a master device to a slave device when initiating a connection includes all the parameters that will be used throughout the connection, for instance the connection interval (time between 2 connections for data transmission), the hop increment (how the frequency hopping should be performed) and the transmission window (how much time each connection event lasts).

Data channel packets are sent in data channels and are used for the master and the slave to exchange information between them. The format is presented in figure 3.8, this includes the header that indicates the type of message being sent, the payload, either data or control, and the Message Integrity Check (MIC) which is only present when the connection is encrypted.

Figure 3.8: BLE data channel packet structure.

The data channel payload has 2 types; link-layer data or link-layer control. The first is used to send Logical Link Control and Adaptation Protocol (L2CAP) data while the second is used to control the connection and change parameters within the channel. The Attribute Protocol (ATT) data is explained in further detail in section 3.3.4.

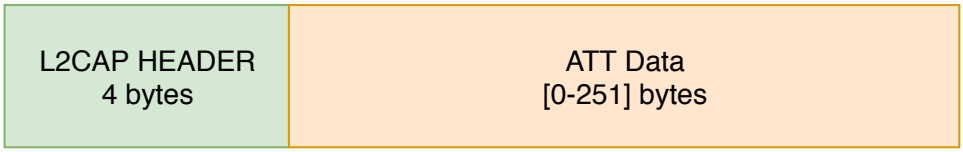


Figure 3.9: BLE data channel payload structure.

Connection Events

The data channel packets can only be exchanged in connection events that occur at intervals defined by the master in the Connection Request. The Connection Interval can be defined between 7.5 ms and 4 s in 1.25 ms intervals.

Connection events are active as long as either the Slave or the Master are transmitting data. However the master must terminate the connection event at least 150 µs, which is the Inter Frame Space, before the beginning of the next connection event. Figure 3.10 shows multiple connection events, with just a single or multiple data transmissions.

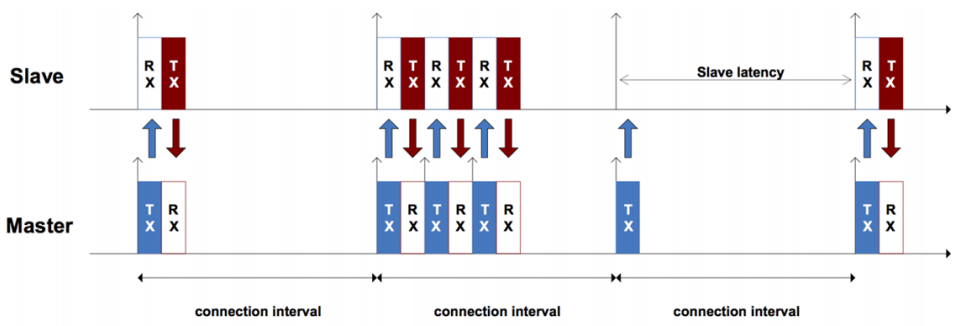


Figure 3.10: BLE connection events. [61]

Connection events occur even if no data needs to be transmitted between the master and the slave. This is performed to ensure that the link is still active.

If some connections occur without any response from the slave, the master can terminate the connection. This is determined using the Supervision Timeout. If a device does not receive any Data Packet PDU before the Supervision Timeout has elapsed since the last received one, the connection is considered lost.

The Supervision Timeout is calculated according to equation 3.1 [57], where *connSlaveLatency* is the maximum number of connection events a slave can skip and *connInterval* is the Connection Interval.

$$SupervisionTimeout = (1 + connSlaveLatency) \times connInterval \times 2 \quad (3.1)$$

3.3 Host Layer

The Host Layer implements several blocks responsible for communicating the Controller Layer and with the application. These blocks manage some higher level aspects of the BLE protocol such as security management and services provided.

3.3.1 L2CAP

L2CAP is the layer between the higher Generic Access Profile (GAP), Generic Attribute Profile (GATT) and application host layers and the lower layers in the controller. The L2CAP layer is responsible for protocol multiplexing and data segmentation and reassembly as shown in figure 3.11.

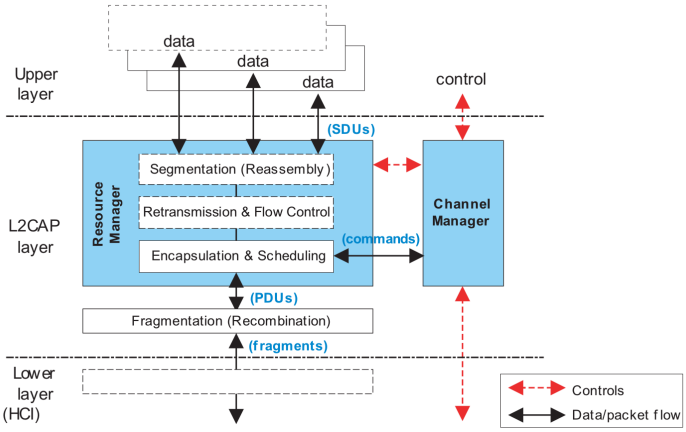


Figure 3.11: L2CAP structure. [57]

As presented in figure 3.11, the L2CAP layer has 2 main blocks: a Resource Manager and a Channel Manager. The Channel Manager is responsible for handling the creation, management and closing of L2CAP channels with commands sent from the upper layers, lower layers or the Resource Manager. The Resource Manager handles scheduling, fragmentation, retransmission and flow control, sending Service Data Unit (SDU) to the upper layers and PDU to the lower layers.

All the L2CAP packet fields are little-endian except the information payload field. The endianness of the the protocols encapsulated within the L2CAP information payload is protocol specific.

Channel Types

Each connection between 2 devices creates a channel. Each of these channels has a Channel Identifier (CID) that can be dynamically allocated or fixed depending on the type of channel. Three types of channels exist:

- Connection-oriented
  Used to connect two devices and transmit data between the two (ensures reliable data transmission)

- Connectionless data
  Used for broadcast or unicast data without considering the setup of a actual channel (unreliable data transmission). It is not supported in BLE

- L2CAP signalling
  Used for channel configuration and signalling (disconnection, echo requests, configuration, flow control credits, etc)

Fragmentation and Recombination

The L2CAP layer is responsible for fragmentation and segmentation of messages. Upon establishing the connection each device specifies the supported Maximum Transmission Unit (MTU) and Maximum Payload Size (MPS). For the data transmitted over the channel the lower values will be used.

The MTU corresponds to the maximum size of a SDU, the maximum packet that the upper layers can send and receive to the L2CAP layer, e.g. the maximum ATT payload size. The MPS corresponds to the maximum payload that the L2CAP layer can accept from the lower layers and therefore to the size of the packets sent over the air.

Therefore the L2CAP connection with a MTU of 100 and a MPS of 27 can send data with a maximum of 100 bytes broken over 27 bytes packets; which would require 5 packets.

The messages that are fragmented and provided to the link-layer in one device need to be received and handled in the peer device to be recombined to form an SDU before being sent to the upper layers.

If a message is received with a SDU size greater than the maximum MTU the connection will be terminated, since even if the message is properly reassembled it could not be correctly provided to the upper layers. If any packet is greater than the MPS the connection shall also be terminated.

Modes of Operation

L2CAP specifies 6 different modes of operation [57]

- Basic L2CAP Mode

- Flow Control Mode
- Retransmission Mode
- Enhanced Retransmission Mode
- Streaming Mode
- Low Energy (LE) Credit Based Flow Control Mode

Each of these modes operate in a different way, Basic Mode is equivalent to the Bluetooth v1.1 mode. In flow control, retransmission and enhanced retransmission all the PDUs are numbered and acknowledged; flow control detects that packets were lost but does not resend them, the retransmission modes retry to send the data. Streaming mode is used for real-time isochronous traffic, data that requires to be received at a specific time and with a specific order. LE credit based flow control mode is the only mode that shall be used for LE L2CAP connection oriented channel [57].

LE Credit Based Flow Control

Connection oriented channels using LE credit based flow control include some extra information in the PDU header sent (*cf.* figure 3.12).

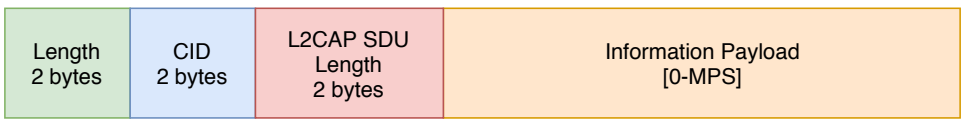


Figure 3.12: L2CAP connection oriented channel packet.

As discussed in section 3.3.1, the packets must be divided to ensure that all the information can be properly sent over the air.

When using the credit based flow control method each of the devices in the connection provides to the other a certain number of credits, upon connection, besides the information of the MPS and MTU. When a packet is sent a credit is used. Devices can only send messages while they have credits available. This ensures that the receiving device always can receive and store the messages sent. The BLE devices need to provide more credits to the device they are connected to ensure that the communication can continue, the credits are sent using signalling messages.

In example in a connection between two devices, A and B, with the following characteristics:

	MPS	**MTU**	**Credits to provide**
Device A	30	250	10
Device B	40	100	5

The connection will use a MPS of 30 and a MTU of 100. Device A will provide 10 credits to Device B while receiving 5 credits from it.

Device A wants to send 80 bytes and devices B will respond with a 60 byte payload. Figure 3.13 shows the messages sent from A to B, in which A spent 4 credits having only 1 left. Figure 3.14 shows the messages sent from B to A, in which B spent 3 credits having 7 left.

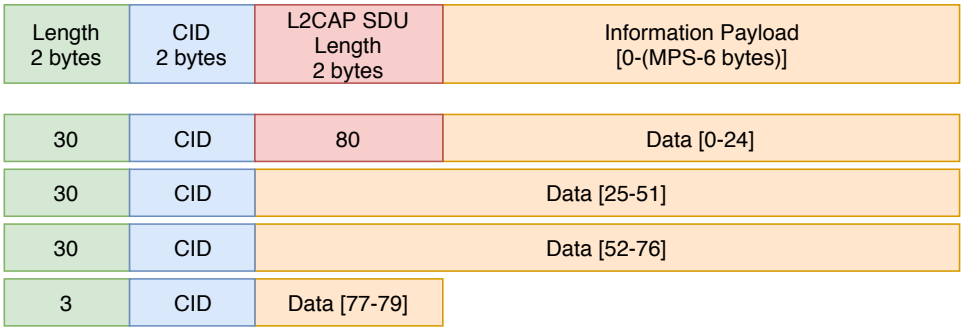


Figure 3.13: Data transmission from A to B.

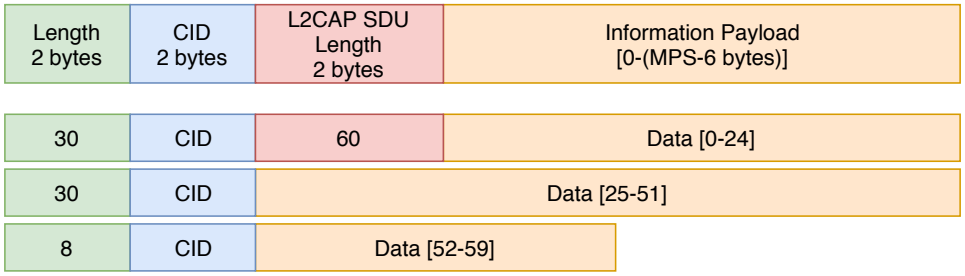


Figure 3.14: Data transmission from B to A.

After this transmission device B would have to provide A with more credits otherwise A could only send 1 packet.

3.3.2 GAP

The GAP represents the base functionality common to all Bluetooth devices such as modes and access procedures used by the transports, protocols and application profiles. GAP services include device discovery, connection modes, security, authentication, association models and service discovery.

It defines if the device is visible or not and how all the devices can interact with it.

The GAP defines 4 possible roles that will interact and influence the ones defined in the link layer. The roles are peripheral, central, observer and broadcaster.

The observer and broadcaster pair are related to the scanner and advertiser roles in the link layer. A device that is defined as a broadcaster advertises periodically sending packets with data and other devices cannot connect to it. The observer role scans for advertising data only and does not try to connect to any devices. These two roles form the basis of the mesh support for Bluetooth that was defined in version 5.0 of the protocol [57].

The peripheral and central pair are related to the master and slave link layer roles. A device with a GAP central role will scan for advertising devices and initiate a connection with a device, this device has a master role in the link layer. When a device is defined by the GAP layer as a peripheral it will send periodic connectable advertise messages, it is configured as a advertiser in the link layer, upon receiving a connection request it becomes a slave in the link layer and a connection is established.

3.3.3 Security Manager

The security manager is responsible for handling the rules and algorithms used by the GAP for the security of a BLE connection. A connection operates at a given security mode, with each mode having several security levels. The security offered by the different modes and levels are presented in table 3.1.

Security Mode 1	
Level 1	No authentication No encryption
Level 2	Unauthenticated pairing with encryption
Level 3	Authenticated pairing with encryption
Level 4	Authenticated LE Secure Connections pairing with encryption

Security Mode 2	
Level 1	Unauthenticated pairing with data signing
Level 2	Authenticated pairing with data signing

Table 3.1: BLE security modes.

All connections start in Security Mode 1 Level 1, where no authentication nor encryption is used and the information is exchanged freely. Afterwards

both devices can agree on increasing the security mode and level of the connection via pairing and/or bonding. The different pairing mechanisms provide various levels of security, such as encryption and authentication. Security Mode 2 includes an extra layer of security by signing all the data to ensure it was not tampered with.

Figure 3.15 represents the process in which the pairing and the bonding occur. After establishing a connection either device can trigger the start of the pairing process, exchanging features that will be used to compute keys that are sent to the one another. If these keys are validated the connection is encrypted and the devices are paired. Afterwards, more information can be exchanged to ensure the bonding, which corresponds to storing the pairing information for future reuse, effectively trusting any device with that information.

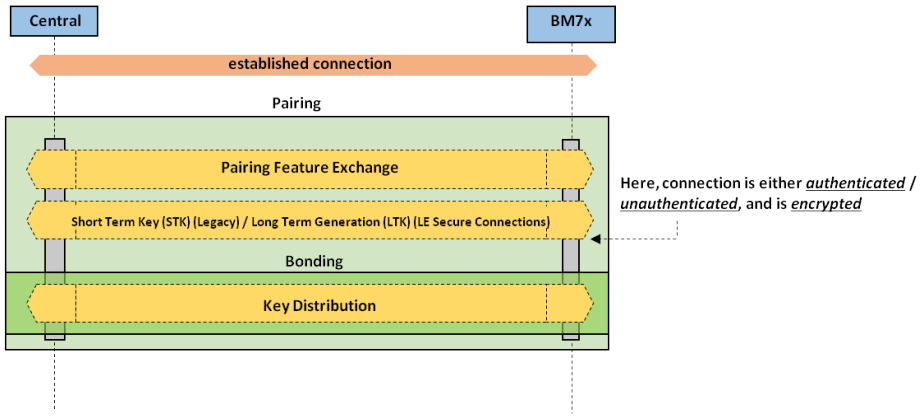


Figure 3.15: BLE pairing and bonding process [62].

3.3.4 ATT

The ATT is the communication protocol between a server and a client.

The ATT client sends commands, requests, and confirmations to the ATT server. The ATT server sends responses, notifications and indications to the

client. This is the means through which two devices can communicate. The client reads and writes in the server attributes which can be changed by the server itself. Each server can have more than one client connection and devices can be simultaneously a server and a client.

Usually devices with a GAP central role are clients and devices with a GAP peripheral role are servers but this is not required.

An attribute is a piece of labelled data used to exchange information. Attributes contain an handle, a type, a value and a set of permissions as show in figure 3.16.

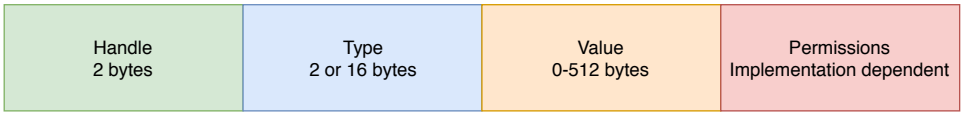


Figure 3.16: BLE attributes

The attribute can have a total length bigger than the maximum link layer PDU (*cf.* figure 3.9), this is possible because the L2CAP layer will handle the fragmentation and reconstruction of the packets to fit the maximum defined MPS (*cf.* 3.3.1).

3.3.5 GATT

The GATT describes the functionality of the attribute server and client. It describes the services, characteristics and attributes used in the server. It provides interfaces for discovering, reading, writing and indicating of service characteristics and attributes. GATT is used on BLE devices for profile service discovery.

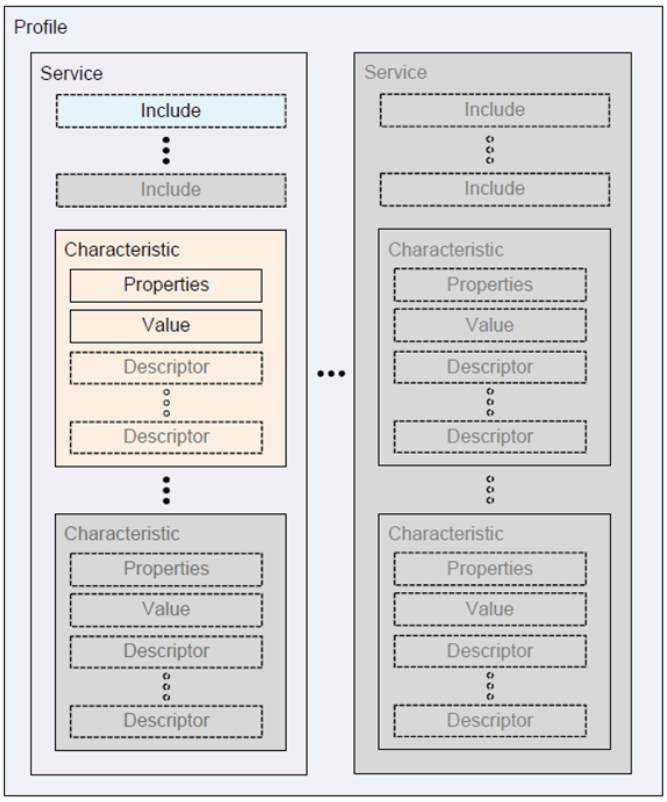


Figure 3.17: GATT Profile Hierarchy [57].

In figure 3.17, the hierarchy of each GATT profile is represented. Each profile has one of more services available and each service has one or more characteristic. BLE clients write and read characteristic values. Profiles are used to aggregate services. The specification of profiles not always exist if the profile only includes one service.

3.4 BLE Profiles and Services

The BLE specification defines multiple standard profiles and services that can be used depending on the application. The profile specification was explained in 3.3.5.

All of the existing profiles and services are listed in the Bluetooth SIG website GATT Specification page [64] with a specification for each one.

Some of the existing profiles are Human Interface Device Service (HIDS), used for interfaces such as keyboards and mouses, Phone Alert Service Profile (PASP) used in wearables with phone communicate for alerts, Current Time Service (CTS) for displaying time in BLE devices such as watches. Bluetooth devices can implement multiple profiles simultaneously.

In this work the service used will be the Internet Protocol Support Profile (IPSP) which will be explained in detail in the next section.

3.4.1 IPSP

The IPSP is a profile designed by the Internet Work Group (WG) of the Bluetooth SIG that defines what are the requirement of a Bluetooth device to communicate using IPv6 over BLE [65].

The specification defines 2 types of roles: Nodes and Routers. Each Bluetooth device can implement one or the other or both. The router can route IPv6 packets while the node can only produce or consume such packets.

Unlike Wi-Fi, the connection is performed from router to node. Nodes either advertise directly to the router, using directed advertisement packets or the router can connect automatically to nodes that in the undirected advertisement packet include the IPSP Universally Unique Identifier (UUID).

This profile does not make use of GATT and defines the Internet Protocol Support Services (IPSS) only for device and service discovery. GAP is used to discover device and setup the connection and manage the security modes. All the messages are transmitted using L2CAP LE Connection Oriented Channels with LE Credit Based Flow Control.

Figure 3.18 shows the different blocks that compose the IPSP stack. All the colored blocks were explained in this chapter while the grey blocks will be explained in detail in chapter 4

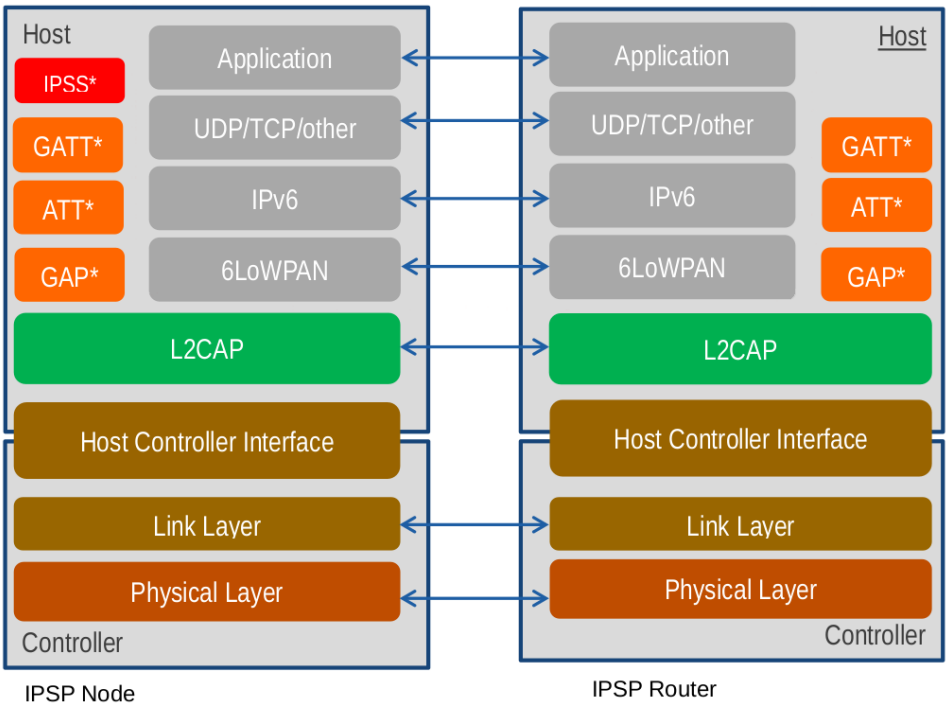


Figure 3.18: IPSP stack. Adapted from [65]

A minimum supported MTU of 1280 bytes is defined to ensure the profile works correctly.

Chapter 4

6LoWPAN Protocol

The current chapter will describe the 6LoWPAN protocol, its differences and advantages when compared to IPv6 and its usage on top of BLE.

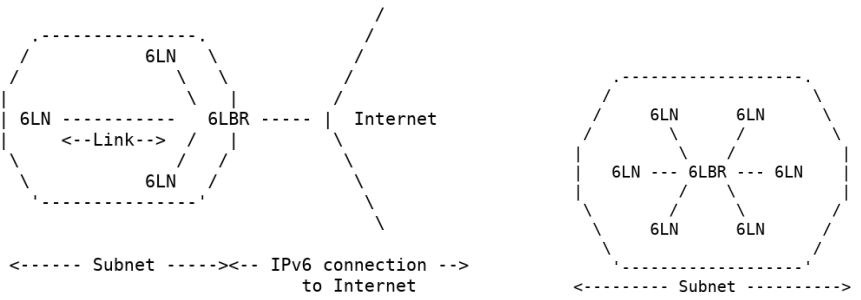
6LoWPAN is a standard defined by the IETF initially for the transmission of IPv6 packets over IEEE 802.15.4 networks in RFC 4944 [54] which was then updated by RFC 6282 [66], RFC 6775 [67], RFC 8025 [68] and RFC 8066 [69]. These documents define how the protocol should be used and compressed, how the neighbour discovery is performed and all the required information to use 6LoWPAN.

The implementation of 6LoWPAN over BLE is defined in RFC 7668 [75] which references the previous mentioned RFCs while presenting the differences between the use of BLE and IEEE 802.15.4 networks.

The use of an Internet Protocol is interesting for IoT networks since it simplifies the process of transmitting the information gathered by devices to central servers since the data only needs to be properly routed to reach the destination without further translation.

6LoWPAN networks can have two possible topologies, one where the network

is connected to the internet via a border router (*cf.* figure 4.1a) and another where the network is isolated from the internet but still operates normally (*cf.* figure 4.1b).



(a) Internet connected network (b) Isolated network

Figure 4.1: 6LoWPAN networks [75].

4.1 IPv6

Before entering into detail in the 6LoWPAN standard it is important to understand some of the most important aspects of IPv6 to determine why it was chosen as the basis for 6LoWPAN instead of Internet Protocol version 4 (IPv4).

IPv6 is a standard designed by the IETF and described in RFC 8200 [72]. It was first proposed in 1995 in RFC 1883 [70] and then updated in 1998 in RFC 2460. It is a new version of the IP designed to succeed the specification of IPv4.

The main differences between IPv4 and IPv6 are, according to RFC 8200 [72]:

- Expanded Addressing Capabilities - expanding the address size from 32 bits to 128 bits;
- Header Format Simplification - dropping or making optional some

header fields;

- Improved Support for Extensions and Options - changes to the header option encoding making forwarding more efficient;

- Flow Labelling Capabilities - allowing multiple packets to be treated as a single flow;

- Authentication and Privacy Capabilities - adding support for authentication, data integrity and confidentiality.

Considering the expected growth of the number of IoT devices, the increase in the address space from 4,294,967,296 devices in IPv4 (which is already saturated) to the 340,282,366,920,938,463,463,374,607,431,768,211,456 for IPv6 ensures that all the devices will be able to directly connect to the internet if required. The simplification of the header format is also useful since it reduces the overhead which is something of great importance for low energy devices.

Another important aspect of IPv6 that makes it suitable for the basis of the low power networks is neighbour discovery since it allows for on the fly discovery of the network and determining that neighbours are still reachable. Neighbour discovery does not make use of broadcast messages avoiding having to reach every device to find neighbours. [73]

IPv6 also allows for Stateless Address Autoconfiguration (SLAAC) [74], which is built on top of neighbour discovery and permits devices connecting to a network to auto assign IP addresses without the need for Dynamic Host Configuration Protocol (DHCP). This is useful for low energy devices that might not be capable of running a full DHCP capable stack.

4.2 Header compression

6LoWPAN RFC 6282 defines how the compression of IPv6 headers is performed for low power networks. This enables the reduction of the large IPv6 headers down to some bytes.

In IPv6 usually two headers are present: the IPv6 header and the IPv6 next header, the first corresponds to the essential information for data transmission while the next header contains extra non-essential-information for the data sent. RFC 6282 defines how to compress both types of headers, allowing to fully elide the next header.

The 6LoWPAN header includes the dispatch, the compressed IPv6 header and the inline IPv6 header fields (*cf.* figure 4.2)

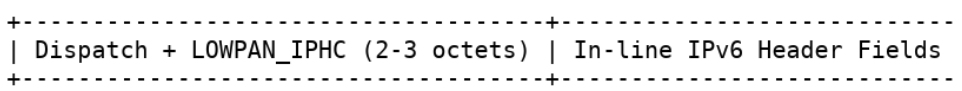


Figure 4.2: 6LoWPAN header [66]

4.2.1 IPv6 header compression

In the best scenario the IPv6 header can be compressed from 40 bytes down to 2 bytes with link-local communication; when routing over multiple hops it can be compressed down to 7 bytes.

The header has the format shown in figure 4.3, it includes Traffic Class Flow Label (TF), Next Header (NH), Hop Limit (HLIM), Context Identifier Extension (CID), Source Address Compression (SAC), Source Address Mode (SAM), Multicast Compression (M), Destination Address Compression (DAC) and Destination Address Mode (DAM). These bit fields will define what options will be carried inline and what options can be omit-

ted. Some of these options are briefly detailed in this section and further information about all of them can be consulted in RFC 6282 [66].

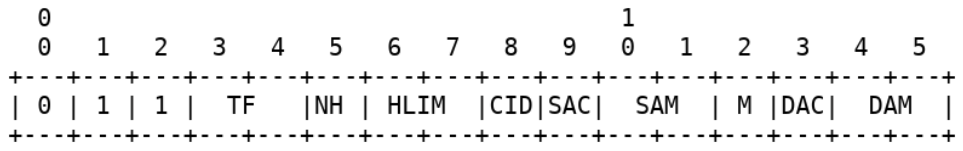


Figure 4.3: 6LoWPAN header encoding [66]

The NH field defines whether the Next Header information is compressed or not, this is further explained in section 4.2.2.
The HLIM field defines 3 options that can be determined from the compressed values (1, 64 or 255) or that the Hop Limit is present inline. This determines how many hops a packet can perform before being discarded.
Combined, the SAC and SAM determine what type of compression is used for the source address, if any, and can be combined with the CID to further compress it.
The DAC and DAM fields behave exactly as the SAC and SAM but to determine the compression applied to the destination address. The extra M field is used to determine if the address is multicast or not.

The CID field will enable Context Identifier Extension this will cause the Source Context Identifier (SCI) and Destination Context Identifier (DCI) fields to be carried after the DAM and before the IPv6 header fields. These two fields represent the prefixes used for the Source and Destination Addresses.

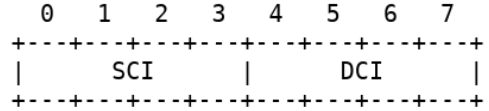


Figure 4.4: 6LoWPAN SCI and DCI encoding [66]

The fields following the 6LoWPAN header are the same as the ones in the IPv6 header apart from the the ones elided and some assumptions. The full IPv6 header is shown in figure 4.5, when using 6LoWPAN the version field is not present since version 6 is always assumed. The Traffic Class and Flow Label depend on the TF field value. The payload length is present and has the full value. The Next Header and Hop Limit are dependent on the NH and HLIM fields. The Source Address and the Destination Address, which in IPv6 always have 128 bits each, can be almost fully elided depending on the configuration of the SAC, SAM, M, DAC and DAM fields.

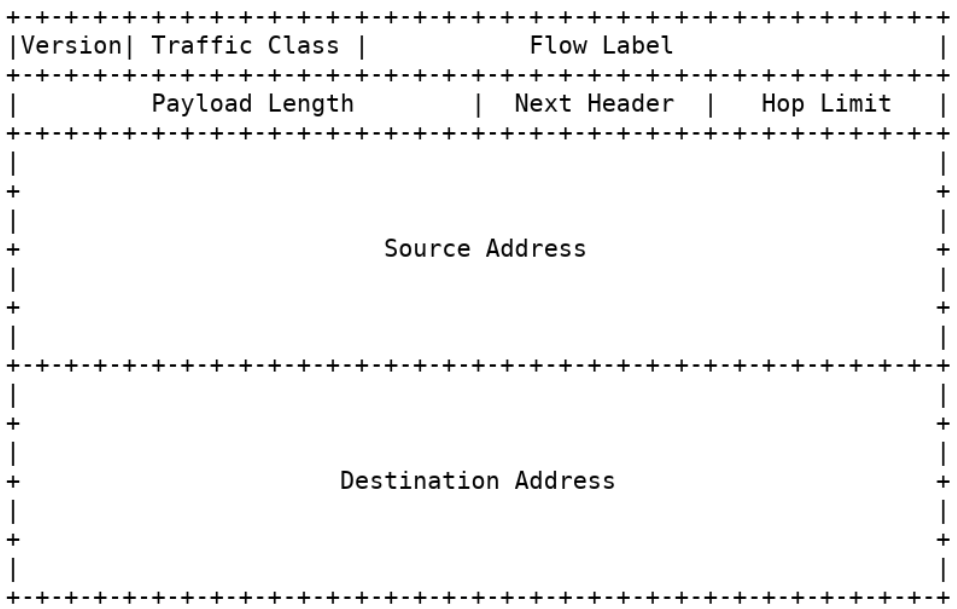


Figure 4.5: IPv6 header [72]

4.2.2 IPv6 next header compression

When the NH bit is set the IPv6 Next Header field is elided and the 6LoWPAN header is used. The configuration of both fields is then as presented in figure 4.6.

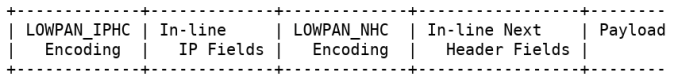


Figure 4.6: IPv6 head and next header configuration [66]

The Next Header Compression includes a variable length ID that identifies the type of compressed next header (TCP, UDP, Internet Control Message Protocol (ICMP), etc) and then the compressed fields of the protocol.

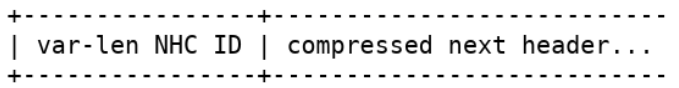


Figure 4.7: IPv6 next header compression [66]

RFC 6282 defines the compression for UDP messages and how to compress the ports the checksum and all the other required fields. Other proposals exist for the encoding of TCP traffic.

4.2.3 IPv6 extension header compression

Since the 6LoWPAN requires compressed next headers to be preceded by a IPv6 compressed header or another IPv6 compressed next header, the IPv6 Extension Header is compressed using the same method as the IPv6 next headers. As shown in figure 4.8 the IPv6 Extension Header Identifier (EID) is present and the NH is included to allow a next header to be present.

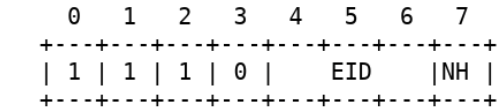


Figure 4.8: IPv6 extension header compression [66]

The IPv6 is used almost unmodified except for the Next header field that is

elided when the NH bit is set and the length field that is compressed and therefore limits the use of extensions that have no more than 255 bytes.

4.3 Neighbour Discovery

Neighbour discovery in 6LoWPAN is similar to the methods used in IPv6, however some applicability limitations of the IPv6 discovery need to be overcome due to the low-power nature of the 6LoWPAN systems.

The IPv6 neighbour discovery is defined in RFC 4861 [73] while the specification for 6LoWPAN is described in RFC 6775 [67].

The discovery process is performed using 4 main messages: Router Advertisement (RA), Router Solicitation (RS), Neighbour Advertisement (NA) and Neighbour Solicitation (NS) which are built on top of Internet Control Message Protocol version 6 (ICMPv6). These messages are sent either by routers or nodes.

All nodes send NS messages and reply with NA messages, this is performed since one of the changes from 6LoWPAN neighbour discovery is that all interactions are initiated by the host to allow support for sleeping nodes.

Nodes that are routers reply to RS messages sent by nodes to discover routers with RA messages. These messages include all the information required for network registration and SLAAC.

The neighbour discovery process is also used to determine the best route to a given host and therefore devices are required to store that information or construct it as needed.

4.4 6LoWPAN over BLE

The RFC 7668 defines the particular implementation of 6LoWPAN over BLE and includes some adaptations to ensure that the protocol can be properly used considering the BLE specifications. The standard considers that a Bluetooth version 4.1 or higher is required, this matches the support for the IPSP in the Bluetooth Specification.

The standard considers only the implementation using a BLE star topology however it considers that two peripherals can communicate via the central and poses no limitation to having peripherals that are central to other networks and can therefore create wider networks.

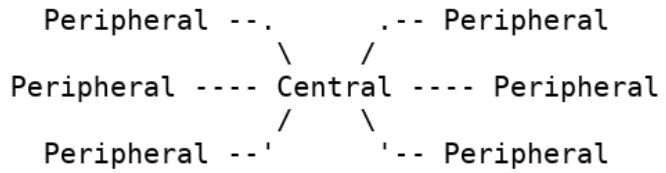


Figure 4.9: BLE star topology [75]

Since BLE already supports fragmentation in the L2CAP layer none of 6LoWPAN fragmentation features are required and for the application layers the packets are always assumed to be sent non fragmented.

The RFC also defines the protocol stack and how 6LoWPAN is included in it, as shown in figure 4.10.

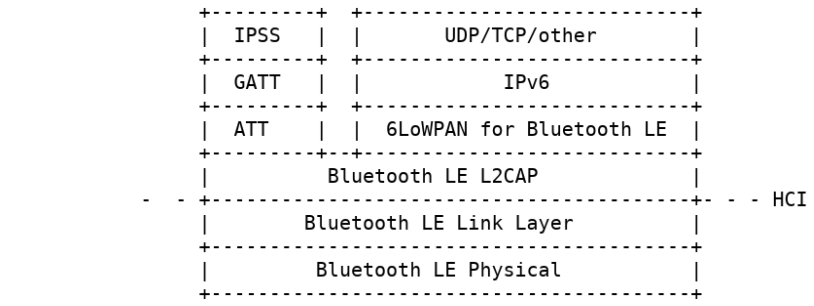


Figure 4.10: 6LoWPAN BLE stack [75]

4.4.1 6LoWPAN addresses

The addresses in 6LoWPAN are obtained as in IPv6. The only advantage is that they can be elided in multiple situations.

IPv6 defines multiple address types, of these the two most important ones are: link-local addresses and global addresses.

The link-local address exists only in the same link and always have a $fe80$ prefix, as defined in RFC 4291 [76], as can be seen in figure 4.11. This address can only be used for communication inside the link, which due to the structure of BLE links can only be used for communication between a central and a peripheral.

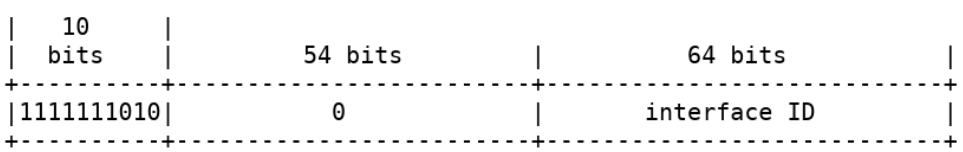


Figure 4.11: IPv6 link-local address structure [76]

The global address includes the global routing prefix, the subnet Identifier (ID) and the interface ID. This address is used to communicate between devices not in the same link, which is always the case when using BLE. Due

to the BLE topology, any communication between two peripherals occurs in different links since it is always routed via the central.

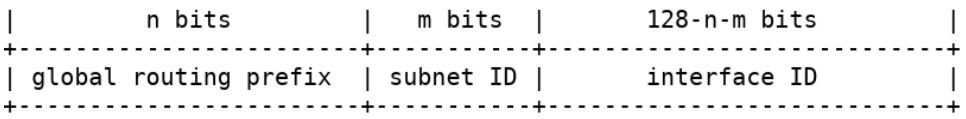


Figure 4.12: IPv6 global address structure [76]

Due to the Stateless Address Autoconfiguration capabilities of IPv6 the devices can define their own address, to do so the BLE device address should be used. The BLE device address is 48 bits long, to form the 64 bit Interface Identifier (IID) two octets are added in the middle of the device address, $0xFF$ and $0xFE$ (*cf.* figure 4.13).

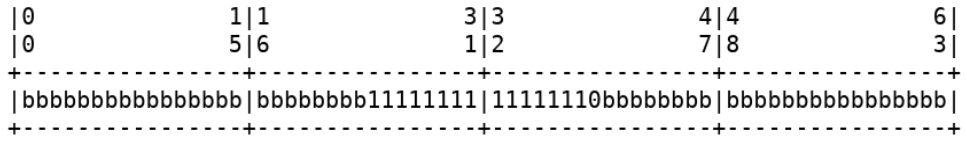


Figure 4.13: Formation of IID from BLE device address [75]

The 6LoWPAN over BLE RFC also states that the global address should not include the IID by default unless it is useful for header compression or if the address is private.

The neighbour discovery in BLE 6LoWPAN networks is slightly different requiring the RA messages to include the "on-link" flag stating that the address belongs to a link. This is required since the BLE star topology requires to have 1 single link between a central and a peripheral.

This page intentionally left blank.

Chapter 5

System Global Architecture

This chapter will present the problems focused and the approach to solve them, the existing solutions and the devices used by the CEiiA for the different applications, presenting the existing limitations and the choices made to solve them.

5.1 Problem Approach

This section will describe the problem that currently exists and the solution for addressing it. It will also explain the need for these solutions and what are the benefits of such implementation.

This thesis, as well as the problems and solutions presented here, will have a focus on IoT solution applied in vehicles which are by nature mobile.

Currently most mobile IoT applications make use of GSM for data transmission, ensuring that the information can be relayed to a central server timely and is available when needed. However the solution is sometimes not cost and energy efficient since the system must always be active and transmitting

data.

5.1.1 Historical Data Offloading

In one common scenario, vehicles that are tracked using Global Positioning System (GPS) communicate their position with a central server for application such as traffic evaluation and route planning. However, nowadays, the systems installed to allow the vehicle tracking have extra functionalities and are capable of gathering vehicle status informations from the Controller Area Network (CAN) bus and report that information back. Some systems also include other useful sensors, such as air quality sensors, that provide an extra set of data leveraging the existing systems.

The main issue with the scenario described above is that not all of this information is useful in real-time (such as air quality data, vehicle status message, driving profiles, etc), while some is useless if received outside of a certain time window (GPS position, vehicle alarms, etc). The first set of information provides intelligence in a big data context where the vast information received from multiple sensors over time can provide an insight on the city and vehicles as a whole. The second set needs to be made available to operators immediately to allow for faster action and responses providing a smoother functioning of the operation.

The proposed solution that will be evaluated in the thesis is how to leverage lower power consumption networks, combined with message routing and selection methods to separate data that should be transmitted by the devices in real time and data that should be kept to be relayed when in proximity to a gateway. These gateways could be positioned in strategic locations, such as overnight parking spots and common traffic routes.

A diagram of the scenario is presented in figure 5.1. In this scenario the

Node would communicate real-time data to the Backend and store historical data which is not required in real-time to transmit opportunistically when in the proximity of a Gateway.

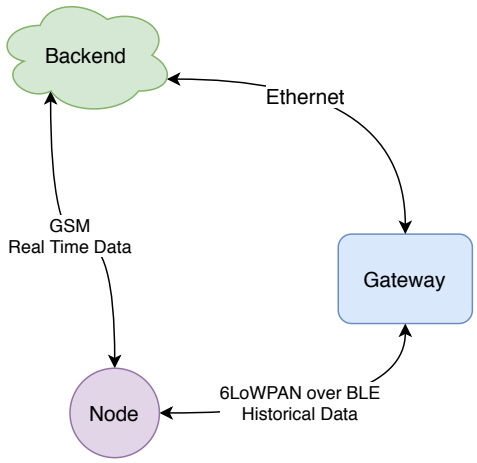


Figure 5.1: Diagram of the Historical Offload Scenario

5.1.2 Offline Functionality Support

Another major concern when using GSM in vehicles as the main form of communication with a central managing server both for reporting data and receiving commands is the spotty or total lack of coverage that sometimes exist in cities despite the big deployment of cellular base stations.

The areas between tall building and in underground parking stations usually do not have perfect coverage making it difficult to ensure constant service availability and allow for the use of the vehicles in underground parking. The platform operation provides multiple vehicle sharing services that have devices installed that communicate with the server via GSM.

One of the existing issues is the impossibility of using the vehicles in underground parking lots. The implementation of a low power network in selected

underground locations and low coverage zones could increase the areas on which the vehicles could be used and parked. The gateways would then relay the information to the server via Ethernet.

The low power network could also be used to determine the approximate location of the vehicles via detection and triangulation.

A diagram of the network proposed is presented in figure 5.2, this network is composed by multiple Nodes and Gateways. In the presented scenario, Node 1 can communicate via the Backend with single hop via Gateway 1. However Nodes 2, 3, 4 and 5 require 2 hops to reach the Backend. Nodes 2 and 3 via Gateway 2 and Node 5 via Gateway 3. Node 4, with the connections represented by dashed lines is in range of Gateway 2 and 3, it can communicate via both or just one depending on the best route to the destination. Node 6 requires 3 hops via Gateways 4, 3 and 1. In this scenario, the Nodes and the Gateways could communicate with each other in the ad-hoc network formed.

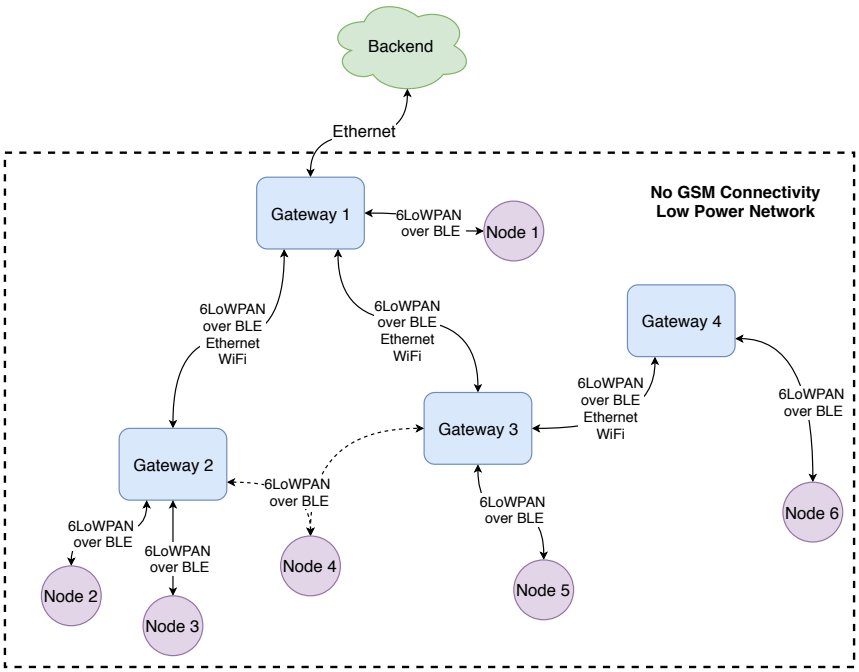


Figure 5.2: Diagram of the Offline Functionality Network

5.1.3 Semi-stationary devices

One major scenario that some mobile IoT devices face is that, despite their nature, most of the time the devices will not move and be stationary in some known position, for instance bicycles in sharing services. This leads to a situation where, similarly to the Historical Data Offloading, data could be permanently sent to a gateway positioned nearby except during the transition between stations.

If any data is required when the vehicle is moving it could be provided by cellular technology, otherwise only some short-range communication method is required. This would greatly reduce the maintenance costs associated with such a system while ensuring its correct operation.

Figure 5.3 shows a Node moving between two stations. Initially it is communicating with Gateway 1 then it moved to Gateway 2 and started communicating with it. During the trip the Node did not communicate with the Backend.

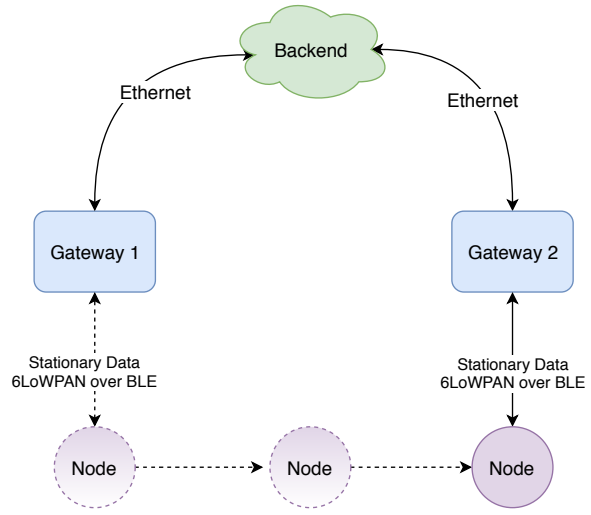


Figure 5.3: Diagram of the Semi-Stationary Device Operation

5.2 Current Solution

This section will briefly present the current existing solutions at CEiiA to easily clarify the proposal that will be presented after.

The devices are constantly reporting all the data that is gathered to the Backend via GSM or Ethernet as can be seen in figure 5.4.

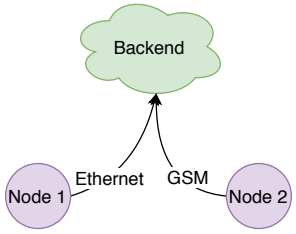


Figure 5.4: Diagram of the current solution

All the solutions integrate different sets of sensors and have BLE capabilities. However not all the devices have the same power constraints and expandability due to the design choices made.
The existing devices will dictate some of the requirements of this work, since one of the goals is to increase the capabilities of the existing devices by leveraging the existing hardware.

All the current in-house solutions use Ethernet to transmit data in the fixed solutions and cellular communication for the mobile applications. This is limited since some data is not relevant in real time and is constantly being transmitted to the Backend. Another limitation is that sometimes devices are close to a fixed station and transmitted their information via the cellular network incurring in higher costs, both in terms of energy and money.

5.3 Proposed Solution

This section will present some of the decisions made as a solution for the problems presented in chapter 5.1. The hardware and software decisions will be justified and the conceptual ideas behind the solutions will be shown.

The solution tries to leverage existing technologies and open-source solutions to ensure that the work can be expanded by avoiding custom, proprietary technologies.

The work will focus on a subset of the issues presented, the subset chosen is one that represents a larger solution and could be applied to both presented approaches. Figure 5.5 is a diagram of the network setup proposed.

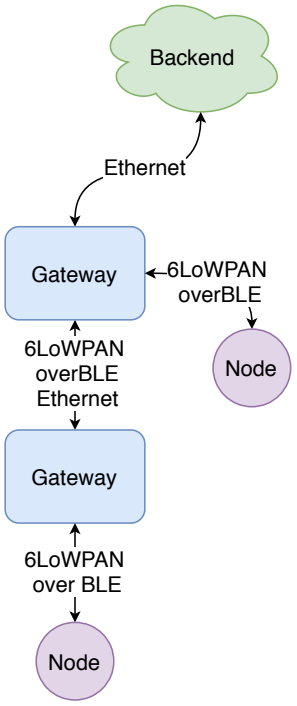


Figure 5.5: Diagram of the proposed solution

The setup includes the three main components to validate that will allow the validation of the network:

- Node to Gateway communication
- Gateway to Server communication
- Node to Server communication

This solutions assumes that the Gateways do not have power constraints and can be placed in locations with non-interruptible power. Considering that power will be more available than an Ethernet connection only one of the Gateways would have wired connection to the Internet.

Since this work aims to leverage the existing products already developed, the goal is to use the current hardware solutions, that are already capable of communicating with the Backend and have BLE capabilities and therefore could behave both as Gateways and Nodes.

Ideally the devices that can communicate via Ethernet and have lower power constraints would act as Gateways but could also act as Nodes. The non Ethernet enable devices, due to their limitations since their focus is on mobile applications which have bigger power constraints, would only act as Nodes.

The proposed solution is detailed in figure 5.6. The Gateway and the Node are similar but with different functions, the devices could have the roles interchangeably depending on which ones are connected to the Backend. The devices include a application layer where the services are running, a 6LoWPAN layer were the packets are converted from IPv6 to 6LoWPAN and a BLE layer responsible for handling the communications.

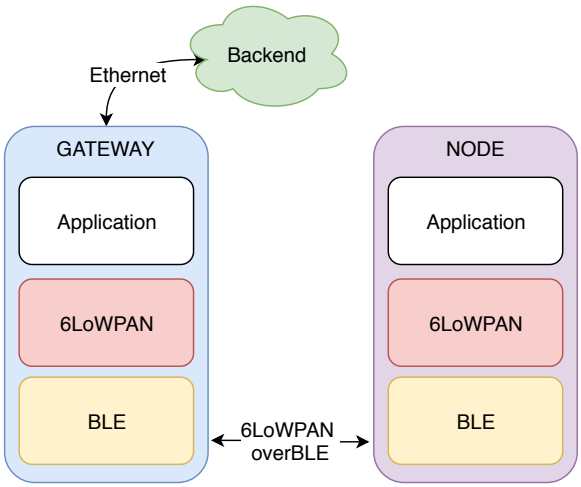


Figure 5.6: Diagram of the proposed Gateway and Node setup

This page intentionally left blank.

Chapter 6

System Development

This chapter is dedicated to the hardware and software development. The following sections will detail the implementations in the different layers from the controller to the application level.

6.1 Hardware

The existing devices were designed with modularity in mind to ensure that they could be applied to multiple projects and use different technology stacks. The main hardware board functionalities can be increased via expander boards that stack onto it and communicate with the central processor via multiple interfaces.

The Gateway and Node hardware topology is show in figure 6.1. This system is composed by 2 different hardware platforms, a main board (represented in yellow) and a nrf52832 development kit (represented in orange). The main board runs the application, the 6LoWPAN stack and the BLE host, while the nrf52832 development kit runs the controller part of the BLE stack.

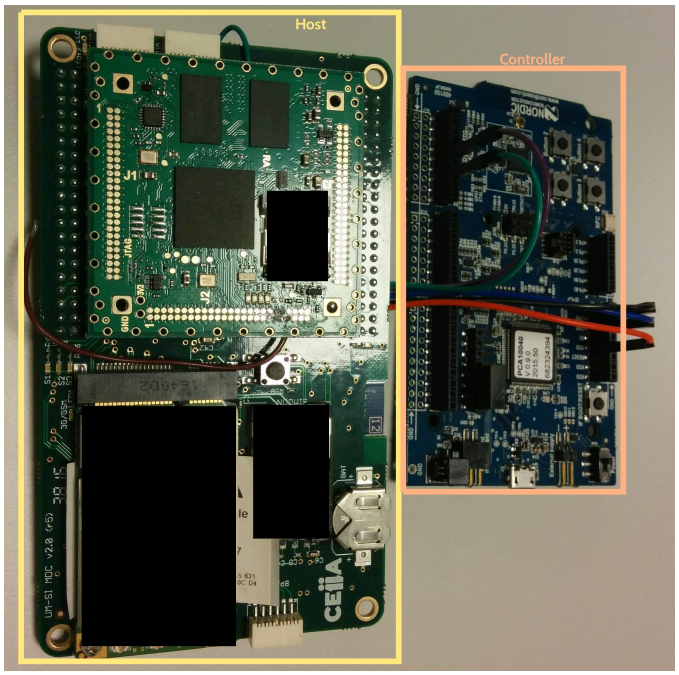


Figure 6.1: Gateway and Node hardware platform

To interface both devices, due to the limitations of the host hardware, UART was chosen as the HCI transport layer. The interface uses the parameters defined by the BLE specification and presented in section 3.1, however flow control is not used due to the unavailability of the signals. Power is also provided to the controller by the main board, figure 6.2 shows the diagram of the connections while figure 6.1 shows the final assembled hardware, with the Host and the Controller identified.

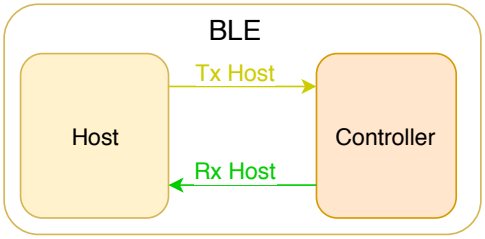


Figure 6.2: Diagram of the hardware connections

For the thesis the node and the gateway use the same hardware setup to simplify the implementation and perform tests without impact of different hardware platforms.

6.2 BLE Controller

The BLE controller is the nrf52832 development kit, the kit is running an open source stack, Zephyr Operating System (OS) [78], which is supported by the Linux Foundation.

Zephyr OS aims to be a small open source real time operating system for IoT embedded devices and is targeted at mainly microcontrollers. It has native support for the nrf52832 development kit and implements the full controller with HCI over UART support.

This open source stack was chosen over the Software Development Kit (SDK) provided by Nordic since it implements the standard HCI interface while the Nordic stack only uses a proprietary interface between a host and a controller. The Zephyr OS stack is also recommended by Nordic when trying to implement a full BLE controller with their hardware [79].

The stack is freely available in Github [80] and includes a UART HCI example compatible with the nrf52832 development kit. Due to the hardware setup only 2 changes were required:

- Disabling Flow Control - since the connections were not available
- Reducing the baudrate to 115200 - since without flow control the host could not communicate at higher speeds

6.3 Linux Kernel

The Gateway and the Node run the application, the 6LoWPAN stack and the BLE on top of a custom OS based on the Linux Kernel 4.17, generated using Buildroot [81].

To ensure that all the required kernel modules were available; BLE, IPv6 and 6LoWPAN support were compiled and preloaded into the kernel.

Before including the required modules into the kernel the version had to be incremented since the original kernel was 4.4.5 and did not include the full support for 6LoWPAN over BLE. In order to do this different patches had to be generated to guarantee that the board booted properly with version 4.17. However, after some initial analysis it was observed that some issues with the 6LoWPAN over BLE existed in the Linux kernel that prevent the connection of more than one Node to a Gateway. This would limit the extent of the possible tests. The current implementation did not allow for a device to be simultaneously a BLE master and slave, limiting the possible topologies to star ones.

Due to the IPv6 to IPv4 translation required the kernel also had to be compiled with support for routing and Network Address Translation (NAT) of IPv6 packets as well as ip6tables to ensure that the rules could be properly applied to the received packets. This modules were not required in the node but only in the gateway since the node has no need to perform routing of information between networks.

6.4 Userspace Applications

Above the kernel, multiple userspace applications were used to perform the different required tasks. Most of these application were only used in the Gateway since the majority of the work required is to route the packets between the different networks. Since the system was generated using Buildroot, most of these packages add to be either enabled on the build configuration, manually added to the package list or upgraded to ensure that the required versions were supported.

6.4.1 BlueZ

Both the node and the gateway run Bluez [82] as the BLE host. Bluez is the official Linux Bluetooth stack and implements the 4.2 BLE Core specification and some profiles defined by Bluetooth SIG. This application was used to connect the host to the controller and perform some initialization tasks before handing the control over to the 6LoWPAN kernel module.

Before initializing the controller it is required to attach it to the host, essentially connecting the BlueZ tools to the HCI transport layer with the correct parameters (*cf.* Listing 6.1).

```
1 btattach -B /dev/ttyS2 -S 115200 -P h4 --noflowctl &
```

Listing 6.1: Attach BlueZ to controller

After attaching to the controller, the gateway needs to be setup as a central device and the node as a server to conform with the IPSP specification. To perform this configuration the bluetoothctl tool is used and the commands presented in listing 6.2 and 6.3 are performed.

On the Gateway, the power is turned on (*cf.* line 1) and the agent is configured as default. Afterwards, the scanning is started to detect BLE devices

(*cf.* line 3).

```
1 power on
2 default-agent
3 scan on
```

Listing 6.2: BlueZ configuration steps - Gateway

On the Node side, the power is also turned on (cf. line 1), followed by the configuration of the BLE agent. In order to be identifiable an advertisement name is set (*cf.* line 3). Finally the advertisement is started so the device can be detected by scanners (*cf.* line 4).

```
1 power on
2 default-agent
3 set-advertise-name node
4 advertise on
```

Listing 6.3: BlueZ configuration steps - Node

After detecting the peripheral from the central, the connection process no longer uses BlueZ but the 6LoWPAN kernel module. In order to connect both devices, the Gateway must request the connection from the Node as can be seen in listing 6.4, where XX:XX:XX:XX:XX:XX is the Media Access Control (MAC) address of the peripheral and the Y indicates the address is static or random.

```
1 echo "connect XX:XX:XX:XX:XX:XX Y" > /sys/kernel/debug/bluetooth
    /6lowpan_control
```

Listing 6.4: Connect via 6LoWPAN

This connection will then create a network interface both on the gateway and the node. An example of such interface can be seen in listing 6.5. In line 1 the hardware address of the interface is present, which corresponds to the BLE MAC address of the device. Line 2 shows the Link-Local IP address generated from the hardware address as described in section 4.4.1.

```
bt0       Link encap:UNSPEC  HWaddr E3-10-38-FF-FE-17-75-BA
    -00-00-00-00-00-00-00-00
          inet6 addr: fe80::e110:38ff:fe17:75ba%790064/64 Scope:
    Link
          UP POINTOPOINT RUNNING MULTICAST  MTU:1280  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:197 (197.0 B)  TX bytes:197 (197.0 B)
```

Listing 6.5: Connect via 6LoWPAN

6.4.2 IPv6 to IPv4

Since IPv6 is not yet widely propagated, the networks in which the system will be deployed will most likely only support IPv4 therefore support for NAT is required. Since the kernel does not support natively Network Address Translation IPv6 to IPv4 (NAT64) an userspace application had to be used. The NAT64 application selected was TAYGA [83]. This application creates a virtual network interface that translates all the received requests between IPv6 and IPv4. Some forwarding rules are required to ensure that the mapping is properly done.

Tayga creates a interface based on some configurations. In order to achieve a proper translation between both IPv6 and IPv4 and back the configuration was created defining the interface name, in this case **nat64**, the assigned IPv6 and IPv4 addresses and the address pool it could use (*cf.* listing 6.6). Using a dynamic pool allows for IP address reuse in case some nodes no longer communicate with the gateway.

```
tun-device nat64
ipv4-addr 10.40.0.1
prefix 2002:db8:1:ffff::/96
```

```
4 dynamic-pool 10.40.0.0/24
```

Listing 6.6: Tayga configuration

Using the previously described tayga configuration, the interface could be created, but before starting tayga some iptables rules add to be defined since NAT within the same address space (IPv6 to IPv6 and IPv4 to IPv4) is performed by the kernel. Listing 6.7 shows the commands used to start tayga, add routing rules and configure iptables.

```
1 tayga --mktun
2 ip link set nat64 up
3 ip addr add 10.40.0.1 dev nat64
4 ip addr add 2002:db8:1::1 dev nat64
5 ip route add 10.40.0.0/24 dev nat64
6 ip route add 2002:db8:1:ffff::/96 dev nat64
7 iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
8 iptables -A FORWARD -i eth0 -o nat64 -m state --state RELATED,
    ESTABLISHED -j ACCEPT
9 iptables -A FORWARD -i nat64 -o eth0 -j ACCEPT
10 ip6tables -t nat -A POSTROUTING -o nat64 -j MASQUERADE
11 ip6tables -A FORWARD -i nat64 -o bt0 -m state --state RELATED,
    ESTABLISHED -j ACCEPT
12 ip6tables -A FORWARD -i bt0 -o nat64 -j ACCEPT
13 tayga
```

Listing 6.7: Start tayga with iptables rules

To configure tayga and the iptables rules, initially, tayga interface is created according to 6.6 (*cf.* line 1) and the **nat64** interface is brought up (*cf.* line 2). The next step is to add both IPv4 and IPv6 addresses to the **nat64** interface (*cf.* line 3 and 4) so routing can afterwards be performed.
With the interfaces created routing rules need to be created so the IP address of the required ranges are sent to the **nat64** interface to be translated (*cf.* line 5 and 6).

After the interface is properly configured the iptables rules are created. The rules are created both for IPv4 and IPv6. For IPv4 all the traffic from **nat64** is sent to **eth0** (*cf.* line 9) and the responses via the same path (*cf.* line 8). For IPv6, the traffic from the **bt0** interface is forwarded to the **nat64** interface (*cf.* line 12) and all the responses routed back (*cf.* line 11).
Finally, tayga is started to ensure that the IPv6 to IPv4 NAT is running when the messages are exchanged.

The **nat64** interface will receive all the IPv6 requests and map them in the 10.40.0.0/24 address range and convert all the IPv4 addresses in IPv6 addresses in the range 2002:db8:1: ffff ::/96 where the last octets are the actual IPv4 address.

6.4.3 DNS

Since the devices that are connected by IPv6 all the requests are being handled by that protocol including the Domain Name System (DNS) requests. To ensure that the requests are all replied with IPv6 addresses and then mapped to IPv4 addresses in the NAT64 layer, the DNS requests need to be properly mapped. To ensure this behaviour the gateway needs to make a DNS server available and the connected nodes need to use that server. To have this behaviour the userspace application Berkeley Internet Name Domain (BIND) [84] was used since it supports the IPv6 to IPv4 translation. Some configurations were required to ensure that no actual IPv6 addresses were sent as a response since the network has no support for it but DNS servers might serve them anyway.

The BIND configuration is presented in listing 6.8. The configuration will always provide a domain IP address in the **nat64** interface address range so i t could be properly translated to the correct IPv4 address and it will always

provide the masked IPv4 address even if the IPv6 address exists since there is no support in the outside network for IPv6.

```
options {
    auth-nxdomain no;
    listen-on-v6 { any; };
    allow-query { any; };
    dns64 2002:db8:1:ffff::/96 {
        # Provide DNS for all clients
        clients { any; };
        # Always provide the masked IPv4 record
        # even if IPv6 record exists
        exclude { any; };
    };
};
```

Listing 6.8: BIND configuration

Chapter 7

Tests and Results

This chapter will discuss the results obtained during this thesis. Two different types of tests were performed: performance and application. These will be discussed in detail in the next sections.

7.1 Performance Tests

To evaluate the performance of 6LoWPAN over BLE an analysis of the impact of the distance and the BLE connection interval in the Round-Trip Time (RTT) was performed.

The description of the test setup and of the methodology as well as the results will be described in the following sections.

7.1.1 RTT and PING

Before describing the methodology used it is important to understand what is being measured and how.

The Round-Trip Time is the duration of time that a signal takes to be transmitted and acknowledged, in IP networks the signal is usually and ICMP packet.

In IP networks, RTT is usually measured using the PING utility and is a measure of the performance of the network; how long a packet takes to travel to a device and back. This is important since it reflects how fast information can be transmitted between two devices and therefore the health of a network. The RTT can be influenced by the distance and the transmission medium among other factors.

PING is an utility developed by Mike Muss [85] to debug IP networks. It is based on ICMP since devices on the network are required to respond to ICMP requests. The utility determines the round trip time between the instant it send the request and receives the response.

A quite comprehensive study on PING and the different configurations that might impact its performance and use as an accurate measuring tool were performed in *Short Term Behaviour of Ping Measurements* [86]. It claims that no literature was found "claiming directly how accurate or inaccurate ping's own measurements are."

However the work raises some concerns on using PING as a network measurement too, nevertheless these concerns apply mostly as a measurement of one way trip time and to the hidden impact of intermediate routers and the path taken between the two machines.

Considering all of the above, PING was considered to be the ideal tool since: no other machine is present in the connection and therefore there is no influence of the path or the intermediate routers performance, the measurements were all analysed on a round trip basis, and ICMP requests and responses are handled on the kernel level reducing the influence of applications running

on the destination.

Besides measuring RTT, the PING utility also analyses packet loss. It accomplishes this by checking whether the responses received match with the requests sent. This is also a good measure of network health since it can be combined with RTT to verify if any node is dropping packets or if they are being lost in the transmission medium.

7.1.2 Methodology

To ensure that the tests are reproducible both within this work and in possible future projects related to this thesis, a methodology was defined for the tests.

Each test consists of a setup with 2 devices a node and a gateway, both the node and the gateway are the same hardware running different software. The gateway is connected with an Ethernet cable via a computer to provide connection to the internet.

Figure 7.1 shows the test setup prepared for the tests at 1 m, with the Gateway at the front of the image and the Node in the back. For the different test distance the Gateway is always static while the Node changes its position.

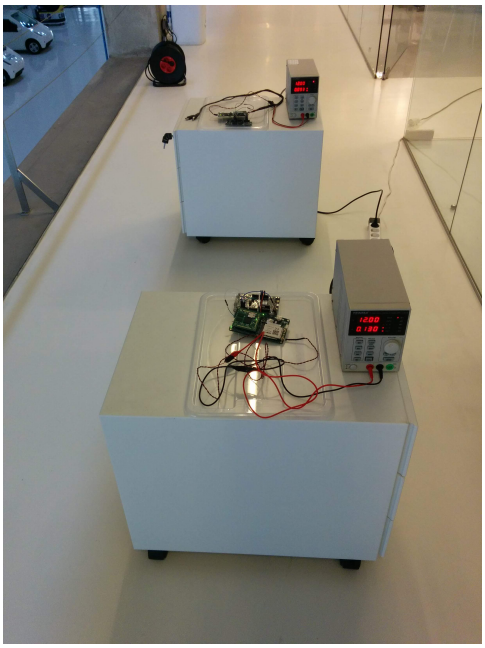


Figure 7.1: Test Setup at 1 m

Each test will evaluate the result of 500 pings with 3 different BLE connection intervals and 10 different distances.

The 3 different BLE connection intervals are 7.5 ms, 50 ms and 100 ms. 7.5 ms is the minimum allowed connection interval, 50 ms is the default value used by the Linux kernel and 100 ms is the double of that one.

The tests were performed at 1 m, 2 m, 5 m, 10 m and then every 10 m up to 70 m. These distances ensure that a wide range is evaluated while gathering more information at smaller distance to better understand the possible impact in the RTT.

These tests were repeated using as destination both the Link-Local address and Global address of the Gateway.

For the 50 ms connection interval, the test were also performed establishing a Virtual Private Network (VPN) connection and sending pings to the VPN

Gateway server. This connection was established using OpenVPN both on the client and the server.

7.1.3 Results

In this section all the results obtained from the performance tests are shown. All the tests were performed according to the methodology presented in the section 7.1.2.

Some exceptions to these tests were made. The tests at 60 m for 50 ms connection interval were performed with only 200 pings since the connection would always drop before reaching 500 pings and the tests at 70 m were not performed since the connection was not stable enough. The tests with 100 ms connection interval were only performed up to 40 m

To remove possible outliers in the tests, the 10 higher and lower RTT values were removed before performing the analysis. This was performed to ensure that artefacts that could skew the results were filtered.

7.1.3.1 Link-Local Tests

The results presented in this section were obtained from performing the performance tests to the Link-Local IP address at multiple distances and for multiple BLE connection intervals.

Tables 7.1, 7.2 and 7.3 present the numerical values for the minimum, mean, maximum and standard deviation for each of the distances and each of the connection intervals.

For a 7.5 ms connection interval, the mean varies between a minimum of 42.543 ms at 2 m and a maximum of 57.331 ms at 60 m.

	RTT (ms)			
Distance	**Minimum**	**Mean**	**Maximum**	**Standard Deviation**
1 m	39.869	43.919	84.691	5.82
2 m	39.793	42.543	45.793	1.299
5 m	39.922	44.099	56.404	3.199
10 m	39.822	42.611	46.257	1.309
20 m	39.962	45.093	59.411	3.947
30 m	39.939	44.13	54.73	3.127
40 m	39.811	43.432	51.878	2.574
50 m	40.589	51.043	74.31	7.769
60 m	41.281	57.331	92.762	11.835
70 m	40.621	51.476	79.164	8.381

Table 7.1: Link-Local IP address RTT with a 7.5 ms connection interval.

For a 50 ms connection interval, the mean varies between a minimum of 95.848 ms at 1 m and a maximum of 199.492 ms at 60 m.

	RTT (ms)			
Distance	**Minimum**	**Mean**	**Maximum**	**Standard Deviation**
1 m	70.814	95.848	119.302	14.049
2 m	71.074	101.702	172.865	21.022
5 m	70.311	101.834	194.698	23.128
10 m	70.716	113.578	211.954	32.46
20 m	71.117	123.391	259.35	41.194
30 m	71.297	118.394	233.443	35.479
40 m	71.551	143.229	313.543	52.637
50 m	71.326	156.668	344.155	61.785
60 m	82.967	199.492	420.439	80.778

Table 7.2: Link-Local IP address RTT with a 50 ms connection interval.

For a 100 ms connection interval, the mean varies between a minimum of 173.959 ms at 2 m and a maximum of 207.749 ms at 30 m.

	RTT (ms)			
Distance	**Minimum**	**Mean**	**Maximum**	**Standard Deviation**
1 m	121.671	173.959	310.645	35.545
2 m	122.472	175.418	287.587	31.912
5 m	122.011	182.408	317.994	41.851
10 m	121.863	183.618	343.327	45.331
20 m	122.641	187.02	360.776	47.532
30 m	122.022	207.749	452.425	65.8
40 m	121.422	187.418	384.457	49.868

Table 7.3: Link-Local IP address RTT with a 100 ms connection interval.

The graphical representation of the data presented in tables 7.1, 7.2 and 7.3 is shown in figures 7.2, 7.3 and 7.4 respectively.

It can be seen that with an increase of distance there is an increase of the maximum and average RTT as well as the standard deviation. This implies that the RTT varies more with the increase in distance. The minimum remains almost constant.

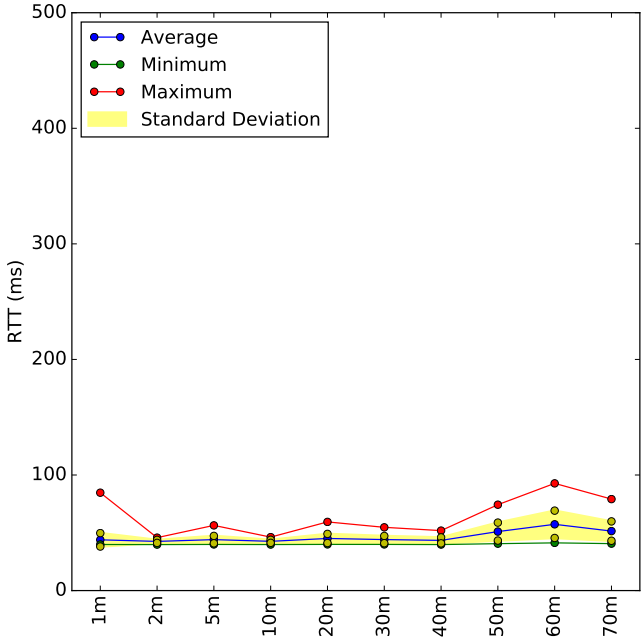


Figure 7.2: Link-Local IP RTT with a 7.5 ms connection interval

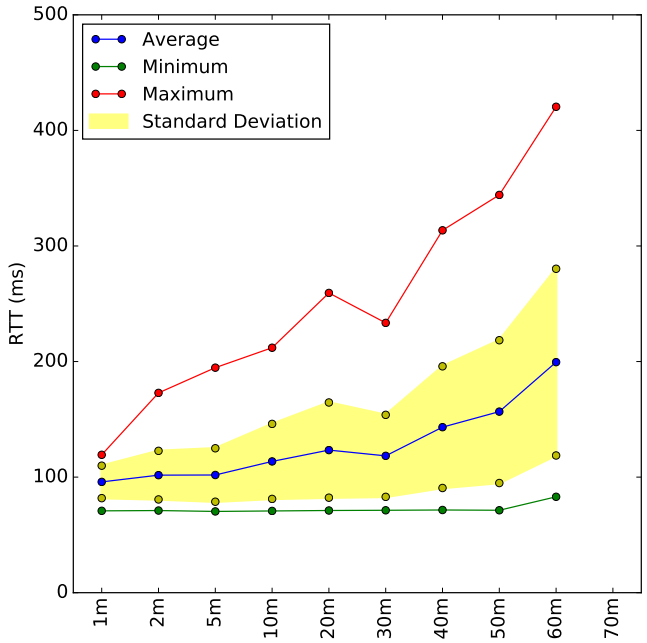


Figure 7.3: Link-Local IP RTT with a 50 ms connection interval

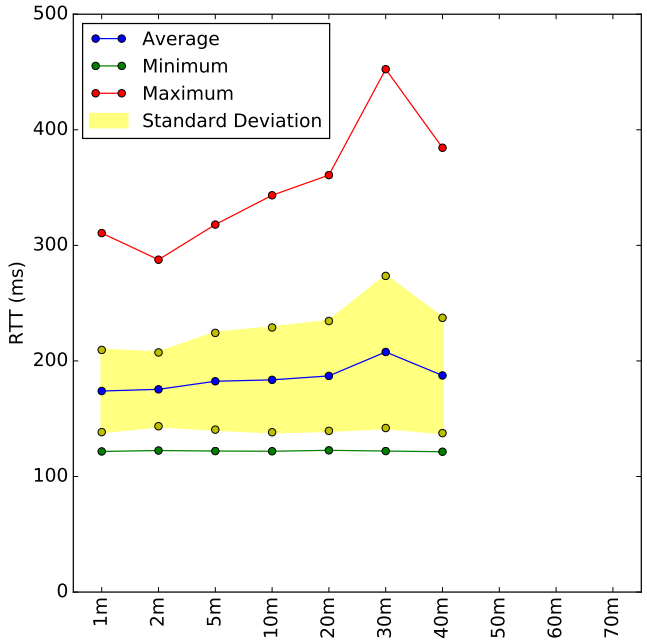


Figure 7.4: Link-Local IP RTT with a 100 ms connection interval

7.1.3.2 Global Tests

The results presented in this section were obtained from performing the performance tests to the Global IP address at multiple distances and for multiple BLE connection intervals.

Similarly to the Link-Local tests, the numerical values for the minimum, average, maximum and standard deviation for each of the distances and each of the connection intervals are presented in tables 7.4, 7.5 and 7.6.

For a 7.5 ms connection interval, the mean varies between a minimum of 49.889 ms at 2 m and a maximum of 68.179 ms at 60 m.

	RTT (ms)			
Distance	**Minimum**	**Mean**	**Maximum**	**Standard Deviation**
1 m	47.838	50.089	55.147	1.429
2 m	47.4	49.889	55.1	1.417
5 m	47.782	50.318	58.552	1.819
10 m	47.5	49.816	55.4	1.534
20 m	48.001	52.989	67.138	4.202
30 m	47.961	52.342	65.686	3.948
40 m	47.943	51.801	64.58	3.403
50 m	48.52	60.314	85.52	8.589
60 m	49.204	68.179	115.977	13.29
70 m	48.518	59.775	84.471	8.06

Table 7.4: Global IP address RTT with a 7.5 ms connection interval.

For 50 ms connection interval, the mean varies between a minimum of 102.354 ms at 1 m and a maximum of 195.434 ms at 60 m.

	RTT (ms)			
Distance	**Minimum**	**Mean**	**Maximum**	**Standard Deviation**
1 m	76.418	102.354	143.677	14.318
2 m	76.243	110.494	206.756	24.829
5 m	76.334	107.68	181.178	22.04
10 m	76.626	113.713	202.426	27.496
20 m	76.439	120.048	234.569	34.941
30 m	76.784	128.14	269.071	41.786
40 m	77.369	149.487	322.2	52.658
50 m	80.386	181.427	415.047	74.81
60 m	87.475	195.434	377.264	77.607

Table 7.5: Global IP address RTT with a 50 ms connection interval.

For a 100 ms connection interval, the mean varies between a minimum of 195.881 ms at 2 m and a maximum of 233.96 ms at 30 m.

	RTT (ms)			
Distance	**Minimum**	**Mean**	**Maximum**	**Standard Deviation**
1 m	127.899	197.294	313.887	46.591
2 m	128.679	195.881	313.363	45.895
5 m	128.575	200.689	363.046	49.503
10 m	128.205	210.614	398.676	58.987
20 m	127.994	204.574	386.367	57.35
30 m	129.155	233.96	470.124	76.278
40 m	128.482	212.786	406.185	60.961

Table 7.6: Global IP address RTT with a 100 ms connection interval.

The graphical representation of the data from tables 7.4, 7.5 and 7.6, as was presented for the Link-Local tests, is shown in figures 7.5, 7.6 and 7.7 respectively.

In the Global tests, an increase of distance also corresponded to an increase of the maximum and average RTT as well as the standard deviation. This implies, as in the Link-Local tests that the RTT varies more with the increase in distance. The minimum remains almost constant.

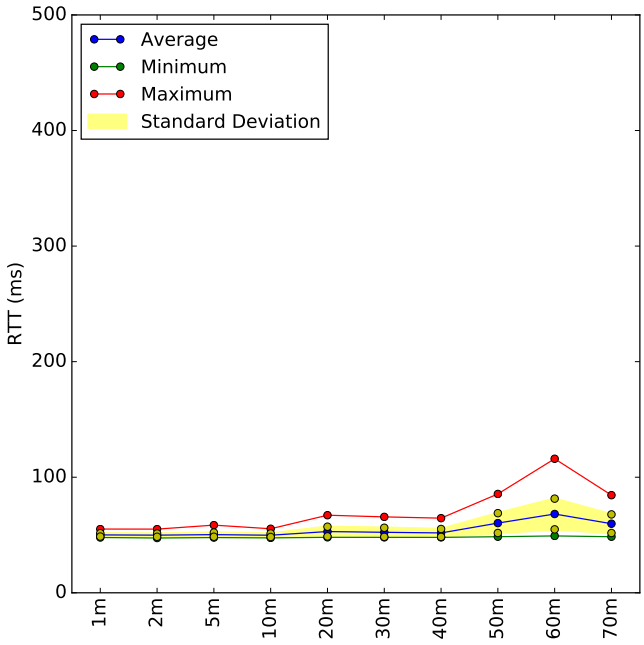


Figure 7.5: Global IP address RTT with a 7.5 ms connection interval

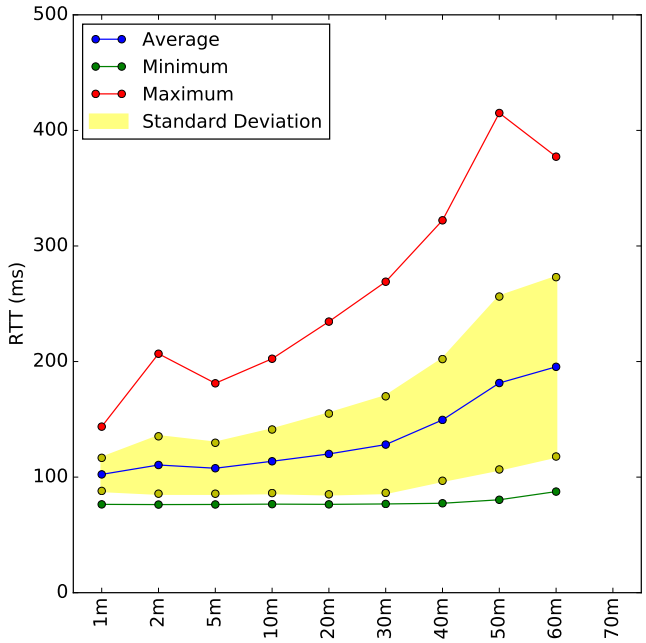


Figure 7.6: Global IP address RTT with a 50 ms connection interval

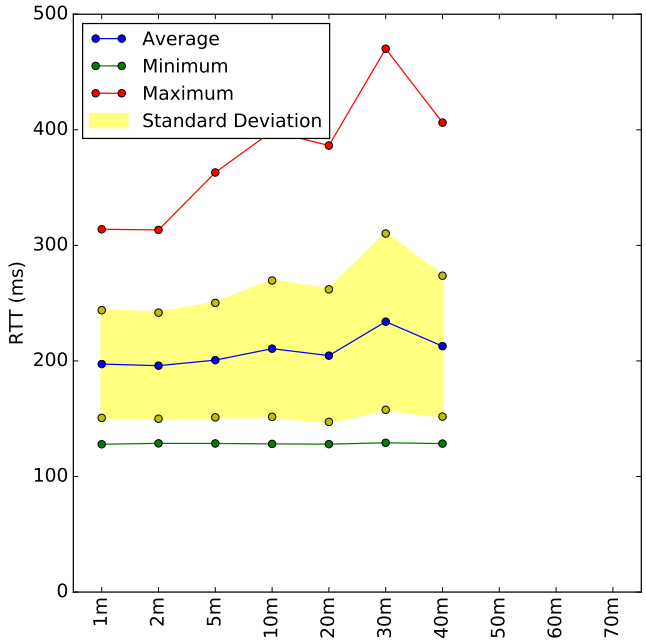


Figure 7.7: Global IP address RTT with a 100 ms connection interval

7.1.3.3 OpenVPN Gateway Tests

The results presented in this section were obtained from performing the performance tests to the OpenVPN Gateway IP address at multiple distances. All the OpenVPN tests were performed with a 50 ms connection interval to analyse the impact of a more common connection would have on the RTT.

The OpenVPN tests also imply the translation between IPv6 and IPv4 to be able to reach the server.

Table 7.7 has the numerical values for the minimum, mean, maximum and standard deviation for each of the distances.

The mean varies between a minimum of 728.021 ms at 5 m and a maximum of 1056.192 ms at 40 m.

	RTT (ms)			
Distance	**Minimum**	**Mean**	**Maximum**	**Standard Deviation**
1 m	167.0	815.14	7545.0	1294.088
2 m	168.0	741.09	5361.0	1050.644
5 m	168.0	728.021	6140.0	1063.318
10 m	169.0	703.394	5129.0	936.09
20 m	176.0	860.737	7647.0	1203.814
30 m	171.0	881.65	8014.0	1291.839
40 m	180.0	1056.192	7114.0	1351.328
50 m	183.0	819.352	5135.0	921.236
60 m	223.0	1043.839	10022.0	1552.1

Table 7.7: OpenVPN gateway IP address results in milliseconds.

The graphical representation of the data in table 7.7 is shown in figure 7.8. From the results obtained the impact of the distance in the RTT are not noticeable, this is due to the impact of the VPN protocol and the fact that

the packets are routed through the wider internet, leading to higher trip times.

The maximum value is much higher and inconstant when compared with both the Link-Local and Global tests performed with a 50 ms connection interval. This might be due to the path taken to reach the OpenVPN server or the load in that server that might influence the response and the time it reaches the server.

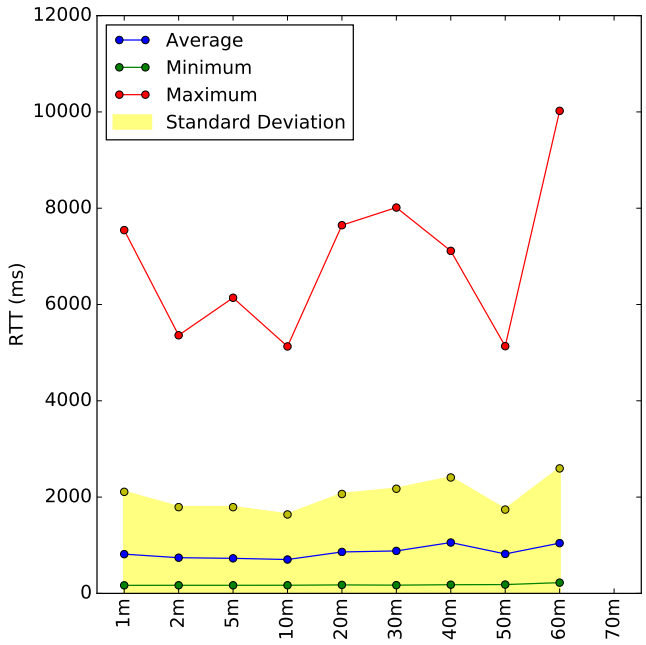


Figure 7.8: OpenVPN Gateway IP RTT in milliseconds

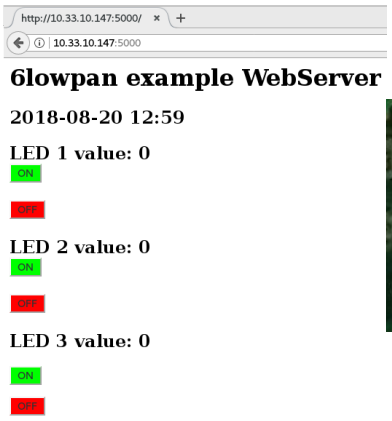
7.2 Application tests

In order to test not only the performance but also the usability of 6LoWPAN over BLE an application was developed that used a similar setup that the actual applications running on the devices. This allows to verify that the chosen stack can handle not only ICMP packets but be used as a medium to serve standard HTTP requests.

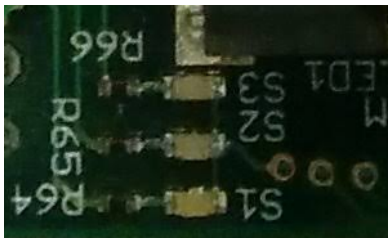
The application consists of a webpage developed using HTML with a Python backend. It allows a user to access the webpage and define the status of 3 LEDs present on the device. It also displays the status of the LEDs and the current device time.

All of the interactions are performed over HTTP with the packets being sent through an OpenVPN tunnel.

The device used to run in the application was the one described in the hardware setup (*cf.* figure 6.1). Figure 7.9 shows the webpage with the status led turned off and the corresponding Light-Emitting Diode (LED) state. Figure 7.10 shows the same as 7.9 but with LED 1 and 3 turned on.

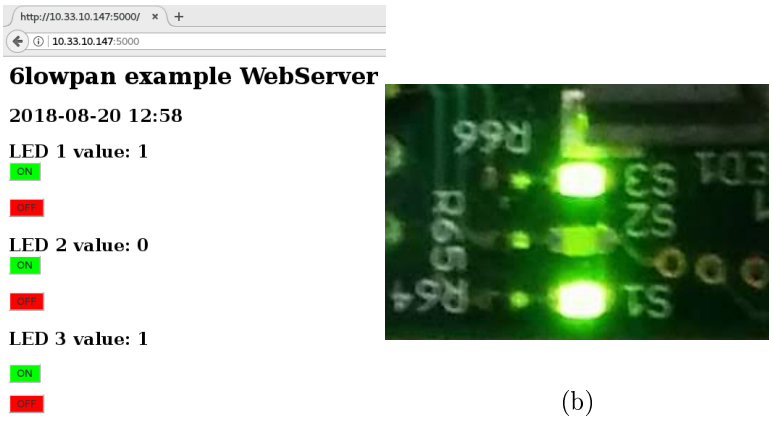


(a)



(b)

Figure 7.9: Webpage (a) and LED (b) status in OFF state



(a) (b)

Figure 7.10: Webpage (a) and LED (b) status in ON state

7.3 Discussion

From the tests performed the setup behaves as expected with the increase in distance causing an increase in RTT when analysing each of the used connection intervals. However contrary to what might be expected there is no packet loss, as presented by the ping utility, this is due to the retry mechanism that the standard BLE link layer implementation includes, this leads to a very stable connection in terms of what the application is concerned.

From the analysis of the results it can also be seen that the minimum is stable along the different distances for the same connection interval. This is interesting since it reflects the fact that under optimum transmission conditions the distance has almost no impact in the RTT. What has a bigger impact in the RTT at greater distances is the higher probability of interferences in the transmitted signal causing an increase of RTT.

The slightly higher RTT seen when using the Global Address when comparing with the Local Address is also as expected since the payload is less

compressed in the 6LoWPAN layer due to the inclusion of the network prefix in the transmitted packets.

The OpenVPN tests show the impact of the translation and the transmission of a higher volume of data over the BLE link. Although the RTT for the pings are quite high, these are expected and do not make the connection unusable as can be seen by the application tests performed. The latency of the BLE connection is hidden by the overhead added by OpenVPN connection and the packet routing via the internet.

The results obtained for the different connection intervals are also coherent with the BLE specification since a lower connection interval means that the two devices communicate more often resulting in a lower latency and lower recovery times if a packet is dropped on the physical layer.

One result that was not expected was the fact that a lower connection interval made communications possible at greater distances. This is caused by the higher interval between communications and the increase in dropped packets, with the distance increase, on the BLE physical layer (which are not perceived on the upper layers) that cause the connection to be terminated by timeout.
Since all the tests were performed with the same supervision timeout (420 ms, which the default supervision timeout in the Linux Kernel) the lower connection intervals could have more BLE packets undelivered before being disconnected than the higher ones. This implies that for the 100 ms connection interval, at least 4 consecutive packets can be lost before the connection is terminated while for a 7.5 ms connection window up to 56 packets can be lost.

A phenomenon that was observed are sudden sporadic drops in the 6LoWPAN over BLE connection that are not recovered. These occur due to connection

loss at the BLE link layer and are easily recovered manually. However due to the implementation of the kernel module the interface is only created upon establishing the connection and does not include a mechanism for automatic recovery.

This page intentionally left blank.

Chapter 8

Conclusions and Future Work

This thesis intended to evaluate network protocols for low-power short range data transmission. Considering the multiple alternatives that are currently available 6LoWPAN over BLE was chosen due to the problems that were presented, ensuring close range transmission of data in areas without cellular connectivity and opportunistically transmitting data over a lower cost link when compared to GSM, and the existing hardware devices limitations.

6LoWPAN over BLE proved to be a valid choice for such applications since the knowledge of IP protocols and the tools that already exist can be reused seamlessly with such a stack. When using the 6LoWPAN over BLE stack alongside other IP stacks (Ethernet, Wi-FI), applications can be developed without specific knowledge of the underlying link and transparently use any of the Internet Protocols.

The impact of the distance on the RTT is not as high as expected with the BLE link being active almost up to the theoretical limit and for distances up to 50 m the stack presents no considerable degradation. The impact of the connection interval is quite noticeable on both the RTT and the distance up to which the communication can be established when considering the

Supervision Timeout used.

All the tests performed present results coherent with both the BLE and the 6LoWPAN specifications. In terms of data transmission distances and configuration impact, and also in terms of data compression in different scenarios with different protocols.

However, since both 6LoWPAN and BLE are quite recent technologies the implementations are not mature enough, for instance, the Linux kernel support is broken for more than two nodes connected to a gateway and neither IOS nor Android support it, it can be hard to utilise properly and interface with multiple different devices. Nevertheless, due to the potential of this stack the support will increase with both proprietary and open source implementations.

Different connection intervals could be used in different applications depending on the resiliency required and the power consumption allowed. The correct configuration of all the BLE link can also impact the usability of the solution at higher distances.

This thesis proved the usability of a stack combining BLE with 6LoWPAN according to the defined standards. Nevertheless proper support for the stack should both be developed and fixed in the Linux kernel implementation. This will allow for the development of Gateways that support multiple connected nodes and benefit from the contributions made by community.

Further work should be done to evaluate the power consumption of this stack, for different connection intervals, to determine its usability in low power nodes deployed in real scenarios. The possibility to implement 6LoWPAN over BLE mesh should also be analysed to remove the overhead of the connection in both the data transmission time and the power consumption that might occur from establishing the connection. A mesh network would also

allow the transmission of data between nodes and the creation of a more resilient network.

Some future work could be performed to try to determine the packet loss depending on the distance on the BLE link layer so that in some applications the supervision timeout could be tuned to the use case.

This page intentionally left blank.

Bibliography

[1] CEiiA - About Us - History. Accessed in January of 2018. Available in: https://www.ceiia.com/history/.

[2] A Caixa Negra - Registo do novo edifício do CEiiA. Accessed in January of 2018. Available in: http://www.acaixanegra.com/works/ceiia/.

[3] Internet of Things forecast. Accessed in July of 2018. Available in: https://www.ericsson.com/en/mobility-report/internet-of-things-forecast.

[4] ITU-T Study Group 13, Recommendation ITU-T Y.4000/Y.2060 - Overview of the Internet of things, 06/2012

[5] IoT Standards and Protocols. Accessed in January of 2018. Available in: https://www.postscapes.com/internet-of-things-protocols/.

[6] Designing the Internet of Things: How to Think about the Internet of Things (IoT). Accessed in January of 2018. Available in: https://www.micrium.com/iot/devices/.

[7] Designing the Internet of Things: Embedded Devices. Accessed in January of 2018. Available in: https://www.micrium.com/iot/thing/.

[8] 11 Internet of Things (IoT) Protocols You Need to Know About. Accessed in January of 2018. Avail-

able in: https://www.rs-online.com/designspark/eleven-internet-of-things-iot-protocols-you-need-to-know-about.

[9] Different Wi-Fi Protocols and Data Rates. Accessed in January of 2018. Available in: https://www.intel.com/content/www/us/en/support/articles/000005725/network-and-i-o/wireless-networking.html.

[10] Lora Alliance Technology. Accessed in January of 2018. Available in: https://www.lora-alliance.org/technology.

[11] NB-IOT vs. LoRa vs. Sigfox. Accessed in January of 2018. Available in: https://www.link-labs.com/blog/nb-iot-vs-lora-vs-sigfox.

[12] CAT-M1 vs NB-IoT – examining the real differences. Accessed in January of 2018. Available in: https://www.iot-now.com/2016/06/21/48833-cat-m1-vs-nb-iot-examining-the-real-differences/.

[13] IoT Technology Guidebook. Accessed in January of 2018. Available in: http://postscapes2.webhook.org/internet-of-things-technologies.

[14] Zigbee 3.0. Accessed in January of 2018. Available in: http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/.

[15] Bluetooth - Technology - Radio Versions. Accessed in January of 2018. Available in: https://www.bluetooth.com/bluetooth-technology/radio-versions.

[16] Bluetooth - Technology - Topology Options. Accessed in January of 2018. Available in: https://www.bluetooth.com/bluetooth-technology/topology-options.

[17] Bluetooth low energy (BLE) fundamentals. Accessed in January of 2018. Available in: https://www.embedded.com/design/connectivity/4442870/Bluetooth-low-energy--BLE--fundamentals.

[18] G. Mulligan, **The 6LoWPAN architecture**, Proceedings of the 4th workshop on Embedded networked sensors, 2007

[19] C. Bormann, M. Ersue, A. Keranen, **RFC7228 - Terminology for Constrained-Node Networks**, IETF, May 2014

[20] R. Silva, **IoT on Shared Vehicles**, ISEP, 2016

[21] Understanding IoT Protocols – Matching your Requirements to the Right Option. Accessed in January of 2018. Available in: https://solace.com/blog/use-cases/understanding-iot-protocols-matching-requirements-right-option.

[22] MQTT vs Websockets vs HTTP/2: The Best IoT Messaging Protocol?. Accessed in January of 2018. Available in: https://systembash.com/mqtt-vs-websockets-vs-http2-the-best-iot-messaging-protocol/.

[23] Internet of Things: Battle of The Protocols (HTTP vs. Websockets vs. MQTT). Accessed in January of 2018. Available in: https://www.linkedin.com/pulse/internet-things-http-vs-websockets-mqtt-ronak-singh-cspo.

[24] M. Belshe, R. Peon, M. Thomson, **RFC7540 - Hypertext Transfer Protocol Version 2 (HTTP/2)**, IETF, May 2015

[25] HiveMQ - MQTT Essentials. Accessed in January of 2018. Available in: https://www.hivemq.com/mqtt-essentials/.

[26] CoAP - RFC 7252 Constrained Application Protocol. Accessed in January of 2018. Available in: http://coap.technology/.

[27] Z. Shelby, K. Hartke, C. Bormann, **RFC7252 - The Constrained Application Protocol (CoAP)**, IETF, June 2014

[28] AMQP - Features. Accessed in January of 2018. Available in: `https://www.amqp.org/product/features`.

[29] From MQTT to AMQP and back. Accessed in January of 2018. Available in: `http://vasters.com/blog/From-MQTT-to-AMQP-and-back/`.

[30] R. Cohn, A Comparison of AMQP and MQTT, StormMQ

[31] I. Fette, A. Melnikov, **RFC6455 - The WebSocket Protocol**, IETF, December 2011

[32] Websockets Are Not Magical. Accessed in January of 2018. Available in: `http://timkellogg.me/blog/2015/03/01/websockets-are-not-magic`.

[33] An Overview of XMPP. Accessed in January of 2018. Available in: `http://xmpp.org/about/technology-overview.html`.

[34] XMPP - Internet of Things (IoT). Accessed in January of 2018. Available in: `https://xmpp.org/uses/internet-of-things.html`.

[35] What is DDS?. Accessed in January of 2018. Available in: `http://portals.omg.org/dds/what-is-dds-3/`.

[36] Building the Internet of Things with DDS. Accessed in January of 2018. Available in: `http://www.embedded-computing.com/embedded-computing-design/building-the-internet-of-things-with-dds`.

[37] J. Moy, **RFC2328 - OSPF Version 2**, IETF, April 1998

[38] G. Malkin, R. Minnear, **RFC2080 - RIPng for IPv6**, IETF, January 1997

[39] **ISO/IEC 10589:2002 - Information technology – Telecommunications and information exchange between systems – Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service**, ISO/IEC, November 2002

[40] D. Savage, J. Ng, S. Moore, et al. **RFC7868 - Cisco's Enhanced Interior Gateway Routing Protocol**, IETF, May 2016

[41] T. Clausen, P. Jacquet, **RFC3626 - Optimized Link State Routing Protocol**, IETF, October 2003

[42] T. Clausen, C. Dearlove, P. Jacquet, U. Herberg **RFC7181 - The Optimized Link State Routing Protocol Version 2**, IETF, April 2014

[43] D. Johnson, Y. Hu, D. Maltz, **RFC4728 - The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks for IPv4**, IETF, February 2007

[44] T. Winter, P. Thubert, A. Brandt, et al. **RFC6550 - RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks**, IETF, March 2012

[45] Gateways Power Nearly Every IoT Market as ABI Research Forecasts Global Shipments to Exceed 64 Million Units in 2021. Accessed in January of 2018. Available in: `https://www.abiresearch.com/press/gateways-power-nearly-every-iot-market-abi-researc/`.

[46] IoT gateways and Industrial IoT gateways – usage and evolutions. Accessed in January of 2018. Available in: `https://www.i-scoop.eu/internet-of-things-guide/iot-gateways/`.

[47] Rigado. Accessed in January of 2018. Available in: https://www.rigado.com.

[48] Rigado - Vesta IoT Gateway. Accessed in January of 2018. Available in: https://www.rigado.com/products/iot-gateways/.

[49] Libelium - Technical Overview. Accessed in January of 2018. Available in: http://www.libelium.com/products/plug-sense/technical-overview/.

[50] Libelium - Sensor Cloud. Accessed in January of 2018. Available in: http://www.libelium.com/products/meshlium/wsn/.

[51] Nest. Accessed in January of 2018. Available in: https://nest.com.

[52] Works with Nest. Accessed in January of 2018. Available in: https://nest.com/works-with-nest/.

[53] NORDIC - nRF52832. Accessed in March of 2018. Available in: http://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52832.

[54] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, **RFC4944 - Transmission of IPv6 Packets over IEEE 802.15.4 Networks**, IETF, September 2007

[55] R. Veeramally, Building IPv6 Mesh Network with Zephyr OS, Embedded Linux Conference, March 2018

[56] Bluetooth - Specifications. Accessed in July of 2018. Available in: https://www.bluetooth.com/specifications.

[57] Bluetooth SIG Proprietary, **Bluetooth Core Specification v5.0**

[58] Microchip Developer Help - Introduction to Bluetooth® Low Energy. Accessed in July of 2018. Available in: http://microchipdeveloper.com/wireless:ble-introduction.

[59] Microchip Developer Help - Bluetooth Low Energy Channels. Accessed in July of 2018. Available in: http://microchipdeveloper.com/wireless:ble-link-layer-channels.

[60] Microchip Developer Help - BLE Link Layer Roles and States. Accessed in July of 2018. Available in: http://microchipdeveloper.com/wireless:ble-link-layer-roles-states.

[61] Microchip Developer Help - Bluetooth Low Energy Connection Process. Accessed in July of 2018. Available in: http://microchipdeveloper.com/wireless:ble-link-layer-connections.

[62] Microchip Developer Help - Bluetooth Low Energy Security Modes and Procedures. Accessed in July of 2018. Available in: http://microchipdeveloper.com/wireless:ble-gap-security.

[63] Bluetooth - Specifications - GATT Specification - GATT Overview. Accessed in July of 2018. Available in: https://www.bluetooth.com/specifications/gatt/generic-attributes-overview.

[64] Bluetooth - Specifications - GATT Specification. Accessed in July of 2018. Available in: https://www.bluetooth.com/specifications/gatt.

[65] Internet WG, **Internet Protocol Support Profile**, Bluetooth SIG, December 2016

[66] J. Hui, P. Thubert. **RFC6282 - Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks (IPv6) Specification**, IETF, September 2011

[67] Z. Shelby, S. Chakrabarti, E. Nordmark, C. Bormann **RFC6775 - Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)**, IETF, November 2012

[68] P. Thubert, R. Cragie. **RFC8025 - IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch**, IETF, November 2016

[69] S. Chakrabarti, G. Montenegro, R. Droms, J. Woodyatt. **RFC8066 - IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) ESC Dispatch Code Points and Guidelines**, IETF, February 2017

[70] S. Deering, R. Hinden. **RFC1883 - Internet Protocol, Version 6 (IPv6) Specification**, IETF, December 1995

[71] S. Deering, R. Hinden. **RFC2460 - Internet Protocol, Version 6 (IPv6) Specification**, IETF, December 1998

[72] S. Deering, R. Hinden. **RFC8200 - Internet Protocol, Version 6 (IPv6) Specification**, IETF, July 2017

[73] T. Narten, E. Nordmark, W. Simpson, H. Soliman. **RFC4861 - Neighbor Discovery for IP version 6 (IPv6)**, IETF, September 2007

[74] S. Thomson, T. Narten, T. Jinmei. **RFC4862 - IPv6 Stateless Address Autoconfiguration**, IETF, September 2007

[75] J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, Z. Shelby, C. Gomez. **RFC7668 - IPv6 over BLUETOOTH(R) Low Energy**, IETF, October 2015

[76] R. Hinden, S. Deering. **RFC4291 - IP Version 6 Addressing Architecture**, IETF, February 2006

[77] NORDIC - nRF52 DK. Accessed in July of 2018. Available in: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52-DK

[78] Zephyr Project - A SMALL, SCALABLE OPEN SOURCE RTOS FOR IOT EMBEDDED DEVICES. Accessed in July of 2018. Available in: https://www.zephyrproject.org/

[79] Nordic Semi **Zephyr Bluetooth Low Energy Controller** November 2017. Available in: http://infocenter.nordicsemi.com/pdf/nwp_029.pdf

[80] Zephyr OS - Source Code. Accessed in July of 2018. Available in: https://github.com/zephyrproject-rtos/zephyr

[81] Buildroot - Making Embedded Linux Easy. Accessed in July of 2018. Available in: https://buildroot.org/

[82] BlueZ - Official Bluetooth Protocol Stack. Accessed in July of 2018. Available in: http://www.bluez.org/

[83] TAYGA - Simple, no-fuss NAT64 for Linux. Accessed in July of 2018. Available in: http://www.litech.org/tayga/

[84] BIND - Versatile, Classic, Complete Name Server Software. Accessed in July of 2018. Available in: https://www.isc.org/downloads/bind/

[85] The story of PING. Accessed in August of 2018. Available in: http://ftp.arl.mil/mike/ping.html

[86] X. Deng, **Short Term Behaviour of Ping Measurements**, University of Waikato, 1999

This page intentionally left blank.

Appendice A. Timeline

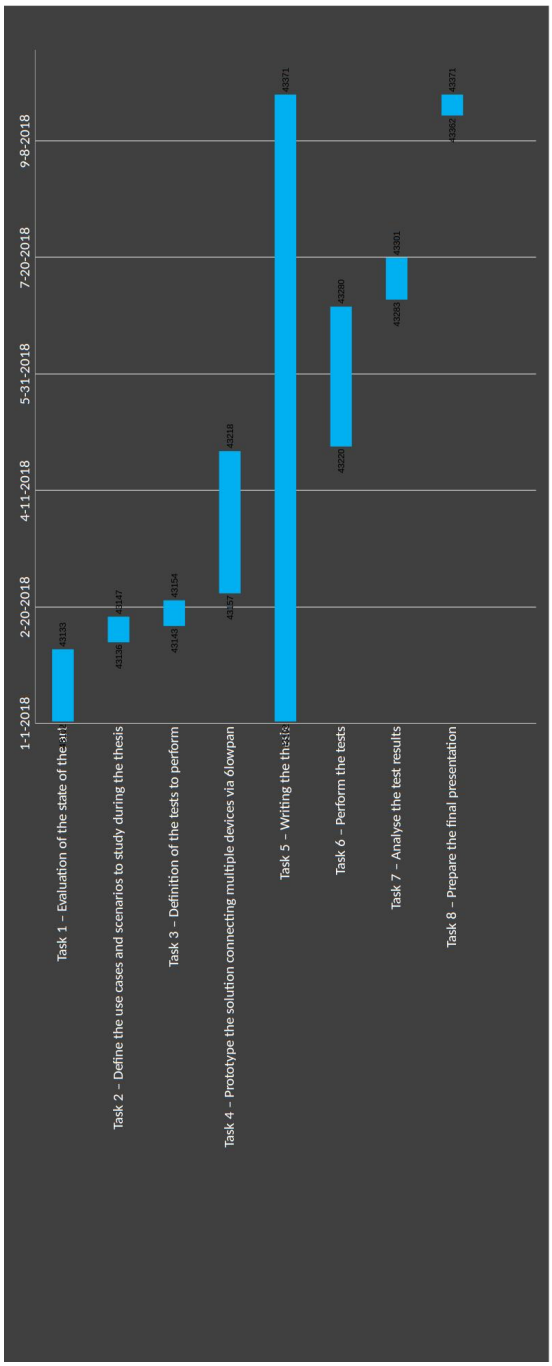


Figure 1: Project Timeline

This page intentionally left blank.