Contributions to Bluetooth Low Energy Mesh Networks

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Enginyeria Telemàtica

by

Seyed Mahdi Darroudi

Doctor of Philosophy
PhD Programme of Network Engineering
Network Engineering Department
Universitat Politècnica de Catalunya

Advisor: Prof. Carles Gomez

Castelldefels, Barcelona, Spain, July 2020

Acknowledgement

"Learn knowledge all along the way from crib to coffin" *prophet Muhammad*

First of all, I would like to express my sincere gratitude to my fantastic advisor, Prof. Carles Gomez, for his continuous support all along the PhD research process. He really provided me motivation while supported me with his immense knowledge. He had a great contribution of rising the idea, developing the structure of the research and consolidated the idea to get maturated. Besides, he was very much supportive on all relevant publication we have had during these years. I have also taken the support and help of Dr. Lluis Casals Ibañez and Dr. Rafael Vidal Ferré and I would like to thank them so much.

My father always has been a hero in my life. He has been an important motivation and pattern in my study career and I have always taken advantage of his leadership. Moreover, I would like to thank my mother and my parent-in-law for all of their emotional and financial supports during these years.

Finally, I would like to dedicate this document to whom I couldn't pass this hard way without her support. She has been alongside me all during these years and she has been not only supportive, but also promising. She is my unexampled and unrivaled wife. She and my lovely daughters have taken many difficulties and life limitations during my PhD program that I own them very much for their endurance and tolerance.

Table of Contents

List of Figures

List of Tables

List of Acronyms

6BLEMesh: IPv6-based BLE Mesh networks
6CO: 6LoWPAN Context Option
6LBR: 6LoWPAN Border Router
6Lo: IPv6 over Networks of Resource-constrained nodes
6LoWPAN: IPv6 over Low power WPAN
6LN: 6LoWPAN Node
6LR: 6LoWPAN Router
AFH: Adaptive Frequency Hopping
ALBER: Adaptation Layer between BLE and RPL
AMP: Alternate MAC and PHY
AoA: Angle of Arrival
AoD: Angle of Departure
AODV: Ad-hoc On-demand Distance Vector
APL: Application layer
APS: Application Support
ARO: Address Registration Option
ASK: Amplitude-Shift Keying
ATT: Attribute protocol
BLE: Bluetooth Low Energy
Bluetooth 4.0: Bluetooth Core Specification Version 4.0
Bluetooth 4.1: Bluetooth Core Specification Version 4.1
Bluetooth 4.2: Bluetooth Core Specification Version 4.2
Bluetooth 5.0: Bluetooth Core Specification Version 5.0
Bluetooth 5.1: Bluetooth Core Specification Version 5.1
Bluetooth 5.2: Bluetooth Core Specification Version 5.2
Bluetooth SIG: Bluetooth Special Interest Group
BMN: BLE Mesh Network
BM-FN: Bluetooth Mesh - Friend Node
BM-LPN: Bluetooth Mesh Low Power Nodes
BPSK: Binary Phase-Shift Keying
BR: Basic Rate
CCCD: Client Characteristic Configuration Descriptor
CRC: Cyclic Redundancy Check
CSMA/CA: Carrier Sense Multiple Access with Collision Avoidance
CSR: Cambridge Silicon Radio
DAD: Duplicated Address Detection
DAG: Directed Acyclic Graph
DAO: Destination Advertisement Object
DF: Direction Finding

DI: Data Interval
DIO: DAG Information Object
DIO: DODAG Information Object
DIS: DAG Information Solicitation
EATT: Enhanced Attribute Protocol
ECI: Expected number of Connection Intervals
EDR: Enhanced Data Rate
ETX: Expected Transmission Count
FLR: Frame Loss Rate
FN: Friend Node
GAP: Generic Access Profile
GATT: Generic Attribute profile
GFSK: Gaussian Frequency Shift Keying
HCI: Host Controller Interface
ICM: Industrial Scientific Medical
IETF: Internet Engineering Task Force
IFS: Inter Frame Space
IIID: IPv6 address Interface Identifier
IoT: Internet of Things
IPSP: Internet Protocol Support Profile
ITU: International Telecommunications Union
L2CAP: The Logical Link Control and Adaptation Protocol
LE: Low Energy
LTK: Long Term Key
LPN: Low Power Node
MD: More Data
MHTS: MultiHop Transfer Service
NA: Neighbor Advertisement
ND: Neighbor Descovery
NDN: Named Data Networking
NESN: Next Expected Sequence Number
NS: Neighbor Solicitation
NWK: Network layer
O-QPSK: Offset Quadrature Phase-Shifting Keying
QPSK: Quadrature Phases-Shift Keying
P2P-RPL: Point-to-Point extension of RPL
PHY: Physical layer
PIO: Prefix Information Option
RIP: Routing Information Protocol
RPL: Routing Protocol for Low power and lossy networks
RT-BLE: Real Time BLE
RTT: Round-Trip Time
SMP: Security Manager Protocol
SN: Sequence Number

STK: Short Term Key
TDMA: Time Division Multiple Access
TTL: Time To Live
UUID: Universally Unique IDentifier
WHAN: Wireless Home Automation Networks
WPAN : Wireless Personal Area Network
ZDO: ZigBee Device Objects

1. Introduction

This chapter provides the motivation, the goals and the organization of this PhD thesis. The motivation for this study is presented in section 1.1. The main contributions from this PhD thesis are listed in section 1.2. Finally, the structure of this PhD thesis document is described in section 1.3.

1.1. Motivation

The Internet of Things (IoT) is a technical megatrend that aims to provide Internet connectivity to resource-constrained, embedded devices [1]. Due to its potential to transform our society, the IoT is nowadays attracting the interest of academia, industry, standards development organizations, and public administration. Some prominent IoT use cases include home automation, industry automation, environmental monitoring, smart cities, and smart grid, among others.

Billions of devices are expected to be connected to the IoT. Given that many IoT devices are characterized by energy constraints, IoT technologies need to enable low energy operation. At the same time, throughput requirements are typically relaxed in IoT applications. Many wireless network technologies comprising Bluetooth Low Energy (BLE), IEEE 802.15.4 (and related protocol architectures, such as ZigBee and Thread), Z-Wave, LoRaWAN, NB-IoT, Sigfox, etc. comply with the mentioned requirements, and are used as enabling technologies in IoT networks [2-7].

Within the wide range of IoT technologies, BLE, also marketed as Bluetooth Smart, has emerged as a major low-power wireless technology [8]. Leveraging a design that can reuse classic Bluetooth circuitry to a large extent, BLE has gained a dominant position as a low-power technology in consumer electronics devices, such as smartphones. This allows low energy communication between the latter and other devices such as sensors, actuators, wearables, etc. [9].

While BLE is currently exhibiting high momentum, it has also been facing significant challenges. A major drawback of a BLE network has been limited coverage range, since BLE was originally designed to follow the star network topology. For example, Wireless Home Automation Networks (WHANs) often require mesh topologies to enable communication between two end devices in a home. For this reason, technologies that support mesh networks are being used in WHANs [10,11]. However, in such a relevant domain, the original star-topology-based BLE approach would limit its use for many applications. A similar problem could be found in any scenario (e.g. industrial, urban, agricultural, etc.) where direct connectivity between any two endpoints might not always be possible [2,12].

In order to cope with BLE network coverage limitations, two main approaches have been proposed by the community. The first one is based on reducing the BLE physical layer bit rate, in order to increase link range while keeping the star topology network model, as introduced by the Bluetooth 5.0 specification [13]. However, this scheme suffers from the hard coverage limitation of a star topology, i.e., extending network coverage beyond one hop is not possible in such topology. Furthermore, a star topology network does not offer path diversity, which is a crucial property in wireless systems in order to cope with radio propagation impairments and node failures. The second approach to overcome BLE network coverage limitations relies on enabling a BLE mesh network. While this model involves the complexity of requiring mesh network mechanisms for end-to-end communication, it allows overcoming the coverage and path diversity limitations of a star topology. These features have attracted the interest of academy and industry, which have developed numerous BLE mesh network solutions by following different techniques [9-29]. Furthermore, in 2015, this interest led to two standards development organizations, Bluetooth Special Interest Group (Bluetooth SIG) and Internet Engineering Task Force (IETF), to launch standardization of mesh functionality for BLE nodes. The Bluetooth SIG produced the Bluetooth Mesh solution, whereas the IETF is as of today completing a specification for IPv6-based BLE Mesh networks (6BLEMesh). Remarkably, the latter is led by us, and it is a contribution of this PhD thesis.

As it has been shown, there has been, and there is, a need to design new mechanisms and evaluate existing mechanisms to enable BLE mesh networks. This PhD thesis aims at advancing state of the art in the presented research area by making the contributions outlined in the next section.

1.2. Results and contributions

The main contributions of this PhD thesis are the following:

a) Surveying and creating a taxonomy of the existing BLE mesh networking solutions.

b) Developing an energy consumption model for Bluetooth Mesh.

c) Designing an IPv6-based solution to enable BLE mesh networks (i.e. 6BLEMesh), in order to enable Internet connectivity for such networks.

d) Developing and validating by simulation an analytical model to predict the connectivity of connection-based BLE mesh networks (such as 6BLEMesh).

e) Implementing and evaluating 6BLEMesh on a real prototype.

f) Comparing the performance of both standardized BLE mesh networking approaches (i.e. Bluetooth Mesh and 6BLEMesh) based on crucial performance metrics, such as latency, energy consumption, message count, and reliability.

1.3. Organization of this thesis

This document is organized in 7 chapters. Chapter 2 presents the state of the art relevant to this PhD thesis, focusing on Bluetooth, BLE, and BLE mesh networking functionality. In Chapter 3, we model the energy consumption of Bluetooth Mesh, considering the impact of its main parameters. The design and an evaluation of 6BLEMesh are provided in Chapter 4. In Chapter 5, we compared Bluetooth Mesh and 6BLEMesh in terms of several performance metrics. Finally in Chapter 6, we conclude the document with the main remarks from this PhD thesis, and propose a number of directions for future work.

2. State of the art: Bluetooth Low Energy and mesh networks

In this chapter, we overview the state of the art of BLE, with a focus on solutions or features relevant for BLE mesh networking. In section 2.1, we explain the evolution from Bluetooth to BLE. In section 2.2, we describe the main features of BLE, in its original star topology design. In section 2.3, we survey and classify BLE mesh networking proposals. In section 2.4, we introduce two BLE mesh networking standardization efforts. Finally, in section 2.5, we complement the state of the art given in this chapter by introducing mesh networks of devices using low power technologies different from BLE.

2.1. From Bluetooth to Bluetooth Low Energy

Bluetooth was created by Ericsson in 1994 as a wireless alternative to data cables for short range communication by exchanging data using radio transmissions. Although it was pronounced dead in 2003 [1], it has become popular in portable devices for audio and data communication.

When the IEEE began the discussion on the technology to be used as low bit rate Wireless Personal Area Network (WPAN), which would eventually become IEEE 802.15.4, several proposals were presented. One of them intended to offer the same radio as Bluetooth but required less power, offering lower bit rate. This proposal was not accepted, and subsequently was pushed in the Wibree Forum as a solution for short

distance communication under the name of "Wibree". The Wibree Forum merged with Bluetooth SIG by mid-2007 and, since that date, a low energy Bluetooth wireless feature, called Bluetooth Low Energy (BLE), has been developed as part of the Bluetooth specification. A key milestone in the specification process was reached in December 2009, with the announcement of the adoption of this technology feature as part of the Bluetooth Core Specification Version 4.0 (Bluetooth 4.0) [14]. Subsequent revisions of that specification are Bluetooth Core Specification Version 4.1 (Bluetooth 4.1), Bluetooth Core Specification Version 4.2 (Bluetooth 4.2), Bluetooth Core Specification Version 5.0 (Bluetooth 5.0), Bluetooth Core Specification Version 5.1 (Bluetooth 5.1), and Bluetooth Core Specification Version 5.2 (Bluetooth 5.2) [13,15-18].

Bluetooth 4.0 and later versions offer two forms of Bluetooth wireless technology systems: Basic Rate (BR) and Low Energy (LE) – the latter corresponding to BLE -. Both systems include device discovery, connection establishment and connection-based communication mechanisms. The BR system includes optional Enhanced Data Rate (EDR) and Alternate MAC and PHY (AMP) layer extensions. The LE system includes features designed to enable lower current consumption, lower complexity and lower cost than the BR/EDR system. The LE system is also designed for applications with lower data rates and has lower duty cycles. In consequence, depending on the application, one system (including any optional parts) may be more optimal than the other.

There may be devices implementing either BR/EDR and LE systems (dual-mode) or only one of them (single-mode). Single-mode BLE is intended for small devices (like watches and sports sensors), and it will enable button cell battery operated devices. Many applications such as healthcare, sports and fitness, security and home automation enhanced with the availability of BLE [12] .

A new radio interface, a new protocol stack, and a new profile architecture were defined for BLE in Bluetooth 4.0. From a marketing perspective, the devices that support BLE are named "Bluetooth Smart" devices. Bluetooth devices that support BLE and BR/EDR are called "Bluetooth Smart Ready" devices. As shown in Figure 1, while classic Bluetooth and Bluetooth Smart devices use different mechanisms and cannot communicate with each other, Bluetooth Smart Ready devices are compatible with both of them [19] .

Figure 1: Different variants of Bluetooth [20].

2.2. Bluetooth Low Energy (BLE)

As introduced above, BLE was defined for the first time in December 2009 by the Bluetooth Special Interest Group (SIG) as part of the Bluetooth 4.0 specification. BLE is used to deliver smaller units of data with remarkably low power consumption, in contrast with older versions of Bluetooth [14].

This section describes the main features of BLE in the Bluetooth specifications, emphasizing the aspects related to mesh network topology support.

2.2.1. Bluetooth Low Energy protocol stack

BLE defines a protocol stack (Figure 2), which is divided into two main sections: the Controller and the Host. The Controller is in charge of lower layer and hardware related tasks, while the Host operates on top of the Controller and provides upper layer functionality. The Controller comprises the Physical Layer and the Link Layer. The Host comprises the Logical Link Control and Adaptation Protocol (L2CAP), the Attribute protocol (ATT), the Generic Attribute profile (GATT), the Security Manager Protocol (SMP) and Generic Access Profile (GAP) Layers.

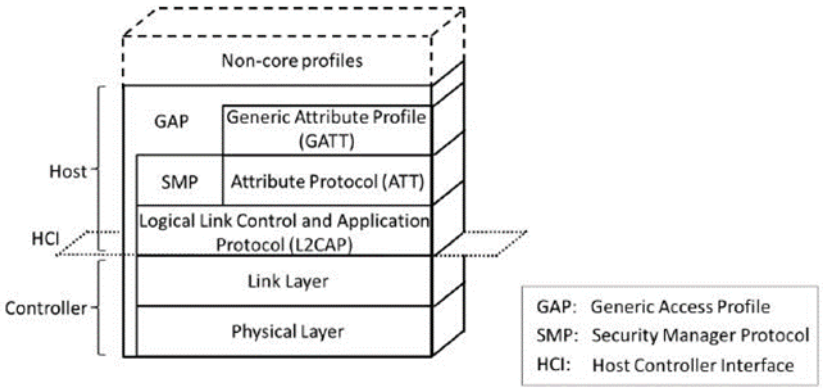


Figure 2: Bluetooth Low Energy protocol stack.

As illustrated in Figure 2, the Controller and the Host communicate via the Host Controller Interface (HCI).

2.2.1.1. Physical Layer

BLE employs the 2.4 GHz Industrial Scientific Medical (ISM) band. It defines 40 different frequency channels in this band, with a 2 MHz channel spacing between adjacent channels. Three out of the 40 channels are advertising channels, which are used for broadcasting, device discovery and connection establishment. The remaining 37 channels are data channels, which are used for sending and receiving data within a connection. As shown in Figure 3, advertising channels are defined to minimize interference with typical IEEE 802.11 channels.

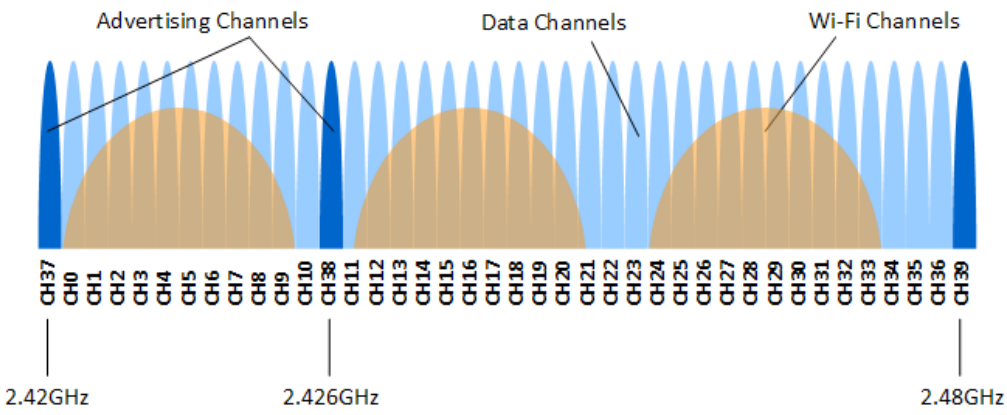


Figure 3: BLE channels Vs Wi-Fi Channels.

An Adaptive Frequency Hopping (AFH) mechanism is used to cope with interference and propagation issues. Gaussian Frequency Shift Keying (GFSK) is the modulation used in the BLE Physical Layer. A modulation index between 0.45 and 0.55 is used to reduce power consumption. The Physical Layer bit rate in Bluetooth 4.x specifications is 1 Mbps (other bit rates, ranging from 125 kbps to 2 Mbps were added in Bluetooth 5.0).

2.2.1.2. Link Layer

The Link Layer is placed on top of the Physical Layer. In BLE, when a node needs to broadcast data, it uses advertising channels to transmit so-called advertising packets carrying such data. The latter are transmitted during an advertising event. In each advertising event, the Link Layer sends advertising packets through one, two or, typically, the three advertising channels available. The node may start another advertising event if needed, and advertising events may occur periodically. On the other hand, the devices that aim to receive advertising packets are called scanners.

For bidirectional data transmission, a connection between two devices needs to be created. To this end, an advertiser needs to announce via advertising channels that it is a connectable device. Then, in response, an initiating device must send a connection request. After that, a point-to-point connection is established, and the two devices can transmit data through the 37 data channels.

The Link Layer defines two different roles for connected devices. The node that was advertising takes the slave role and the scanner node takes the master role.

Within a connection, BLE devices use the aforementioned AFH mechanism for data channel selection. Connection management parameters are announced through the connection request message sent by the initiator/master, but they may be changed during a connection. The connection is divided in time in connection intervals, which

comprise connection events. A connection event is a time interval during which data packets are exchanged between master and slave.

The start point of a connection event is called the "anchor point". The time between two consecutive anchor points is defined by the parameter called *connInterval*, which is a multiple of 1.25 ms in the range from 7.5 ms to 4.0 s. At the beginning of a connection event, the slave is waiting to receive a new data packet from the master. Within *connInterval*, a master may schedule anchor points for the communication with other slaves. The master can have multiple connections in parallel with multiple slaves, and is responsible for coordinating them through a Time Division Multiple Access (TDMA) scheme. Figure 4 depicts an example of TDMA schedule where there are 5 timeslots within *connInterval* time.

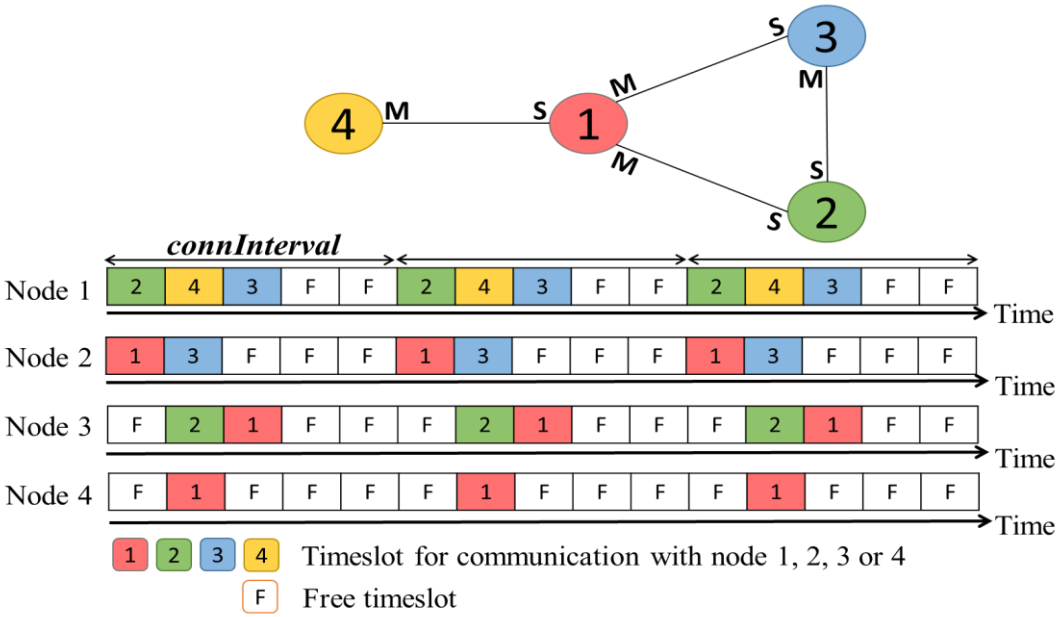


Figure 4: A possible TDMA schedule. M or S denote the master or slave role of a node in the corresponding link, respectively.

A connection event has to finish at least one Inter Frame Space (IFS) before the next anchor point. The IFS is an idle gap between two consecutive packets that is equal to 150 µs.

A connection event is closed for one of the following reasons:

- If none of the devices has more data to transmit, the connection event will be closed and the slave will not be required to listen until the beginning of the next connection event. Data channel packets include a More Data (MD) bit which signals whether the sender has more information to transmit.

- Two consecutive data packets are received with Cyclic Redundancy Check (CRC) error. In order to allow bit error detection, all data units include a 24-bit CRC code.
- A bit error affects the access address. The access address is a unique address that identifies a BLE connection, regardless of the device address that each node has.

Link Layer connections use a stop-and-wait flow control mechanism based on cumulative acknowledgments, which at the same time provides error recovery. Each data channel packet header contains two one-bit fields called the Sequence Number (SN) and the Next Expected Sequence Number (NESN). The SN bit identifies the packet, whereas the NESN indicates which packet from the peer device should be received next. If a device successfully receives a data channel packet, the NESN of its next packet will be incremented, and that packet will be used as an acknowledgement. Otherwise, if a device receives a packet with an invalid CRC check, the NESN of the received packet cannot be relied upon. This forces the receiving device to resend its last transmitted packet, which will be equivalent to a negative acknowledgement.

In addition to *connInterval*, there are also two parameters with significant impact on the performance of a connection, which are presented next:

- ***connSlaveLatency***: This parameter gives the slave device the option of skipping a number of connection events. This feature is called slave latency. If the slave does not have any data to send, it can skip connection events, stay asleep, and save power. The slave device selects whether to wake up or not on a per-connection event basis. The slave can skip connection events but must not skip more than those allowed by the slave latency parameter or if the connection fails. The *connSlaveLatency* must have a value in the range of 0 to ((*connSupervisionTimeout* / *connIntervalMax*) -1), where *connIntervalMax* is the maximum value for *connInterval*, which is negotiated during the connection establishment. *connSlaveLatency* shall be less than 500. When c*onnSlaveLatency* is equal to zero, the slave node has to listen at every anchor point. In Figure 5, an example of slave latency is shown. *connSupervisionTimeout* is defined next.
- ***connSupervisionTimeout***: This timeout is the maximum inactive time for a connection. If no data unit is received by a connected BLE device during this time, the device terminates the connection and returns to an unconnected state. This parameter value is represented in units of 10 ms. *connSupervisionTimeout* can range from a minimum of 100 ms to 32.0 s, and it must be greater than *connInterval*.

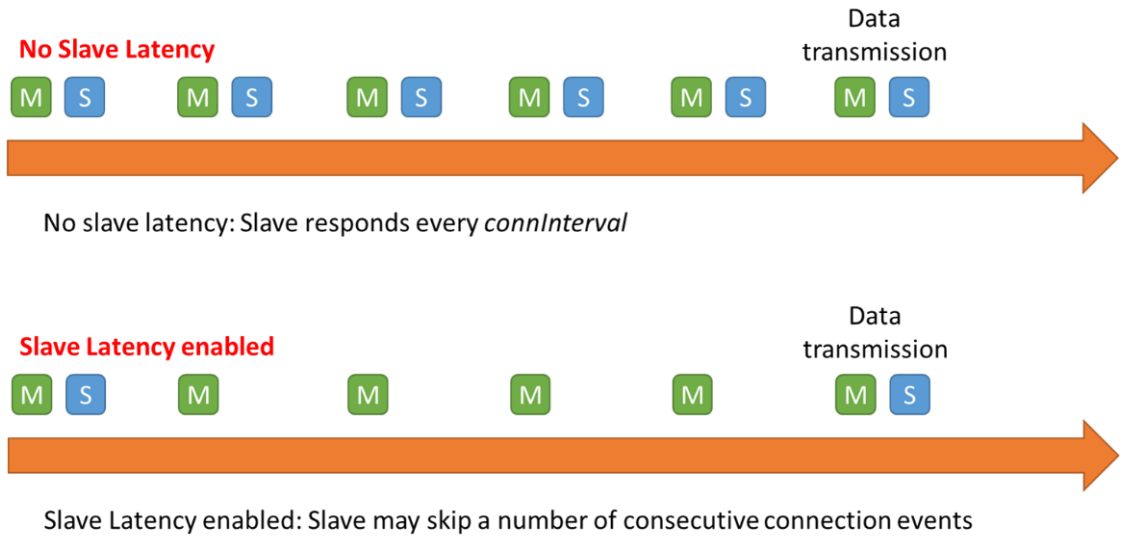


Figure 5: Slave latency example.

2.2.1.3. Logical Link Control and Application Protocol (L2CAP)

The Logical Link Control and Adaptation Layer Protocol (L2CAP) is the lowest layer above the HCI (i.e. the lowest layer in the Host section of the BLE protocol stack). L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing, segmentation and reassembly capabilities. L2CAP allows higher level protocols and applications to transmit and receive messages up to 64 kB in length. The L2CAP supports fragmentation and reassembly, flow control and retransmission.

The L2CAP layer provides logical channels, named L2CAP channels, which are multiplexed over one or more logical links, which are mapped to lower layer links.

2.2.1.4. Attribute Protocol (ATT)

The Attribute Protocol (ATT) defines two roles for devices. A connected device is an ATT server or an ATT client. An ATT server exposes a set of attributes and their associated values to a peer device. These attributes can be discovered, read, and written by an ATT client, and can be indicated and notified by the server.

An attribute is a value that has the following three associated properties:

- Attribute type, defined by a Universally Unique IDentifier (UUID).
- Attribute handle.
- A set of permissions that are defined by each higher layer specification that utilizes the attribute.

A client may send ATT requests to a server, and the server shall respond to all requests that it receives. A device can implement both client and server roles, and both roles can run concurrently in the same device There shall only be one instance of a server on each BLE device; this implies that the attribute handles shall be identical for all supported bearers. For a given client, the server shall have one set of attributes. The server can support multiple clients.

2.2.1.5. Generic Attribute Protocol (GATT)

The Generic Attribute Protocol (GATT) defines a framework that uses ATT in order to discover services and for the exchange of characteristics between connected devices. A characteristic consists of a data set that includes values and properties.

The GATT profile is designed to be used by an application or another profile, so that a client can communicate with a server. The server contains a number of attributes, and the GATT profile defines how to use the ATT to discover, read, write and obtain indications of these attributes, as well as configuring the broadcasting of attributes.

Like the ATT protocol, GATT uses server and client roles as well. However, the roles are not fixed to a particular device. The roles are determined when a device initiates a defined procedure, and they are released when the procedure ends. A device can run both roles simultaneously.

Figure 6 illustrates a simple example of GATT operation, where the computer runs a GATT service client and the temperature sensor corresponds to a GATT service server. The computer initiates a procedure to request data from the sensor. The sensor responds to the computer accordingly.

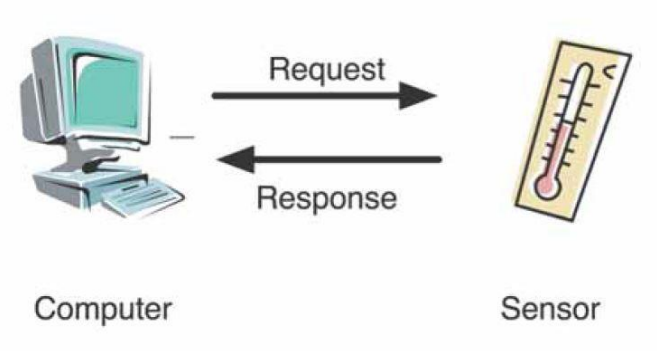


Figure 6: A GATT application example.

2.2.1.6. Security Management Protocol (SMP)

The Security Manager (SM) is an entity that handles pairing, authentication and encryption functionality in BLE radio communication. It uses the SMP for key distribution. Note that BLE security is not a primary topic in this thesis.

2.2.1.7. Generic Access Profile (GAP)

The Generic Access Profile (GAP) defines the generic procedures related to device, service discovery, and connection establishment in BLE. In addition, this profile includes common format requirements for parameters accessible on the user interface level.

There are four GAP roles defined for BLE devices:

- **Broadcaster role**: A device operating in the Broadcaster role can send advertisements. Such device is referred to as a Broadcaster. It shall have a transmitter and may have a receiver.

- **Observer role**: A device operating in the Observer role receives advertisements. This device is an Observer. It shall have a receiver and may have a transmitter.

- **Peripheral role**: A device that accepts the establishment of a Link Layer connection using a connection establishment procedure is called a "Peripheral." Such device will be a slave in the Link Layer connection. A Peripheral shall have both a transmitter and a receiver.

- **Central role**: A device that supports the Central role initiates the establishment of a Link Layer connection. A device operating in the "Central role" is referred to as a Central, and it will be a master in the Link Layer Connection. A Central shall have a transmitter and a receiver.

2.2.2. Bluetooth Low Energy device address

BLE devices are identified by a 48-bit device address. The device address may either be public or random. A public address format follows the IEEE 802 standard. Random addresses are divided into static and private addresses. The private addresses are further divided into two subtypes, namely Resolvable and non-Resolvable addresses. As a summary, all address types are shown in Figure 7.

The static address is assigned once to a device for the whole power cycle duration. The static address may be assigned to another device after the previous power cycle of the device. On the other hand, a private address is randomly changed. The recommended time interval between two consecutive private address changes is 15 minutes.

In random addresses, the first 46 bits are randomized, and the last 2 bits indicate the random address type. BLE does not support avoidance or detection of device address collisions. However, these 48-bit random device addresses have a very small probability of address collision in a typical deployment.

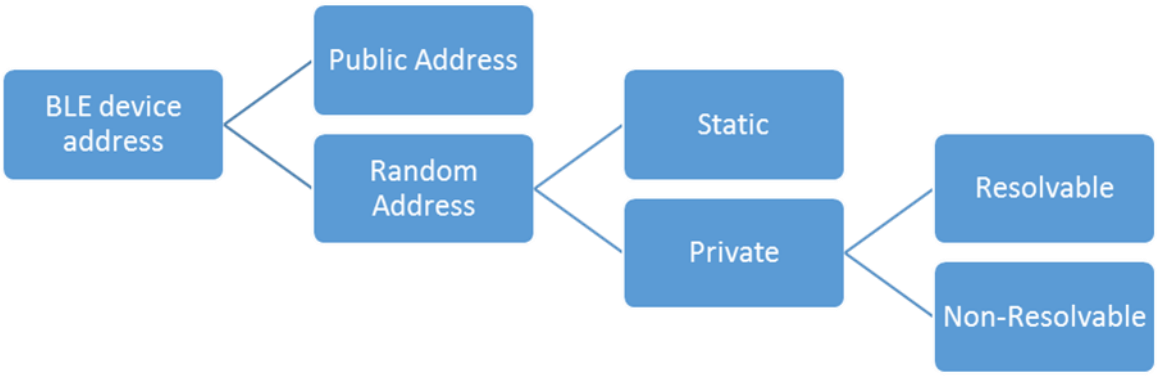


Figure 7: Different kinds of addresses in BLE.

2.2.3. Bluetooth Low Energy specifications and mesh networking support

Six different BLE specification versions have been published to this day. We next briefly overview their main characteristics, focusing on the features that are relevant for mesh networking.

2.2.3.1. Bluetooth 4.0

Bluetooth 4.0 explicitly prohibits a slave node to participate in multiple connections simultaneously with other masters. Thus, the only network topology allowed for a BLE network based on the Bluetooth 4.0 specification is the star topology.

2.2.3.2. Bluetooth 4.1

The Bluetooth 4.1 specification was released in 2013. Bluetooth 4.1 incorporates a fundamental change with regard to BLE mesh network support: a device, regardless of its Link Layer role, can run multiple Link Layer instances simultaneously without limitation. Thus, a slave is allowed to be simultaneously connected to more than one master. In addition, one device can act as a slave in certain intervals, and as a master in others, keeping parallel communications with its neighbors. This opens the door to creating extended network topologies beyond the star topology, such as the mesh topology. However, the architecture and mechanisms for the formation and operation of a BLE mesh network are not defined in the Bluetooth 4.1 specification.

2.2.3.3. Bluetooth 4.2

Bluetooth 4.2, which was published in 2014, incorporates improvements mainly in three areas: Internet connectivity, improved security, and higher throughput. These updates are intended to increase the possibilities of BLE as a technology for the IoT. However, Bluetooth 4.2 does not provide further functionality to support BLE mesh networks.

2.2.3.4. Bluetooth 5.0

Published in late 2016, Bluetooth 5.0 offers improvements in terms of range (a 4-fold improvement compared with Bluetooth 4.2 by using lower data rate transmission), data rate flexibility (offering up to 2 Mbps) and advertising message capacity [21]. However, like Bluetooth 4.2, it does not provide further functionality to support BLE mesh networks beyond those in Bluetooth 4.1 [13].

2.2.3.5. Bluetooth 5.1

Published in 2019, this Bluetooth specification provides improvements in terms of localization, GATT functionality enhancement and advertising enhancement, emphasizing energy efficiency.

On the localizations aspect, the new functionality called Direction Finding (DF) uses two methods called Angle of Arrival (AoA) and Angle of Departure (AoD) to determine not only the distance between two devices, but also the angle of the transmission signal. This option is very useful on indoor localization [17].

Bluetooth 5.1 does not add functionality for BLE mesh networking.

2.2.3.6. Bluetooth 5.2

Published in 2020, Bluetooth 5.2 has been the last Bluetooth specification released as of the writing. This new Bluetooth revision improves user experience on devices where multiple applications use BLE simultaneously. A new Enhanced Attribute Protocol (EATT), which enhances the ATT to support concurrent transactions, and a new L2CAP mode, are now available. Dynamic power control, and support for BLE audio are two further improvements in Bluetooth 5.2 [18]. Bluetooth 5.2 does not provide new BLE mesh networking functionality.

2.3. BLE mesh proposals

In the last few years, there have been various proposals from academia and industry to enable BLE mesh networks. As of the writing, the number of proposals has grown steadily over time [22-41]. We have classified academic solutions into two main categories, namely: Flooding-based and Routing-based solutions (see 2.3.1 and 2.3.2, respectively). The former do not perform routing, instead they broadcast packets throughout the network over BLE advertising channels. The latter use a routing protocol to find a path between two endpoints, and transmit data over BLE data channels. Table 1 summarizes the main characteristics of academic solutions for BLE mesh networks, while Table 2 provides the main performance evaluation results reported by the authors of each solution (when available).

In this section, we first focus on academic, flooding-based solutions (subsection 2.3.1). Next, we study academic, routing-based solutions (subsection 2.3.2). Finally, we also report the publicly known main features of proprietary solutions for BLE mesh networking (subsection 2.3.3).

2.3.1. BLE mesh networks: academic Flooding-based solutions

We next examine two different academic solutions which are based on flooding to allow end-to-end data transmission in a BLE mesh network [22-31,42]. (See Table 1)

Table 1: Main features of academic BLE mesh solutions. Note that, for a given category, solutions are ordered based on the Bluetooth version used (first) and on the year (secondly).

		Proposal reference	Proposal name	Year	Multi-hop paradigm	Bluetooth version	Type of channels	
							Advertising channels	Data channels
Flooding		[22,42]	N/A	2016	Trickle + gossiping	4.0	Data	-
		[23]	BLEmesh	2015	Bounded flooding	4.2	Data	-
Routing	Static	[24]	N/A	2014	Tree-based routing	4.0	-	Data
		[25]	RT-BLE	2016	Pre-configured	4.1	-	Data
	Dynamic	[26]	MHTS	2013	On-demand routing	4.0	Routing	Routing/Data
		[27,28]	BMN	2015	DAG-based routing	4.1	Routing	Data
		[29]	N/A	2015	On-demand routing	4.1	-	Routing/Data
		[30]	N/A	2015	Named Data Networking	4.1	-	Routing/Data
		[31]	ALBER	2016	DAG-based routing	4.1	Routing	Data

A first Flooding-based BLE mesh network proposal is provided in [22,42]. Authors studied how Packet Delivery Ratio (PDR) and latency can be bounded, while keeping low energy consumption. To this end, authors devised a mechanism over Bluetooth 4.0 based on the Trickle algorithm [43]. Their approach follows gossiping [44], whereby traffic is propagated based on a given probability. This probability is determined by a node as a function that decreases with the node's number of neighbors. In addition, Trickle operates based on its own parameters to further filter the decision to

rebroadcast traffic. Authors measured current consumption of BLE nodes, as well as end-to-end packet latency.

On the other hand, a bounded flooding mechanism for BLE mesh networking, called BLEmesh, is presented in [23]. Bounded flooding limits rebroadcasting in intermediate nodes, by only allowing a subset of these to participate in broadcasting operations. In BLEmesh, packets carrying data from a specific sender-destination couple are aggregated in batches. Data, together with control fields which are used to decide which nodes will participate as broadcasters, are carried in the payload of advertisements. The control fields include two lists, namely Forwarder List and Batch Map. The Forwarder List is a prioritized set of intermediate nodes in the path towards the destination which is determined by the sender. The Batch Map identifies the last nodes which have broadcasted data corresponding to a specific batch. When a sender has data ready for transmission, the sender broadcasts the corresponding packet. Each intermediate node that receives a packet compares its priority in the Forwarder List with the one of the last broadcasters shown in the Batch Map. If the node's priority is higher than the one of the last broadcaster, the node sets its own address in the Batch Map as the last broadcaster, and then rebroadcasts the packet (otherwise, the packet is dropped). BLEmesh uses the Expected Transmission Count (ETX) metric to set the Forwarding List value. ETX assigns a cost to a link by estimating the number of transmission attempts needed for successful delivery of a packet via that link [45]. Note that the Batch Map consumes space from already short-sized BLE packets. Authors compared their protocol with basic flooding and unicast source routing, over Bluetooth 4.2. They showed that BLEmesh requires fewer transmissions than basic flooding and unicast source routing as considered in their work.

2.3.2. BLE mesh networks: academic Routing-based solutions

This section overviews academic Routing-based solutions for BLE mesh networks. The section is divided in two parts, which focus on solutions that use static and dynamic routing, respectively.

2.3.2.1. Static routing solutions

Authors in [24] present a solution that creates a static tree topology over Bluetooth 4.0. This scheme includes three kinds of nodes: i) the root node, which is a central device as defined in the BLE specification, ii) intermediary nodes, which actually comprise two subnodes (one acting as a master for nodes located in a lower hierarchical level, and the other acting as a slave for nodes of higher hierarchical level), and leaf nodes, which are set to be peripheral devices. Authors define a simple hierarchy addressing scheme, with two-byte addresses. The addressing scheme allows five tree levels, requiring a bigger address space for deeper networks. Transmission of data from nodes to the root takes place by sending the data from one node to the next one at a higher hierarchical level, and the process is repeated until data reach the root.

The root can also send data to other nodes; in that case, the path is determined based on the destination and intermediate node addresses. Note that addresses in this solution are designed to reflect the hierarchical level and location of a node within the network. This solution is suitable for data collection in WSNs where the root node is a sink node. However, being a tree-based solution, this scheme suffers from the single-node failure problem, and lacks a mechanism to rebuild the network after a node or link failure. Authors measured power consumption, latency and range of this solution experimentally.

Real Time BLE (RT-BLE) is another static routing solution designed over Bluetooth 4.1 intended to enable bounded message delay for BLE mesh networks [25]. In RT-BLE, each node keeps a default route and an alternative route as a back-up. RT-BLE connects subnetworks (comprising a master and its slaves) in order to create an extended BLE network. However, there exist two limitations to network growth: i) a node can establish a Link Layer connection with up to two masters, and ii) a master can establish a connection with at most another master, and in this connection, the former shall play the slave role. In order to avoid overlap of connection events for intermediate slave nodes which are connected to different masters, authors used the Client Characteristic Configuration Descriptor (CCCD), a descriptor available in the GATT layer. CCCD acts as a switch and only allows one connection in active mode at a time, while the rest are kept inactive. Authors analytically calculate the latency in a subnetwork and end-to-end latency between devices in different subnetworks. They also provided experimental inter-subnetwork results by using devices with the X-NUCLEO-IDB05A1 chip from STMicroelectronics.

2.3.2.2. Dynamic routing solutions

Five different dynamic routing-based solutions for BLE mesh networks are presented in this subsection. The solutions vary in the routing approach and in the use of advertising channels, data channels or both for finding routes.

MHTS:

The first BLE mesh solution, called MultiHop Transfer Service (MHTS), was published in 2013 [26]. MHTS was designed over Bluetooth 4.0, based on next-hop, on-demand routing over the GATT layer. MHTS consists of two phases. The first one handles neighbor discovery, connection establishment and route discovery. The second one comprises storing and forwarding the data to be transmitted over the end-to-end path.

During the first phase, neighbor discovery and connection establishment are performed by using common BLE mechanisms. Route discovery is carried out when a packet is ready to be sent but a route to the intended destination is not available in the sender routing table. To initiate route discovery, the sender transmits an advertisement which carries the target destination, the maximum number of hops

between sender and destination (as set by the sender) and the maximum time for the route discovery process. If a neighbor that receives the advertising packet does not know a route to the destination, the neighbor proceeds like the sender and transmits advertising packets requesting for a route towards the destination. This process is repeated until a node that knows a route to the destination is found, which then establishes a Link Layer connection with its precursor neighbor. The latter creates a routing table entry that indicates the neighbor as the next hop towards the destination, the neighbor performs the same operation with its own precursor neighbor, and the procedure is repeated until the source node updates its routing table with the discovered route.

In the second phase, every node in an end-to-end path transmits data as a slave in the Link Layer connection with its next hop, while the latter plays the role of a master in that connection. However, end-to-end transmission is limited by the available memory of BLE devices. With the resources of the CC2640 chip, MHTS can transmit packets over up to 5 hops for a file size of 1 kB [44]. For networks of greater diameter, devices need a larger amount of memory.

BMN:

BLE Mesh Network (BMN) is another routing-based solution that transmits routing messages via advertising channels. BMN uses the Directed Acyclic Graph (DAG) structure as a basis for routing [27,28], inspired by the IPv6 Routing Protocol for Low power and lossy networks (RPL) protocol [46]. BMN was designed over Bluetooth 4.1 and its operation consists of three phases, namely construction, maintenance and optimization.

The construction phase has the goal of establishing Link Layer connections between neighboring devices, determining nodes' parents and creating routing tables. A parent is the next hop for a node in its path towards the DAG root, and thus the parent of a node is placed in a higher hierarchical location than the node. When a node wants to join a network, it transmits DAG Information Solicitation (DIS) messages to announce its presence and solicit DAG information. DAG Information Object (DIO) messages are sent by neighboring nodes in response to DIS messages. Based on the DIO messages received, the node must determine a parent and an alternative parent. A parameter named Rank is defined to specify the quality of routes between nodes and the DAG root. Nodes with lower Rank values are parent (and alternative parent) candidates for new nodes. In BMN, Rank is computed based on nodes' residual energy and distance towards the DAG root. Each node maintains a table where it stores its parent, its alternative parent, and the list of its children (i.e. the nodes that have chosen this one as their parent). On the other hand, the root has a routing table that stores routes for all possible destinations in the network.

The maintenance phase aims to improve BMN parameter settings and forward packets to their intended destinations. To forward a packet, each source node first

looks for a route to the destination in its routing table. If a route is not found, the node sends the packet to its parent. The parent performs the same process, which is repeated until a route to the destination is found. In the worst case, a packet must be sent to the DAG root to be routed to its destination. The optimization phase has the purpose of node weight balancing so that all nodes in the network have nearly equal distance to the DAG root.

BMN sends data messages over data channels, while control messages are sent through advertising channels. Being based on a DAG structure, this solution may suffer issues similar to those of tree-like networks, such as single-node failure (although it is mitigated by alternative parents when available), and congestion in the area close to the root. Authors measured power consumption, latency and data loss experimentally over a network composed of smartphones [27,28].

On-demand scatternet formation and routing:

In [29], authors present a protocol for forming scatternets and on-demand routing for Bluetooth 4.1. Scatternets are network topologies composed of interconnected piconets, while the latter are simple star topology networks comprising a master and its slaves. In order to interconnect piconets, nodes may act as both a master and a slave. This protocol consists of two phases: scatternet formation and route discovery.

In the scatternet formation phase, masters create a list of their connected slaves, and vice versa. Nodes acting as both roles elaborate both slave and master lists. In order to allow connection establishment between a new node and its neighbors, the node alternates scanning and advertising states. The node assumes the master or the slave role, depending on its role as a scanner or an advertiser, respectively, at connection establishment time.

In route discovery, the source node first sends a route request to its master. If the target destination is not in the slave list of the master, the latter initiates a breadth-first search by forwarding the route request to any slave in its piconet which participates in another piconet. Such slaves resend the route request to their masters in the other piconets. This process will continue until the destination is found. By following this process, different routes to a destination are obtained. After collecting all possible routes to a destination, the source node exploits the shortest one and saves it in a route cache. However, network resources are wasted since only one of the discovered routes is used. Another challenge of this mechanism is scalability. Authors evaluated message delay and network throughput experimentally.

Mediation service over Named Data Networking:

Authors in[30] use the Named Data Networking (NDN) [47] to support BLE mesh networks. The Bluetooth version assumed is 4.1. The NDN paradigm changes the networking focus from identifying locations (e.g. as in IP networks, whereby an endpoint communicates with a specific destination where data of interest are), to

identifying the content itself for data retrieval regardless of its location. NDN names every chunk of data with an appropriate URI, and operates over a distributed database which allows determining how an endpoint can retrieve data of interest.

Authors leverage GATT services, characteristics and attributes as the database over which they apply NDN for BLE mesh networks. A Mediation Service [48] is utilized in order to aggregate distributed databases. In this solution, each slice of data is uniquely identified by Universally Unique Identifiers (UUIDs). Two kinds of packets are transferred through the BLE mesh network: Interest and Data. A device requesting data (e.g. the reading of a temperature sensor) sends an Interest packet, and the result is returned in a Data packet following the same route backwards. In order to avoid Interest loops, authors use a Nonce Descriptor, which is a 32-bit random number assigned to each Interest packet. The concept has been proofed by authors over different hardware platforms, but the solution has not actually been evaluated.

ALBER:

Similarly to BMN, another solution based on a DAG structure (in this case created by the RPL protocol), called Adaptation Layer between BLE and RPL (ALBER), has been proposed over Bluetooth 4.1 [31]. ALBER performs 4 different tasks: a) broadcasting RPL control messages through BLE advertising channels, b) broadcasting RPL routing metrics values that reflect BLE link qualities, c) transmitting routing table updates, and d) managing parent changes in order to prevent packet loss.

In order to determine the Rank of a node, in this solution nodes use a metric inspired by ETX. Since the Link Layer in BLE does not provide retransmission information to upper layers, ALBER requires the L2CAP layer to transmit ping packets to parent nodes and uses the obtained Round-Trip Time (RTT) measurements to calculate the Rank value for this node. Authors found experimentally that the RTT in BLE can be computed in terms of *connInterval* periods. Thus, authors defined a new metric called Expected number of Connection Intervals (ECI), which reflects the RTT expressed in terms of *connInterval* periods.

Authors performed an experimental evaluation of their solution. focusing on three main objectives: i) comparing RPL over BLE with RPL over IEEE 802.15.4, ii) determining the impact of the *connInterval* setting, and iii) evaluating the effect of ECI on the performance of RPL over BLE. Regarding the first objective, authors obtained better PDR with RPL over BLE than over IEEE 802.15.4, which they attributed to the adaptive frequency hopping mechanism used in BLE. Moreover, bursty PDR degradation in IEEE 802.15.4 caused frequent parent changes. In relation with the second objective, authors found that PDR performance was improved by reducing *connInterval*. However, increasing *connInterval* decreases energy consumption. Finally, authors found that using ECI reduces the amount of parent changes and improves PDR.

Table 2: Performance reported for BLE mesh network academic solutions.

Proposal	Multi-hop Paradigm	Evaluation Platform	Performance	
			Metrics	Results/Conclusion
[22,42]	Trickle + gossiping	nRF51822 SoC based on ARM Cortex M0	Latency	20 s (3 hops)
			Node lifetime	589 days (6000 mAh battery, 5% duty cycle)
[23]	Bounded flooding	TI CC2540	Packet overhead	16 packets (BLEmesh), 25 packets (source routing), 96 packets (flooding) Note: packet transmission over 5 hops
[24]	Tree-based routing	TI CC2540, SMARTF05 EB	Latency	1.0 s (1 hop), 1.3 s (2 hops), 2.1 s (3 hops)
			Node lifetime	202 days (peripheral), 60 days (central with 3 connected peripherals) Note: *connInterval*=500 ms
[25]	Static routing	X-NUCLEO-IDB05A1 (STM)	Latency	< 0.35 s (5 hops)
[26]	On-demand routing	TI CC2540	Latency	25 s (first packet), 0.79 s (rest of packets) Note: 22-byte packet over 3 hops
[27,28]	DAG-based routing	Smartphone	PDR	98.8% (512-byte file, 5-hop path)
			Latency	2.09 s (2 hops), 2.90 s (3 hops), 3.54 s (4 hops), 4.00 s (5 hops)
			Avg. current consumption	210.9 mA (sender), 228.7 mA (receiver), 234.6 mA (relay) Note: 10-100 kB file transmission
[29]	On-demand routing	Broadcom BCM434x (iPhone 6)	Throughput	~ 6 kbit/s (10-node network)
			Latency	< 5 s (10-node network)
[30]	Named Data Networking	S130 Nordic, TI CC2540, custom prototype	N/A	N/A
[31]	DAG-based routing	MSP430 microcontroller, TI CC2420, Broadcom BCM4356	PDR	~ 100% (*connInterval*=50 ms) ~ 80% (*connInterval*=200 ms)
			Comparison with IEEE 802.15.4	BLE mesh network provides greater PDR, lower number of parent changes and lower overhead
			Impact of ECI metric	Greater PDR (~ 100%) and lower parent changes than without ECI

2.3.3. BLE mesh networks: proprietary solutions

In order to exploit market opportunities for BLE mesh network products, several companies have developed proprietary solutions for BLE mesh networks. A majority of these solutions have been designed for the fields of home automation and/or lighting. However, they may also be suitable for other use cases. We next present a comprehensive set of commercial, proprietary BLE mesh network solutions. Due to the proprietary nature of these solutions, availability of details on the mechanisms used by these solutions is limited. The main features of these solutions are summarized in Table 3.

Table 3: Summary of BLE mesh network proprietary solutions and intended applications (as claimed by each corresponding manufacturer).

Proposal	Name	Year	Mesh paradigm	Intended application
[32]	CSRmesh	2014	Flooding	Lighting, HVAC, switch manager, physical access authorization, smart home
[33]	BLE-MESH.com	2014	Routing	Smart city and home automation.
[34]	Wirepas and Nordic Semiconductor	2014	N/A	Smart home, smart city, smart door lock, lighting, sport, fitness, health, virtual reality
[35]	NXP	2015	Routing	Smart home, smart city, lighting
[36]	Silvair	2016	Flooding	Smart lighting
[37]	Cypress	2016	N/A	Health, fitness, home appliances, toys
[38]	Ilumi MeshTek	2016	Flooding/ Routing	Smart home, remote control, health monitor
[39]	Estimote	2016	Flooding	Beacons for motion detection, guiding.
[40]	Telink Semiconductor	2016	N/A	Lighting, home automation, smart office, smart cities, remote controls, human interface devices, wearable devices
[41]	Mindtree Bluetooth Mesh	2016	Flooding	Multi-purpose

CSRmesh:

Cambridge Silicon Radio (CSR) developed CSRmesh, a proprietary protocol that operates on top of Bluetooth 4.0 and subsequent, which allows forwarding messages across BLE devices in a mesh topology [32].

CSRmesh uses flooding over advertising channels for end-to-end communication. A flat model is used, whereby all devices have the same hierarchical level. Flooding is controlled by using a Time To Live (TTL) mechanism, and by preventing rebroadcast of

the same packet more than once [49]. A CSRmesh network can in theory comprise up to 64,000 devices. Messages can have individual or group recipients. CSR offers CSR101x modules, which support CSRmesh, and the CSRmesh Development Kit, which provides a set of assessment tools and software development for CSRmesh.

Among the proprietary BLE mesh network solutions, CSRmesh appears to be the most popular one, as witnessed by the amount of academic work that is based on this solution [49-51].

Authors in [49] evaluated a BLE mesh network composed of 10 CSRmesh nodes in a building. Performance metrics such as single-hop and multi-hop PDR were analyzed experimentally.

Another CSRmesh case study was presented in [50], although unlike the prior, simulation results were provided in addition to experimental results. Authors evaluated empirically different network scenarios with 19 CSRmesh modules. The maximum PDR was found to be up to 90% for short distance between sender and receiver. Authors showed the benefits of mesh networking to increase PDR over relatively long distances. They also identified a PDR trade-off that depends on the sender packet rate, where PDR is maximized for a sender rate of 4 packet/s.

The same work included the simulation of a 500-node network, whereby nodes were spatially located following a uniform grid pattern and each node had 9 direct neighbors. Authors showed that network size did not have a significant effect on overall end-to-end PDR for the considered node spatial distribution. However, since such node distribution is not usual in realistic mesh network deployments, authors also evaluated a randomly distributed mesh network, where single-node failure and bottleneck problems were found.

A further academic case study based on CSRmesh can be found in [51]. Authors first evaluate CSRmesh, and then improve the basic CSRmesh mechanisms by developing two different solutions: Individual Mesh and Collaborative Mesh. Nodes of the first type only transmit their own data and do not relay packets from other nodes, while nodes of the latter type transmit their own packets and also relay the packets received from other nodes. In addition, a new packet format was designed. These improved mechanisms are compared by the authors with the following two approaches: a Bluetooth 4.0, non-mesh solution, and a mesh solution using basic CSRmesh functionality.

BLE-MESH.com:

BLE-MESH.com is a start-up that created a solution that offers BLE multihop networks. This solution consists of mesh nodes and a mesh gateway, compatible with third party devices. Both mesh nodes and the gateway implement the BLE-MESH routing protocol [33].

Wirepas and Nordic Semiconductor:

Nordic Semiconductor released in 2014 a rebroadcasting mesh solution, namely nRF OpenMesh, for nRF5x family modules [52]. Subsequently, in late 2014, Wirepas and Nordic Semiconductor presented a solution for mesh networks for the nRF51822 BLE chip called Wirepas Pino. This is a proprietary solution that can support a high density of nodes and through which the network topology is continuously self-optimized [34].

NXP:

In 2015, NXP demonstrated a proprietary mesh solution for BLE modules. This solution is based on a synchronized mesh network and routing using BLE. In this solution, each node has its own routing table. Such functionality is available for the QN9020 platform [35].

Silvair:

Silvair developed a proprietary solution for BLE mesh networking, which has been included in smart lighting products. Silvair is one of the major contributors to the development of the Bluetooth specification to the Bluetooth SIG Smart Mesh Working Group.

Silvair solution adds functionality to GATT services and allows direct communication between peripherals (i.e. slaves in traditional BLE), allowing communication over up to 63 hops. One of the concepts used in this solution is connectionless communication [36].

Cypress:

In early 2016, Cypress demonstrated a solution and an implementation for lighting applications that was compliant with the latest proposal at the time from the Bluetooth Smart Mesh Working Group. Cypress offered to be used in health and fitness equipment, home appliances and toys [37].

ilumi MeshTek:

MeshTek is a very recent BLE mesh solution developed by ilumi solutions. In contrast with other proprietary solutions, it can work in two modes. First, it allows broadcasting packets through advertising channels. Secondly, it enables large data packet transfer over a connection-oriented mechanism. Different MeshTek device models are available for different mesh use cases [38].

Estimote beacons:

Estimote introduced their BLE mesh solution with the aim to develop a platform for serving Bluetooth beacons. Since Bluetooth beacons use advertisements to transport data, this solution uses broadcasting to enable mesh communication. These devices announce the readings from sensors of various types. The devices are interconnected

by means of a mesh network topology, which allows control and management settings to be propagated throughout the whole network [39].

Telink Semiconductor:

Telink Semiconductor presented their first BLE mesh lighting solution in early 2016 [53]. Their multi-standard wireless SoC product (i.e. TLSR8269F512) offers BLE mesh networking over Bluetooth 4.2 [54]. This BLE mesh network solution supports concurrent management (e.g. a lighting system can be simultaneously configured and managed by different entities without conflict), group management (i.e. a group of devices can be managed), and real-time message delivery (for a network of up to 200 nodes [40]).

Mindtree Bluetooth Mesh:

Mindtree has developed a solution for BLE mesh networks. The company claims its Bluetooth Mesh's IP to be aligned to Bluetooth SIG's draft Bluetooth Smart Mesh specifications. According to available information on this solution, it supports all mandatory and optional specifications, all states and topology roles, and flooding-based operation. Moreover, it offers application-level encryption. Mindtree products using BLE mesh solutions comprise BlueLitE, EtherMind Bluetooth software [41].

2.4. BLE mesh networks: standardization

Industry and academic interest in BLE mesh networks has triggered two standardization initiatives intended to enable communication between devices in BLE mesh networks. The two standards development organizations responsible for these efforts are the Bluetooth SIG and the IETF, respectively. Such standards are called Bluetooth Mesh and 6BLEMesh, respectively. Figure 8 depicts that Bluetooth Mesh uses flooding over advertising channels for end-to-end packet delivery, while 6BLEMesh leverages Link Layer connections and uses a routing-based approach.

For the sake of clarity and organization, the main details about Bluetooth Mesh, along with an energy consumption evaluation we have carried out, can be found in Chapter 3. Likewise, the main details and energy consumption evaluation results for 6BLEMesh can be found in Chapter 4. Note that we have designed 6BLEMesh and led the corresponding IETF specification, therefore it is one of the main contributions of this thesis.

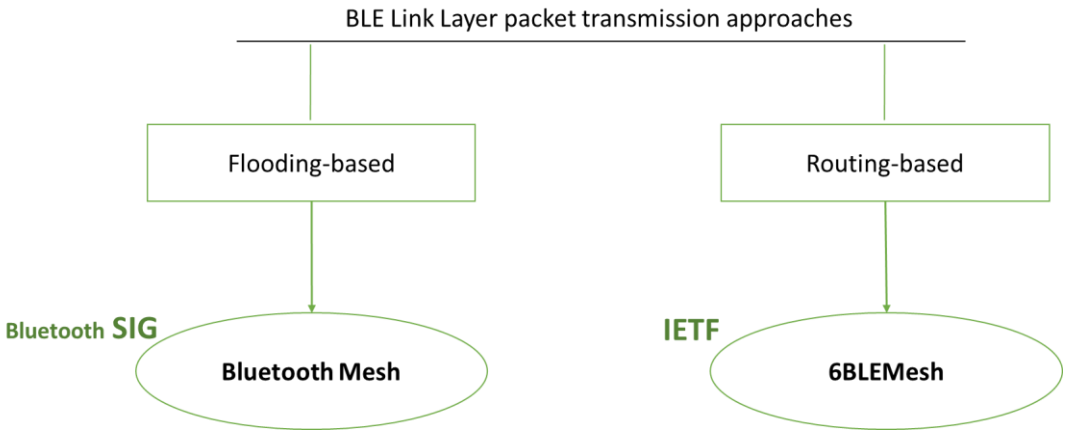


Figure 8: Standardized approaches for BLE mesh networking.

2.5. Non-BLE-based related mesh networks

Many concurrent efforts have been devoted during the last years to mesh networks over technologies different from BLE. This section overviews other technologies that support wireless mesh networks which are related with BLE mesh networks in at least one of the two following aspects: i) they are intended for IoT, or ii) they are based on TDMA principles like BLE. These include IEEE 802.15.4-based mesh networks, Z-Wave, ANT(+) networks and IEEE 802.16a. These technologies are overviewed next.

2.5.1. IEEE 802.15.4

The initial version of the IEEE 802.15.4 standard was introduced in 2003. The first version, i.e. IEEE 802.15.4-2003, was followed by several revisions (IEEE 802.15.4-2006, IEEE 802.15.4-2011 and IEEE 802.15.4-2015, in 2006, 2011, and 2015, respectively). The standard defines physical and MAC layers and aims to provide low complexity, low power consumption and low data rate wireless connectivity among inexpensive devices. The basic IEEE 802.15.4 version operates in ISM 868 MHz, 915 MHz and 2.4 GHz bands. In the 868 and 915 MHz physical layer, IEEE 802.15.4 utilizes Binary Phase-Shift Keying (BPSK), Amplitude-Shift Keying (ASK) and Offset Quadrature Phase-Shifting Keying (O-QPSK) modulation, while in 2.4 GHz physical layer it utilizes Quadrature Phases-Shift Keying (QPSK) modulation. The data rates in IEEE 802.15.4-2003 are 20 kbit/s, 40 kbit/s and 250 kbit/s for the 868 MHz, 915 MHz and 2.4 GHz bands, respectively. There are 1, 10 and 16 different data channels in each ISM band, respectively [55].

At the MAC layer, IEEE defines two main operation modes: with and without beacons. In the former, special nodes called coordinators broadcast beacons periodically. The time between two consecutive beacons is divided into three parts: i) a contention access period, where nodes use Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to access the channel, ii) a contention free period, where nodes

may be assigned a guaranteed timeslot, and iii) an inactive interval where nodes can sleep. In the beaconless mode, nodes use CSMA/CA for channel access [55].

2.5.1.1. ZigBee

ZigBee is a low power wireless protocol stack based on the IEEE 802.15.4-2003 standard. It uses mesh networking and it is targeted to IoT applications.

ZigBee uses IEEE 802.15.4 as its physical and MAC layers, but it defines its own higher level protocol stack (see Figure 9). ZigBee comprises four layers, namely the Physical (PHY) layer, the MAC layer, the Network (NWK) layer and the Application (APL) layer. The NWK layer provides management and data services. The management tasks comprise network formation (which includes the configuration of a new device, starting, joining and leaving a network) and routing [12]. The NWK layer specifically supports addressing and routing for the tree and mesh topologies. In a mesh topology, routes are created on demand and are maintained by using a set of mechanisms based on the Ad-hoc On-demand Distance Vector (AODV) routing protocol [56,57]. With this approach, once a path is found, the relaying nodes store the next hop information in their routing tables. Source routing is another option offered by the ZigBee NWK layer.

The APL layer is composed of three sections: the Application Support (APS) sublayer, the ZigBee Device Objects (ZDOs) and the application framework. The applications themselves are called application objects and are developed by manufacturers for customizing a device for specific applications.

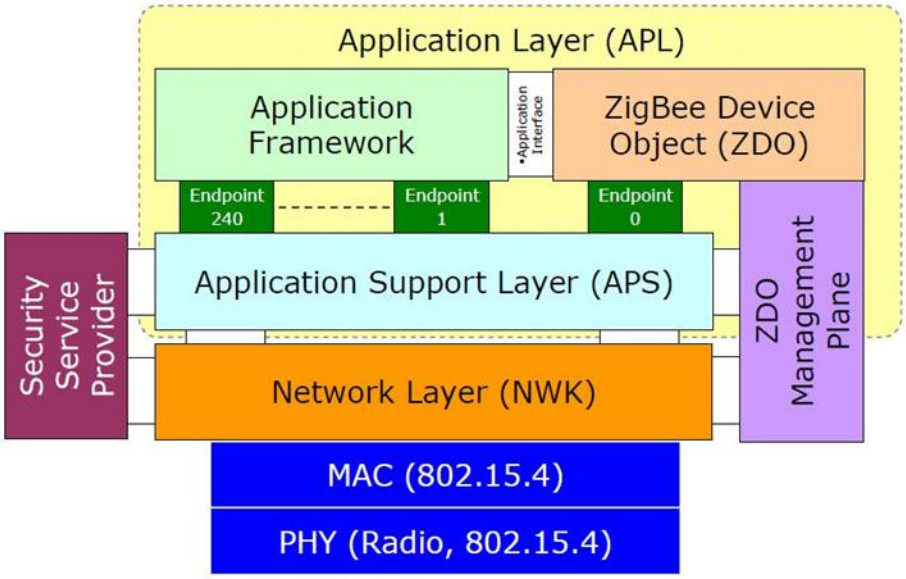


Figure 9: ZigBee protocol stack [58].

2.5.1.2. Thread

Thread is an open standard that defines a protocol stack developed by the Thread Group that was first introduced for reliable, cost-effective, low-power, wireless communication in November 2014. Thread is designed for smart home environments [59].

This standard is based on IEEE 802.15.4 Physical and MAC layers, operating at 250 kbps in the 2.4 GHz band. The IEEE 802.15.4-2006 version of the specification is used in the Thread stack. In Figure 10, the Thread protocol stack is depicted on the left side, while the standards that Thread is using at each layer are depicted on the right.

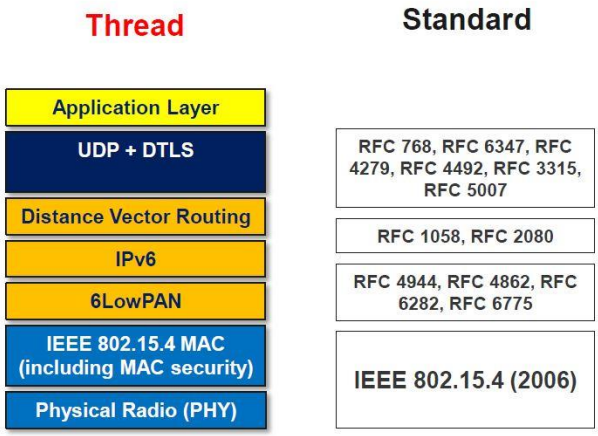


Figure 10: Thread protocol stack [60].

Thread benefits from IPv6 over Low power WPAN (6LoWPAN) to run IPv6 over IEEE 802.15.4 [61] (see section 4.1). Thread supports the mesh topology to transfer data over several hops between two endpoints. Interestingly, the routing protocol used by Thread is based on a distance vector protocol, similar to the Routing Information Protocol (RIP) [62], while RPL has not been chosen [20,63].

2.5.2. Z-Wave mesh networks

A company called Zensys developed Z-Wave as a proprietary wireless protocol stack for home automation. Subsequently, Zensys was acquired by Sigma Designs in 2008, which was in turn acquired by Silicon Labs in 2018. The standard is currently not fully open, in contrast with many wireless standards. Recently, the International Telecommunications Union (ITU) published the Z-Wave PHY and MAC layers as its G.9959 standard, which defines a set of guidelines for sub-1-GHz narrowband wireless devices.

Z-Wave is organized according to an architecture composed of five main layers: the Physical layer, the MAC layer, the Transfer layer, the Routing layer and the Application layer (Figure 11).

At the Physical Layer, Z-Wave uses GFSK modulation. Available data rates are 9600 bit/s, 40 kbit/s and 100 kbit/s. In free space conditions, a range of up to 30 m is possible.

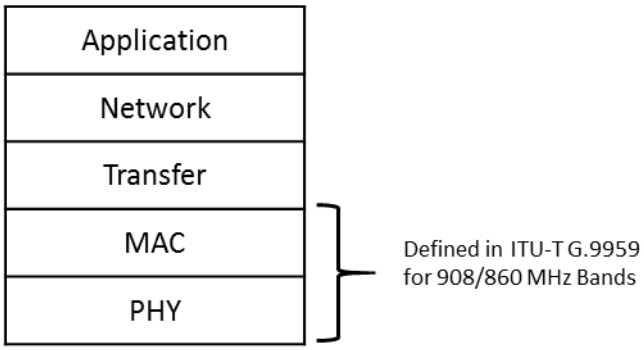


Figure 11: Z-Wave protocol stack.

The MAC layer of Z-Wave defines a collision avoidance mechanism that allows the transmission of a frame when the channel is available. This layer provides frame integrity verification by means of an 8-bit checksum and an optional retransmission mechanism based on ACKs, which is only defined for unicast transmissions. Multicast and broadcast modes are supported.

Z-Wave defines two types of devices, namely: controllers and slave nodes. Controllers send commands to the slaves, which reply to these commands and execute them. A Z-Wave network can have up to 232 nodes.

The Z-Wave Routing layer specifies routing operations on the basis of a source routing approach. When a controller transmits a packet, it includes the path to be followed in the packet. A packet can be transmitted over up to four hops. A controller maintains a routing table that represents the full topology of the network. The routing table is a binary bitmap, which is simple and easy to compress. A portable controller (e.g. a remote control), first tries to reach the destination via direct transmission. If that option fails, then the controller estimates its location and calculates the best route to the destination accordingly. A static controller has the advantage of always knowing its own location in the network. Slaves act as routers and have limited knowledge of the network topology. Routing slaves are a particular type of slave storing static routes and are allowed to send messages to other nodes of the network without being requested to do so.

The Z-Wave routing layer is also in charge of ensuring that a routed packet is correctly forwarded along the end-to-end path. For that purpose, the destination sends an ACK to the source, which is forwarded through the path followed by the data packet in the reverse direction.

The Z-Wave application layer is responsible for the coding and execution of commands in the Z-Wave network. This layer defines application layer messages,

which encode a command and its related parameters. Z-Wave also defines command classes, which are groups of commands. There can be up to 128 command classes used by applications and 256 commands per class. Security services are supported in some Z-Wave products by the use of encryption engines.

The mesh topology is supported in Z-Wave in order to extend the coverage range of a network. Every node is able to determine which nodes are in its direct wireless range. These nodes are called neighbors. During network creation, and later on request, a node is able to inform the controller about its list of neighbors. Using this information, the controller is able to build a table that has all information about possible communication routes in a network.

2.5.3. ANT and ANT+

ANT is a low-power proprietary wireless technology which operates in the 2.4 GHz spectrum. It was created in 2004 by the Dynastream company. The primary goal of this technology is to allow sports and fitness sensors to communicate with a display unit, for example a watch or cycle computer. It typically operates on devices running on a coin cell battery. Similar to other low power consumption technologies, ANT devices may operate for years on a coin cell.

ANT+ has taken the ANT protocol and made the devices interoperable in a managed network, thereby guaranteeing that all ANT+ branded devices work seamlessly.

The inherent ad-hoc nature of the ANT protocol lends itself to supporting the varied and changing requirements of a home automation network. With ANT, a user can have one controller connecting to many sensors, many controllers connecting to one sensor and all combinations between those two scenarios, creating a practical mesh network. However, the mesh operation algorithms (like other procedures) are private and have not been published yet [64].

2.5.4. IEEE 802.11 (Wi-Fi) family

In 2004, a new specification, namely, IEEE 802.11s, amended the IEEE 802.11 standard, aiming to add multihop packet delivery to WLAN. Formerly, multihop WLAN packet delivery was only possible by means of layer three routing or other approaches, but IEEE 802.11s offered the relaying ability at layer two. However, IEEE 802.11s devices are typically not considered to be low power nodes and need to be powered by not constrained energy sources [65].

Another member of IEEE 802.11 family that is more relevant in the scope of this document is IEEE 802.11ah, which was published in 2017. IEEE 802.11ah is a sub-GHz technology that adds low power functionality to the IEEE 802.11 standard, while offering long range (up to 1 km) in a star topology setup. This amendment was designed for IoT environments [66].

3. Bluetooth Mesh energy model

In this chapter, we overview the Bluetooth Mesh standard and model the energy performance of a low power device (more specifically, a battery-operated sensor), denoted Low Power Node (LPN). We develop analytical models for three important energy performance parameters, such as the LPN average current consumption, the theoretical lifetime of a battery-operated LPN, and the energy consumed per each user data bit delivered by the LPN.

The organization of the chapter is as follows. Section 3.1 introduces the Bluetooth mesh standard, describing its protocol stack, different roles and relevant message exchanges between nodes. An analytical energy consumption model is provided in section 3.2. Finally, section 3.3 evalutes energy consumption performance of an LPN in terms of several parameters.

3.1. Bluetooth Mesh overview

In early 2015, the Bluetooth SIG issued a press release announcing the creation of the Bluetooth Smart Mesh Working Group, whose purpose was to develop an architecture to enable support for the mesh topology with BLE [67]. This initiative enjoyed significant support since its inception, with 80 participating companies from a wide range of industries, including automotive, mobile telephony, industrial automation, home automation and consumer electronics. The Bluetooth SIG announced in their 2016 road map that mesh support was part of several enhancements to BLE to better support the IoT [68]. While the aim of the Bluetooth Smart Mesh Working Group was to build a

common platform that can be useful for numerous use cases, home automation is probably the most clearly identified target application domain for BLE mesh networks as per current commercially available proprietary products.

The first specification published by the Bluetooth Smart Mesh working group, in July 2017, was commercially named Bluetooth Mesh (Revision v1.0). The Bluetooth Mesh specification consists of 3 parts: the Mesh profile specification, the Mesh model specification and the Mesh device properties specification. The first one defines the main functionality for enabling end-to-end communication in a mesh network composed of BLE devices, called Bluetooth Mesh nodes. In Bluetooth Mesh, applications are based on a client-server architecture, where servers support resources called states (e.g. on/off variables and their values) and clients operate on such states by using messages (e.g. to toggle a physical switch associated with an on/off variable). A set of states, messages and associated behaviors related with a specific purpose is called a model. The specification defines generic models, and provides guidelines to develop new models. The Mesh device specification describes the minimum capability that a BLE node needs to support as a Bluetooth Mesh node.

The Bluetooth Mesh standard consists of a full protocol stack (Figure 12). At the lowest layer, BLE is leveraged as the means for the physical transmission of Bluetooth Mesh messages. On top of BLE, Bluetooth Mesh defines a set of layers that provide networking and application support functionality. We next review the protocol layers introduced by the Bluetooth Mesh specifications over the BLE protocol stack.

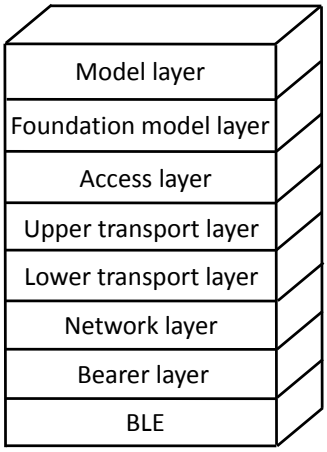


Figure 12: Bluetooth Mesh protocol stack.

The Bluetooth Mesh protocol stack comprises the Bearer layer, the Network layer, the Lower Transport layer, the Upper Transport layer, the Access layer, the Foundation Model layer, and the Model layer.

The Bearer layer defines the lower layer bearer to be used for message transmission. Typically, Bluetooth Mesh messages are sent as advertisements. The Network layer delivers end-to-end data units by means of a controlled flooding mechanism. The Lower transport layer offers reliable segmentation and reassembly for large data units. The Upper transport layer provides support for LPNs, based on a concept called friendship, as well as end-to-end security. The Access layer provides optional end-to-end reliability. The Foundation model layer offers support for management and configuration of a Bluetooth Mesh network. Finally, the Model layer defines a framework for applications.

Since LPNs run on limited energy sources (e.g. small batteries), they remain by default in sleep mode in order to save energy. LPNs can transmit messages at any time, since it is assumed that at least one of their next hop devices will be always ready to receive and forward such messages. However, in order to allow LPNs be able to also receive messages, Bluetooth Mesh Upper transport layer defines the concept of friendship, which is a special relationship between a LPN and a one-hop neighbor which we refer to as Friend Node (FN). The latter, which is selected by the former among its one-hop neighbors, stores messages intended for the LPN while this node is in sleep state. The LPN asynchronously polls the FN for possible incoming messages by sending a request message to the latter. After sending the request, the LPN returns to sleep mode. After *ReceiveDelay* milliseconds, which allow the FN to prepare a response, the LPN starts listening for up to *ReceiveWindow* milliseconds. Upon receipt of a request, the FN can send a stored message to the LPN, if any. After receiving the last stored message, or after the end of the receive interval, the LPN enters sleep mode again. The maximum time between two consecutive requests is defined by the *PollTimeout* parameter (Figure 13).

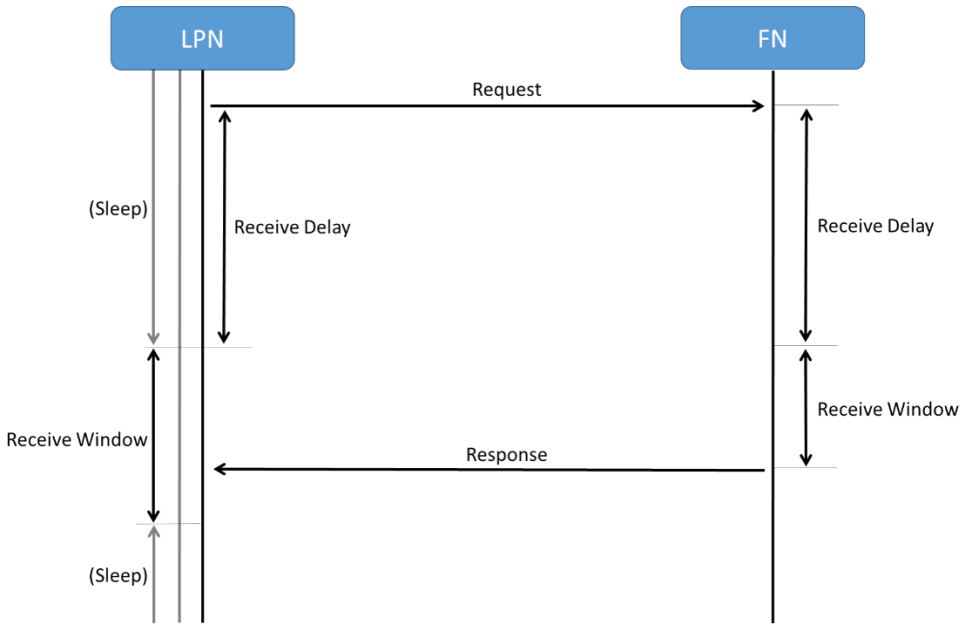


Figure 13: Illustration of a LPN polling a FN, and the related Bluetooth Mesh parameters involved.

Typically, a LPN running as a sensor device will periodically poll the FN and scan the channel after each request for *ReceiveWindow* milliseconds, in addition to sending data messages containing sensor readings (Figure 14). Note that, since advertisements are generally used to carry Bluetooth Mesh messages, one request or one data message transmission is usually performed by sending 3 advertisements.

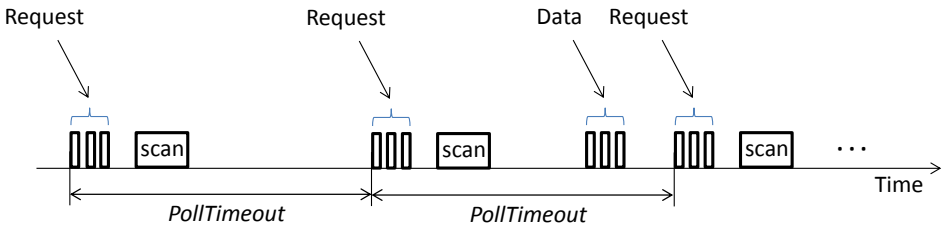


Figure 14: Polling and data message transmission operations carried out by a LPN. In this example, data message transmission is only performed in the second PollTimeout interval.

3.2. Modeling the Energy Consumption of a Bluetooth Mesh Low Power Node

In this section we present analytical models of three important energy consumption parameters of a battery-enabled LPN: average current, battery lifetime, and energy consumed per bit delivered to a neighbor. We assume a LPN running as a battery-operated sensor node that periodically transmits a data message (e.g. containing a sensor reading).

Our first goal is determining the LPN average current consumption, denoted $I_{average}$. As introduced in Subsection 3.1, the LPN stays in sleep mode by default, and every *PollTimeout* sends a request to its FN. In addition, the LPN transmits a data message once every T_{Data}.

In order to capture a realistic behavior, we derive our LPN current consumption model based on measurements carried out on a real BLE device. However, probably due to the novelty of Bluetooth Mesh, the friendship feature is not currently supported in available Bluetooth Mesh implementations. For this reason, we create a model based on the three main actions performed by a LPN when it is not in sleep mode: i) transmitting three advertisements, and ii) scanning the channel. The first action allows modeling the transmission of a request by the LPN, as well as a data message transmission. The second one is useful to model the channel listening performed by the LPN to check for potentially incoming data after a request. We identify and characterize the duration and current consumption of all relevant states corresponding to these two actions, separately, as measured on the tested device.

The device model used in our measurements is a PCA10028 Development Kit from Nordic Semiconductor family (in short PCA10028). This board includes an nRF51422 chipset, which belongs to the popular nRF51 series [69]. This module supports Bluetooth

4.2, as well as a Bluetooth Mesh implementation provided by the manufacturer [69]. During the measurements, the device transmit power is set to 4 dBm. Results are obtained by using an Agilent N6705A power analyzer. Figure 15 illustrates the experimental environment.

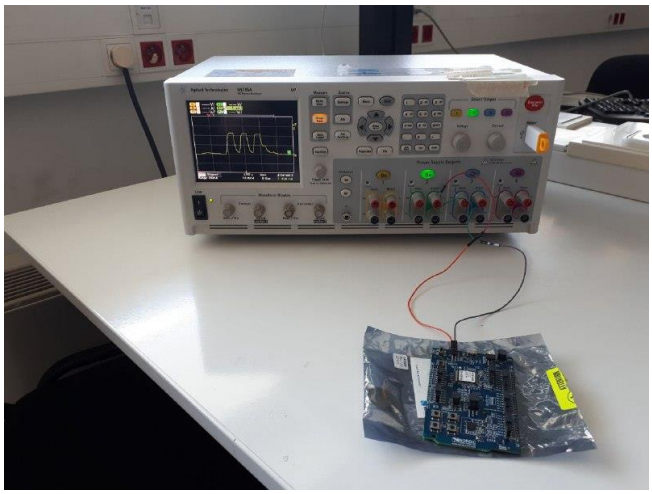


Figure 15: Experimental setup for the current consumption characterization of the nRF51422 PCA10028 board (front) using an Agilent N6705A power analyzer (back).

The current consumption profile of the device during the transmission of three advertisements is illustrated in Figure 16. Table 4 shows the related states, as well as their current consumption and duration values, along with their corresponding variables used in this study. The values shown in Table 4 are obtained as the average from 10 individual experiments. Initially, the device wakes up from sleep mode (state 1), in order to prepare for the transmission of the advertisements. Then, the device transmits a first advertisement (state 2), followed by an interval during which the frequency channel of the radio is changed (state 3). Using the new channel, a second advertisement transmission is performed (state 4), followed by a second frequency channel change (state 5). Then, the third advertisement transmission is carried out (state 6), followed by an interval where the device turns off the radio (state 7), and a post-processing interval follows (state 8). Then, the device prepares for returning to the sleep state over a cool-down interval (state 9). Note that, in our model, we assume that the LPN will return to sleep state before scanning for incoming messages from the FN in order to save energy, since the *ReceiveDelay* parameter may take values between 10 and 511 ms.

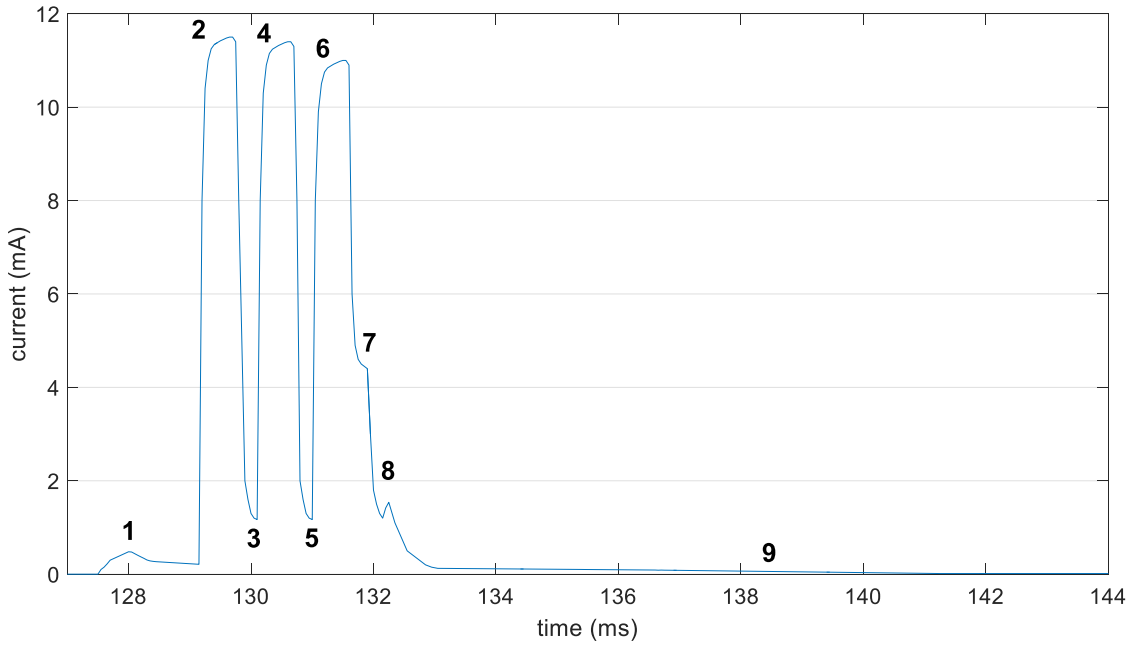


Figure 16: Current consumption of an nRF51422 Development Kit for the different states related with the transmission of three advertisements.

Table 4: Characterization of the states related with the transmission of three advertisements by an nRF51422 Development Kit.

State Number	Description	Duration		Current Consumption	
		Variable	Value (ms)	Variable	Value (mA)
0	Sleep	T_{sleep}	Equation (2)	I_{sleep}	0.015
1	Wake up	$T_{wake\text{-}up}$	1.51	$I_{wake\text{-}up}$	0.38
2	1st transmission	T_{tx1}	0.52	I_{tx1}	8.45
3	1st channel change	$T_{ch\_change\_1}$	0.30	$I_{ch\_change\_1}$	3.66
4	2nd transmission	T_{tx2}	0.53	I_{tx2}	8.85
5	2nd channel change	$T_{ch\_change\_2}$	0.33	$I_{ch\_change\_2}$	3.91
6	3rd transmission	T_{tx3}	0.53	I_{tx3}	8.71
7	Radio off	T_{radio_off}	0.29	I_{radio_off}	5.48
8	Post-processing	T_{post}	0.62	I_{post}	1.2
9	Cool down	I_{cool_down}	14.0	I_{cool_down}	0.074

We next provide the device current consumption profile that corresponds to the scanning action (Figure 17 and Table 5). After the initial sleep state, the device wakes up (state 10) and sets the radio interface in receive mode during the whole scan interval (state 11). Subsequently, the device turns off the radio interface (state 12) and performs a cool-down sequence (state 13) in preparation for returning to sleep mode.

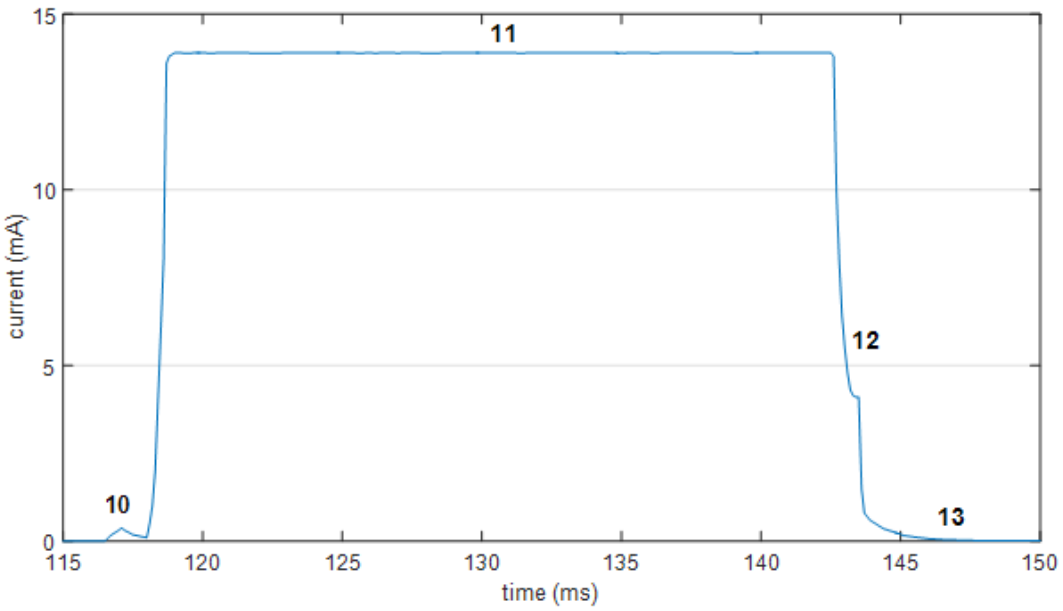


Figure 17: Current consumption of an nRF51422 Development Kit for the states related with performing a scan interval.

Table 5: Characterization of the states related with performing a scan interval by an nRF51422 Development Kit.

State Number	Description	Duration		Current Consumption	
		Variable	Value (ms)	Variable	Value (mA)
10	Wake up pre scan	$T_{wake\text{-}up\_scan}$	1.57	$I_{wake\text{-}up\_scan}$	0.34
11	Scan	T_{scan}	*ReceiveWindow*	I_{scan}	13.9
12	Radio off & processing	T_{radio_off}	0.32	I_{radio_off}	6.44
13	Cool down	T_{post}	26.3	I_{post}	0.07

Based on the profiles of the advertisement transmissions and the scanning interval, we next calculate the LPN average current consumption. Since the latter performs a request-scan cycle every *PollTimeout*, and it transmits a sensor reading every T_{Data}, $I_{average}$ can be obtained as shown next:

$$I_{average} = \frac{1}{PollTimeout}\left(\sum_{i=0}^{13} T_i \cdot I_i + \frac{PollTimeout}{T_{Data}}\sum_{i=1}^{9} T_i \cdot I_i\right) \qquad (3\text{-}1)$$

where I_i and T_i denote respectively the current consumption and the duration of state i in Table 4 and Table 5.

We next determine the average duration of the sleep interval within a *PollTimeout* interval, T_{sleep}. Let T_{act} be the average total duration of all states wherein the device is not in sleep mode. Then, T_{sleep} can be calculated as:

$$T_{sleep} = PollTimeout - T_{act} \qquad (3\text{-}2)$$

where T_{act} can be obtained as shown in the next equation:

$$T_{act} = \sum_{i=1}^{13} T_i + \frac{PollTimeout}{T_{Data}} \sum_{i=1}^{9} T_i \quad (3\text{-}3)$$

As it can be seen, $I_{average}$ can be determined by using equations (3-1)-(3-3). Based on this performance parameter, it is possible to obtain the theoretical lifetime of a LPN, denoted $T_{lifetime}$, assuming a battery capacity of $C_{battery}$ (expressed in mA·h), as shown next:

$$T_{lifetime} = \frac{C_{battery}}{I_{average}} \quad (3\text{-}4)$$

Note that, since the characteristics of a real battery degrade over time, the LPN lifetime calculated above is theoretical. Therefore, the LPN lifetime results provided in the next section can be understood as an upper bound for the achievable lifetime of a real LPN.

Finally, we also model the energy consumed by the LPN per user data bit delivered to a neighbor, $EC_{delivery}$. Let V indicate the battery voltage of the LPN. Let $E[l_{delivery}]$ denote the expected number of user data bits delivered per T_{Data}. We obtain $EC_{delivery}$ as shown in equation (3-5):

$$EC_{delivery} = \frac{I_{average} \cdot V \cdot T_{Data}}{E[l_{delivery}]} \quad (3\text{-}5)$$

where $E[l_{delivery}]$ depends on the Frame Loss Rate (FLR), and on the payload size, denoted $l_{Payload}$. The data message sent by the LPN will be correctly delivered to a next hop if at least one of the corresponding three individual advertisement transmissions that carry the data message is successfully received. Then, assuming that frame losses are uncorrelated, the expected amount of data delivered by the device per transaction is determined as:

$$E[l_{delivery}] = l_{Payload} \cdot (1 - FLR^3) \quad (3\text{-}6)$$

3.3. Evaluation

In this section, we evaluate current consumption, lifetime, and the energy consumed per delivered bit of a battery-operated LPN, by using the models provided in Section 3.1. This section is organized into three different subsections. Each subsection provides evaluation results, along with the corresponding discussion, for each aforementioned energy performance parameter.

3.3.1. LPN Current Consumption

Firstly, we evaluate the average LPN current consumption, based on equations (3-1)-(3-3), as a function of *PollTimeout* and *ReceiveWindow* values that cover the

whole allowed range for these parameters (Figure 18). As a benchmark, results in Figure 18 are obtained under the assumption that data messages are not transmitted.

As shown in Figure 18, the average current consumption decreases with *PollTimeout*, since for large *PollTimeout* values, sleep intervals become dominant. The average current consumption increases with *ReceiveWindow*, since for greater values of this parameter, the LPN radio interface needs to remain in receive mode (thus, consuming a higher amount of current) for a longer time. For example, for *PollTimeout* equal to 10 s, the current consumption ranges from 18.7 μA to 371 μA, for *ReceiveWindow* settings of 1 ms and 255 ms, respectively.

Performance variations for different *ReceiveWindow* values become irrelevant for a *PollTimeout* greater than 10000 seconds, where sleep intervals have significantly greater duration than active ones.

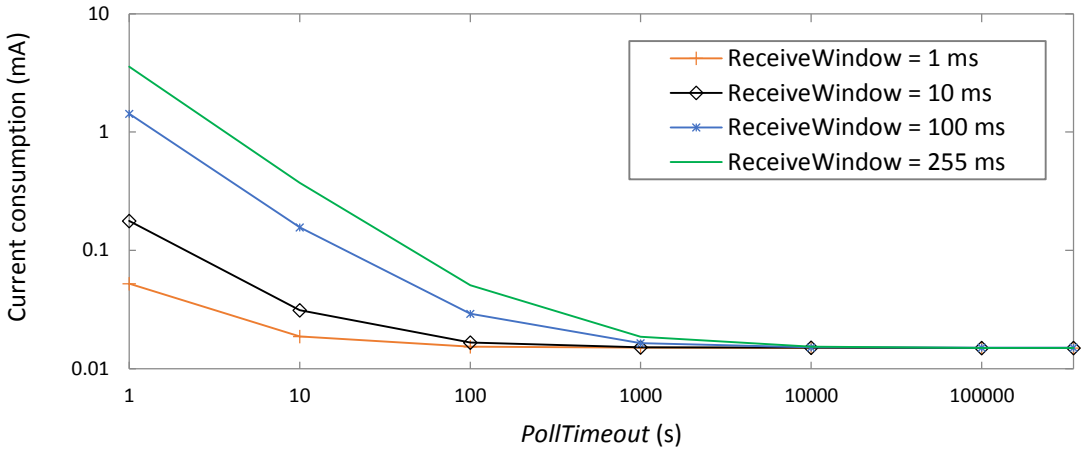


Figure 18: Average current consumption of the LPN, as a function of PollTimeout, in absence of data message transmissions, and for various ReceiveWindow settings.

We next study the impact of the time between two consecutive data message transmissions by the sensor, T_{Data}, on the LPN current consumption. As shown in Figure 19, LPN current consumption decreases with T_{Data}, since increasing T_{Data} reduces the rate at which operations related with data transmission are performed. Such increase grows asymptotically with *PollTimeout*, since with greater values of the latter, data transmission becomes the main activity, other than sleeping, of the LPN.

A significant current consumption increase can be observed for high data message sending rates such as 1 Hz (i.e. T_{Data} =1 s), of a factor up to 2.3 compared with absence of data transmission. However, reducing the data message rate to 0.1 Hz (i.e. T_{Data} =10 s) leads to a current consumption only slightly greater than the one obtained in absence of data transmission.

For high *ReceiveWindow* settings (e.g. *ReceiveWindow* = 255 ms), and for low *PollTimeout* values (e.g. up to 10 s), polling becomes dominant, leading to a high

current consumption (2 orders of magnitude greater than the sleep state current consumption) and rendering the impact of T_{Data} (also referred to as Data Interval) negligible. In that case, data transmission uses the radio interface for a low time, compared with the scan interval duration. Performance differences between the highest and the lowest possible values for *ReceiveWindow* (i.e. 255 ms and 1 ms, respectively), become negligible for *PollTimeout* settings beyond 1000 s. In that region of values, the data transmission rate sets a lower bound on current consumption.

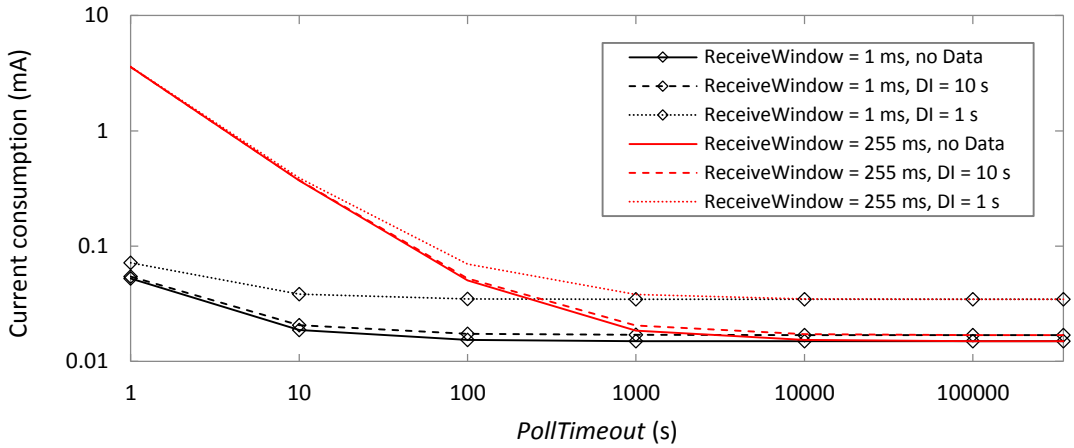


Figure 19: Impact of Data Interval (DI) on the average current consumption of the LPN, for ReceiveWindow settings of 1 ms and 255 ms, as a function of PollTimeout.

3.3.2. LPN Lifetime

We next calculate the theoretical lifetime of a battery-operated LPN, for the same range of scenarios considered in the previous subsection, by using equation (3-4), and the current consumption results obtained in the previous subsection. We assume an ideal battery with a capacity of 235 mAh (e.g. as featured by the prevalent CR2032 button cell battery). Therefore, the calculated LPN lifetime may be lower than that of a real one, as the properties of a real battery degrade over time.

Figure 20 depicts the LPN lifetime, in absence of data message transmission, for the same *PollTimeout* and *ReceiveWindow* settings considered in Figure 18. The theoretical LPN lifetime is inversely proportional to the average current consumption. LPN lifetime grows with *PollTimeout*, with an asymptotic lifetime of 21.8 months that is limited by the sleep state current consumption. For low *PollTimeout* values, the *ReceiveWindow* setting becomes relevant, since active states have relatively significant duration and current consumption compared to sleep intervals. For example, for *PollTimeout* = 10 s, the LPN ranges from 0.87 months to 17.4 months, for *ReceiveWindow* settings of 255 ms and 1 ms, respectively. However, performance differences decrease with *PollTimeout*, with LPN lifetime results exhibiting negligible differences for *PollTimeout* values greater than 10000 seconds.

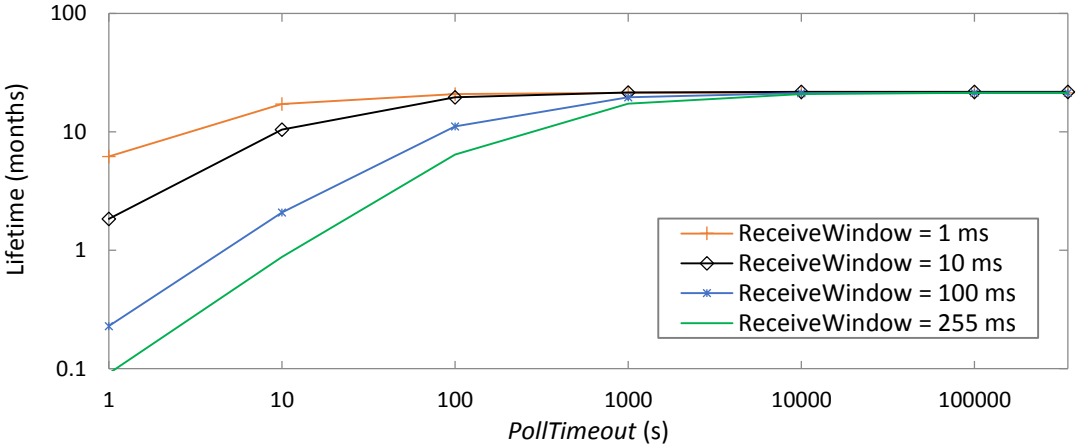


Figure 20: LPN lifetime, as a function of PollTimeout, in absence of data message transmissions, and for different ReceiveWindow settings.

As expected, when the battery-operated LPN also transmits data messages, its lifetime decreases (Figure 21). Impact of data transmission is low; for example, there is up to a maximum of 13% lifetime decrease for a relatively high rate of data message transmission such as 0.1 Hz (i.e. T_{Data} =10 s). The reason is the low contribution of data transmission to current consumption, compared with polling, scanning, and the sleep intervals. Very high data message rates like 1 Hz (i.e. T_{Data} =1 s) produce a significant lifetime decrease, by a factor up to 2.3, compared with a LPN that does not transmit data messages.

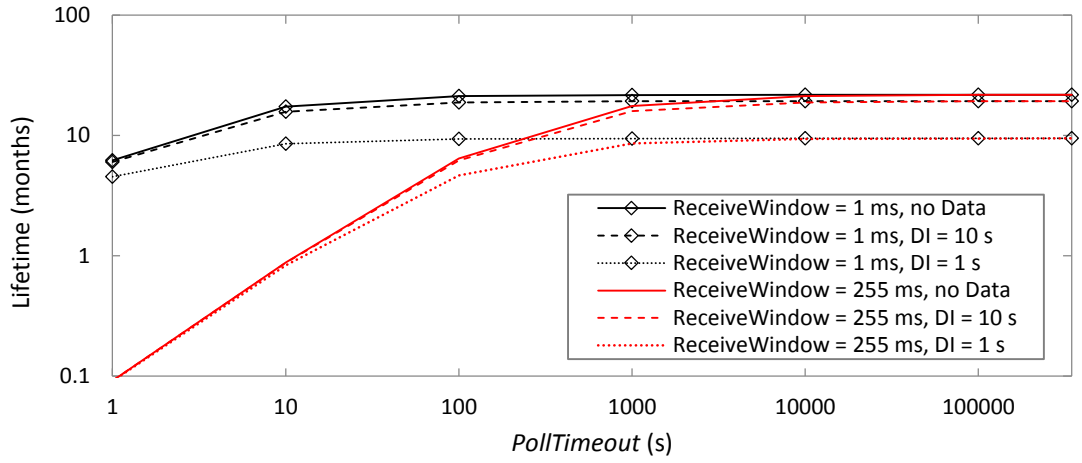


Figure 21: Impact of Data Interval on the LPN lifetime, as a function of PollTimeout, for ReceiveWindow values of 1 ms and 255 ms, and as a function of PollTimeout.

Impact of data transmission is greater for low *ReceiveWindow* settings, since in this case current consumption of data message transmission is relevant compared to that of polling and scanning, whereas sleep intervals are not dominant. For example, for

PollTimeout=10 s, setting T_{Data} to 1 s decreases LPN lifetime by a factor of 2.04 and 1.05, for *ReceiveWindow* settings of 255 ms and 1 ms, respectively, compared with lack of data message transmission.

3.3.3. Energy Consumed per Delivered Bit

In this subsection, we determine the energy consumed per delivered bit, $EC_{delivery}$, for a battery-operated LPN, by using equations (3-5) and (3-6). Figure 22 and 23 illustrate the results obtained as a function of *PollTimeout*, for different *ReceiveWindow* and T_{Data} settings, and for different FLR values. We assume a data message payload size of 20 bytes.

For given T_{Data} and FLR values, $EC_{deliver}$ exhibits a behavior with *PollTimeout* that is similar to that of current consumption, that is, an asymptotical decrease with this parameter (see Figure 18). However, $EC_{deliver}$ increases with T_{Data}. Note that, within T_{Data}, the energy consumed during sleep intervals accumulates over a time close to T_{Data}. Such an increase is greater for low *PollTimeout* values, where polling and scanning operations become relevant and sleep intervals are shorter. For example, for *PollTimeout* = 1 s, $EC_{deliver}$ for T_{Data} = 10 s is 7.5 times greater than that obtained for T_{Data} = 1 s, while as *PollTimeout* increases, the $EC_{deliver}$ difference between using the same respective T_{Data} settings tends to a factor of 4.9.

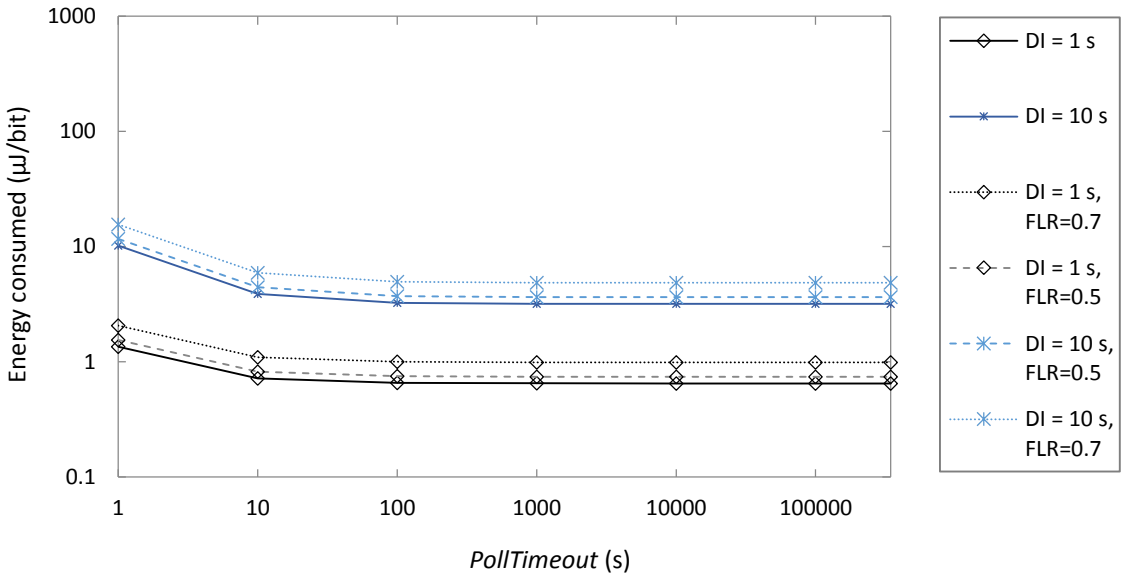


Figure 22: Energy consumed per delivered data bit, as a function of PollTimeout, for ReceiveWindow = 1 ms and different Data Interval and FLR values.

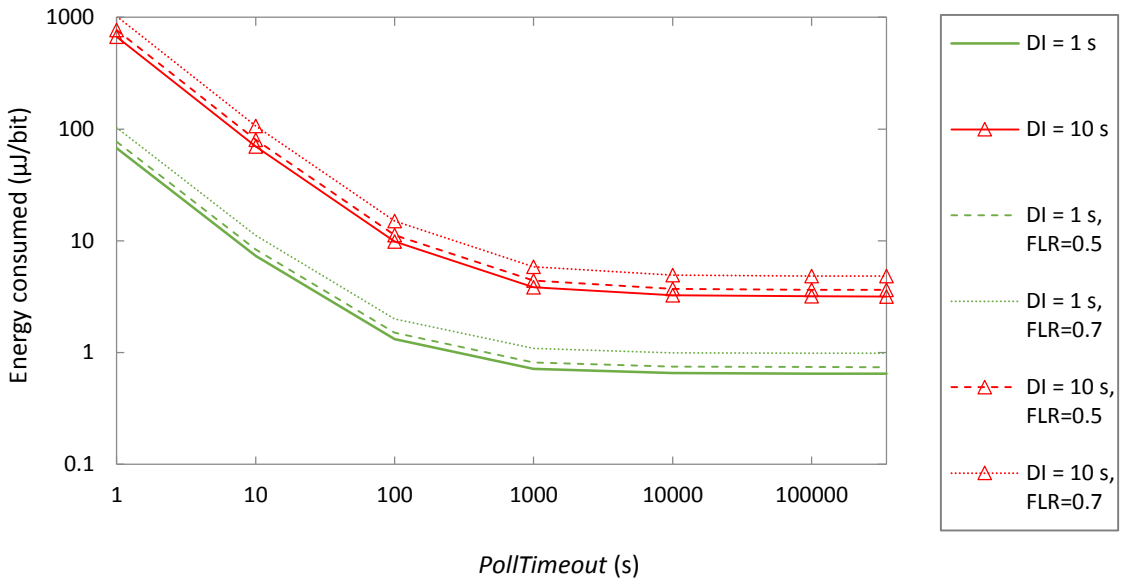


Figure 23: Energy consumed per delivered data bit, as a function of PollTimeout, for ReceiveWindow = 255ms and different Data Interval and FLR values.

A non-zero FLR increases $EC_{deliver}$, since energy is consumed in all messages transmitted, however, a subset of them are not delivered. Because a message is transmitted by sending three advertisements, the energy consumed increases significantly only for relatively high FLR values. For example, FLR=0.5 leads to an $EC_{deliver}$ increase of 14%. For FLR values beyond 0.5, the $EC_{deliver}$ increase grows quickly (e.g. FLR=0.7 yields an $EC_{deliver}$ increase of 52%).

3.4. Conclusions: Bluetooth Mesh energy modeling

In this chapter, we have presented analytical models on important energy performance parameters of a battery-operated LPN, such as current consumption, lifetime and energy consumed per delivered bit. The models take into account the influence of Bluetooth Mesh parameters, such as *ReceiveWindow* and *PollTimeout*, as well as the data message rate. We also evaluate the influence of losses on the energy consumed per delivered bit. We have assumed a LPN that corresponds to a sensor device that periodically transmits data messages.

LPN current consumption decreases with *PollTimeout*, and increases with *ReceiveWindow*. For high *PollTimeout* values, using different *ReceiveWindow* settings leads to negligible performance differences. Remarkably, the data message transmission rate is irrelevant in terms of current consumption, except for very high sending rates in the order of 1 Hz or greater.

Assuming a 235 mAh battery, the asymptotic lifetime for the considered LPN hardware platform is 21.8 months. For low *PollTimeout* values, the *ReceiveWindow* setting becomes

relevant, due to the energy consumption of the LPN during active states, compared to that of sleep intervals. Data transmission reduces LPN lifetime to a greater extent for low *ReceiveWindow* settings, due to the lower impact of sleep intervals.

The energy consumed by the LPN by each delivered bit increases with T_{Data}, as in fact the energy consumed during sleep intervals is significant. Message losses increase the energy cost of data delivery. However, since data transmission is carried out over 3 advertisements, such an increase is only significant for very high FLR (e.g. the increase is greater than 10% for an FLR beyond 0.46).

For a given Bluetooth Mesh network scenario, a suitable parameter configuration will need to take into account energy performance aspects (e.g. by using the models presented in this paper), as well as specific characteristics of the scenario and the intended application requirements.

4. IPv6 Mesh over Bluetooth Low Energy

This chapter presents and evaluates IPv6 Mesh over Bluetooth Low Energy (6BLEMesh). Section 4.1 introduces the background concept of 6LoWPAN. Section 4.2 provides the motivation and main concepts of 6BLEMesh. Section 4.3 describes the 6BLEMesh main features. In section 4.4, the connectivity of such network is analytically studied and evaluated. In section 4.5, we present the prototypes we have implemented to validate the 6BLEMesh functionality, and we perform a 6BLEMesh evaluation.

4.1. 6LoWPAN

IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) is an adaptation layer that was originally designed to efficiently enable IPv6 over IEEE 802.15.4 networks [70]. Like BLE networks, IEEE 802.15.4 networks typically comprise resource-constrained devices, and offer relatively low bit rates. IEEE 802.15.4 networks are fundamentally different from the resource-rich networking environments assumed for IPv6 when it was created. In fact, an adaptation layer between the IPv6 layer and the IEEE 802.15.4 layer is required to comply with IPv6 requirements, and for efficiency.

6LoWPAN comprises three fundamental mechanisms: i) compression of IPv6 and UDP headers, ii) optimized IPv6 Neighbor Discovery (ND), and iii) fragmentation functionality. The first two mechanisms allow energy- and bandwidth-frugal operation. 6LoWPAN header compression exploits intra-packet redundancy and an expectation of typically

used header field values. 6LoWPAN-optimized IPv6 ND reduces use of multicast and allows energy conservation intervals by enforcing interactions initiated by energy-constrained devices. 6LoWPAN fragmentation supports the transmission of 1280-byte packets (as required for IPv6) over the smaller maximum frame payload size of IEEE 802.15.4, of ~100 bytes.

IEEE 802.15.4 supports the mesh network topology. Accordingly, 6LoWPAN defines three node roles for such topology: i) 6LoWPAN Border Router (6LBR) for routers at the edge of the 6LoWPAN network, ii) 6LoWPAN Router (6LR) for routers internal to the 6LoWPAN network, and iii) 6LoWPAN Node (6LN) for non-routing devices. A 6LBR often supports several network interfaces, typically including one that offers Internet connectivity. A 6LBR also manages the configuration of a 6LoWPAN network. 6LRs typically support only one network interface and enable the connectivity between 6LNs and the 6LBR. Since 6LBRs and 6LRs need to generally be ready to receive (and forward) data packets, they often require mains power. 6LNs are typically simple devices that run on limited energy sources.

4.2. Introduction to 6BLEMesh

In order to expand the IoT capillarity and its range of supported technologies, the IETF published the "IPv6 over Bluetooth Low Energy" specification (RFC 7668) in late 2015 [71]. That specification adapted 6LoWPAN [72] in order to support IPv6 over BLE networks, however considering only the star topology [73].

With the aim to extend the functionality of RFC 7668 to enable IPv6 over BLE mesh networks, we have created a new draft specification via the IETF IPv6 over Networks of Resource-constrained nodes (6Lo) working group [74]. This draft specification, which we call 6BLEMesh, assumes that there exist Link Layer connections between a node and its neighbors, through which IPv6 packets can be exchanged. Such connections are established by means of the Internet Protocol Support Profile (IPSP) [75]. In 6BLEMesh, as in RFC 7668, a 6LoWPAN-based adaptation layer is set below IPv6 and atop L2CAP. Such adaptation layer provides IPv6 and UDP header compression (which improves communication efficiency), and optimized IPv6 neighbor discovery (which offers network configuration suitable for constrained devices), both adapted for BLE mesh topologies. A routing protocol is assumed to find paths for communication between end devices. Routing is performed at the IPv6 layer, although the routing protocol to be used is not determined by 6BLEMesh.

As mentioned in section 2.2.1, Bluetooth 4.0 only supports BLE networks that follow the star topology. In consequence, RFC 7668 was specifically developed and optimized for that type of network topology. However, subsequent Bluetooth specifications allow the formation of extended topologies, such as the mesh topology. The functionality described in RFC 7668 is not sufficient and would fail to enable IPv6 over mesh networks composed of BLE links.

The IPSP enables discovery of IP-enabled devices and the establishment of a link layer connection for transporting IPv6 packets. The IPSP defines the Node and Router roles for devices that consume/originate IPv6 packets and for devices that can route IPv6 packets, respectively. Consistently with Bluetooth 4.1 and subsequent Bluetooth versions (e.g. Bluetooth 4.2 or subsequent), a device may implement both roles simultaneously.

We assume a mesh network composed of BLE links, where Link Layer connections are established between neighboring IPv6-enabled devices (see Section 4.3.3.2). The IPv6 forwarding devices of the mesh have to implement both Node and Router roles, while simpler leaf-only nodes can implement only the Node role. In an IPv6 mesh over BLE links, a node is a neighbor of another node, and vice versa, if a link layer connection has been established between both by using the IPSP functionality for discovery and link layer connection establishment for IPv6 packet transport.

4.3. 6BLEMesh main features

In this section, we describe the 6BLEMesh specification in terms of protocol stack, subnet model, link model, and security considerations.

4.3.1. Protocol Stack

Figure 24 illustrates the protocol stack for 6BLEMesh. There are two main differences with the IPv6 over Bluetooth LE stack in RFC 7668: a) the adaptation layer below IPv6 (labelled as "6Lo for mesh over Bluetooth LE") is now adapted for mesh networks of Bluetooth LE links, and b) the protocol stack for IPv6 mesh networks of Bluetooth LE links includes IPv6 routing functionality.

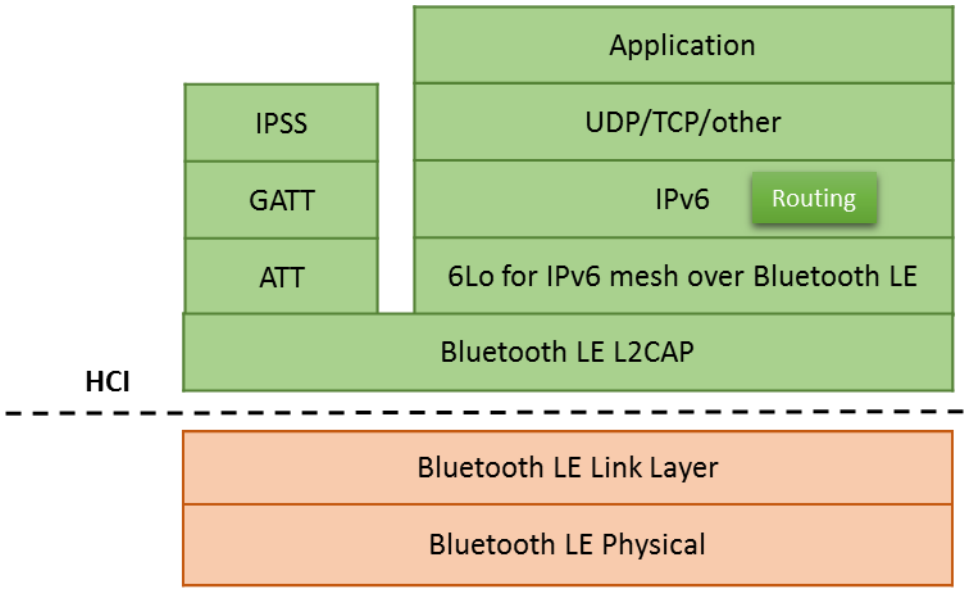


Figure 24: 6BLEMesh protocol stack.

4.3.2. Subnet Model

For 6BLEMesh, a multilink model has been chosen, as further illustrated in Figure 25. As 6BLEMesh is intended for constrained nodes, and for Internet of Things use cases and environments, the complexity of implementing a separate subnet on each peripheral-central link and routing between the subnets appears to be excessive. The benefits of treating the collection of point-to-point links between a central and its connected peripherals as a single multilink subnet rather than a multiplicity of separate subnets are considered to outweigh the multilink model's drawbacks as described in [76].

One or more 6LBRs are connected to the Internet. 6LNs are connected to the network through a 6LR or a 6LBR. A prefix is used on the whole subnet. IPv6 mesh networks over BLE must follow a route-over approach. We do not specify the routing protocol to be used in an IPv6 mesh over BLE, although RPL appears to be the main candidate routing protocol for 6BLEMesh, since it is the routing protocol standardized by the IETF for IoT environments [77]. Nevertheless, other routing protocols have been selected for some IP-based IoT protocol stacks. For example, Thread uses a solution based on the algorithm used in the RIP routing protocol [62,78].

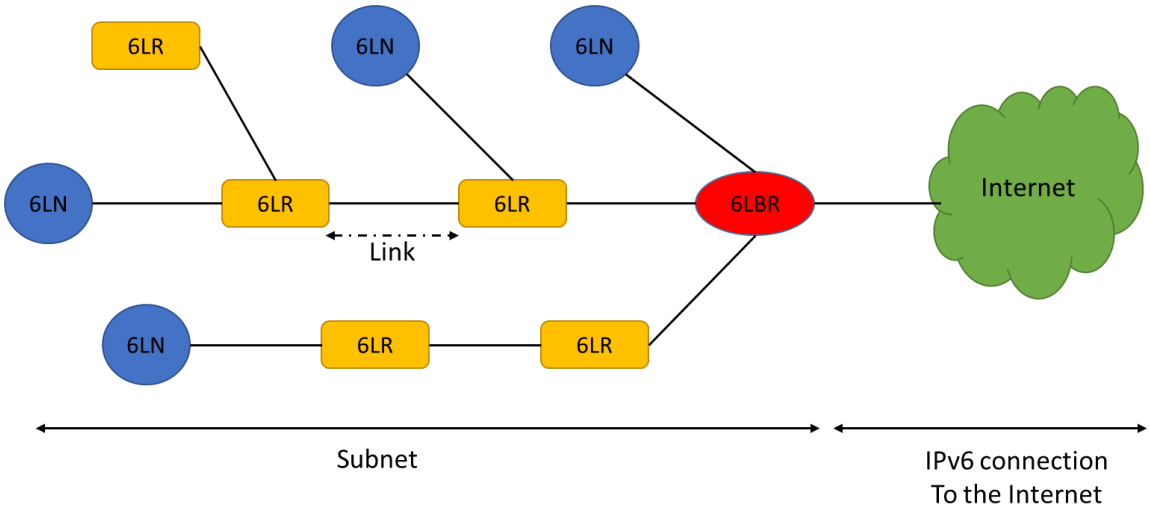


Figure 25: Example of a 6BLEMesh network.

4.3.3. Link Model

In this subsection, we describe stateless address autoconfiguration, ND, header compression and unicast and multicast mapping in 6BLEMesh.

4.3.3.1. Stateless address autoconfiguration

6LN, 6LR and 6LBR IPv6 addresses in an IPv6 mesh over Bluetooth LE are configured as per section 3.2.2 of RFC 7668. Multihop Duplicated Address Detection

(DAD) functionality as defined in section 8.2 of RFC 6775, or some substitute mechanism (see section 4.3.3.2), must be supported.

4.3.3.2. Neighbor Discovery

Neighbor Discovery Optimization for 6LoWPAN, i.e. RFC 6775, describes the IPv6 Neighbor Discovery approach as adapted for use in several 6LoWPAN topologies, including the mesh topology [79]. The route-over functionality of RFC 6775 must be supported in 6BLEMesh. The following aspects of the Neighbor Discovery optimizations for 6LoWPAN are applicable to 6BLEMesh 6LNs:

1. A 6BLEMesh 6LN must not register its link-local address. A 6BLEMEsh 6LN must register its non-link-local addresses with its routers by sending a Neighbor Solicitation (NS) message with the Address Registration Option (ARO) and process the Neighbor Advertisement (NA) accordingly. The NS with the ARO option must be sent irrespective of the method used to generate the IPv6 address Interface Identifier (IID). The ARO option requires use of an EUI-64 identifier [79]. In the case of Bluetooth LE, the field shall be filled with the 48-bit device address used by the BLE node converted into 64-bit Modified EUI-64 format [80]. If the 6LN registers for a same compression context multiple addresses that are not based on Bluetooth device address, the header compression efficiency will decrease.

2. For sending Router Solicitations and processing Router Advertisements the BLE 6LNs must, respectively, follow Sections 5.3 and 5.4 of RFC 6775.

3. The router behavior for 6LRs and 6LBRs is described in Section 6 of RFC 6775. However, as per this specification, routers shall not use multicast NSs to discover other routers' link layer addresses.

4. Border router behavior is described in Section 7 of RFC 6775. RFC 6775 defines substitutable mechanisms for distributing prefixes and context information (section 8.1 of RFC 6775), as well as for DAD across a route-over 6LoWPAN (section 8.2 of RFC 6775). Implementations of this specification must support the features described in sections 8.1 and 8.2 of RFC 6775 unless some alternative ("substitute") from some other specification is supported.

4.3.3.3. Header compression

Header compression as defined in RFC 6282 [81], which specifies the compression format for IPv6 datagrams on top of IEEE 802.15.4, is required as the basis for IPv6 header compression on top of BLE. All headers must be compressed according to RFC 6282 encoding formats.

To enable efficient header compression, when the 6LBR sends a Router Advertisement it must include a 6LoWPAN Context Option (6CO) matching each

address prefix advertised via a Prefix Information Option (PIO) [82] for use in stateless address autoconfiguration.

The specific optimizations of RFC 7668 for header compression, which exploit the star topology and ARO, cannot be generalized in a mesh network composed of BLE links. Still, a subset of those optimizations can be applied in some cases in such a network. In particular, the latter comprise link-local interactions, non-link-local packet transmissions originated and performed by a 6LN, and non-link-local packet transmissions originated by a 6LN neighbor and sent to a 6LN. For the rest of packet transmissions, context-based compression may be used. When a device transmits a packet to a neighbor, the sender must fully elide the source IID if the source IPv6 address is the link-local address based on the sender's Bluetooth device address. The sender also must fully elide the destination IPv6 address if it is the link-local-address based on the neighbor's Bluetooth device address. When a 6LN transmits a packet, with a non-link-local source address that the 6LN has registered with ARO in the next-hop router for the indicated prefix, the source address must be fully elided if it is the latest address that the 6LN has registered for the indicated prefix. If the source non-link-local address is not the latest registered by the 6LN, then the 64-bits of the IID shall be fully carried in-line or if the first 48-bits of the IID match with the latest address registered by the 6LN, then the last 16-bits of the IID shall be carried in-line.

When a router transmits a packet to a neighboring 6LN, with a non-link-local destination address, the router must fully elide the destination IPv6 address if the destination address is the latest registered by the 6LN with ARO for the indicated context. If the destination address is a non-link-local address and not the latest registered, then the 6LN must either include the IID part fully in-line or, if the first 48 bits of the IID match to the latest registered address, then elide those 48 bits.

4.3.3.4. Unicast and multicast mapping

The BLE Link Layer does not support multicast. Hence, traffic is always unicast between two BLE neighboring nodes. If a node needs to send a multicast packet to several neighbors, it has to replicate the packet and unicast it on each link. However, this may not be energy efficient, and particular care must be taken if the node is battery powered. A router (i.e. a 6LR or a 6LBR) must keep track of neighboring multicast listeners, and it must not forward multicast packets to neighbors that have not registered as listeners for multicast groups the packets belong to.

4.3.4. Security Considerations

We have followed the security considerations in RFC 7668. As mentioned above, IPv6 mesh networks over Bluetooth LE require a routing protocol to find end-to-end paths. Unfortunately, the routing protocol may generate additional opportunities for threats and attacks to the network. RFC 7416 provides a systematic overview of

threats and attacks on the RPL [83], as well as countermeasures. In that document, described threats and attacks comprise threats due to failures to authenticate, threats due to failure to keep routing information, threats and attacks on integrity, and threats and attacks on availability. Reported countermeasures comprise confidentiality attack, integrity attack, and availability attack countermeasures. While this document does not state the routing protocol to be used in 6BLEMesh, the guidance of RFC 7416 is useful when RPL is used in such scenarios. Furthermore, such guidance may partly apply for other routing protocols as well.

4.4. Connectivity study

In this section, we provide an analytical model and a study of two crucial connectivity properties of 6BLEMesh: the probability of a node being not isolated, and network *K*-connectivity [84]. The model, which is validated by simulation, relates the aforementioned connectivity properties with the scenario area, the number of nodes, their coverage area, and the number of timeslots (i.e. a fundamental 6BLEMesh parameter, denoted N_{Slot}). The main novelty of the model lies in capturing the impact of N_{Slot} on the two considered connectivity properties (see Section 4.4.1 and 4.4.2). In fact, the N_{Slot} setting controls a crucial trade-off between link capacity and connectivity in 6BLEMesh. N_{Slot} can thus be tuned for the specific features and requirements of a given 6BLEMesh. Therefore, the model will be a useful tool for the planning and evaluation of 6BLEMesh.

Connectivity is a fundamental property of any wireless mesh network. In this section, we provide an analytical model for two important connectivity-related parameters of 6BLEMesh. The first one is the probability that a node can establish a link with at least one neighbor, and thus become non-isolated (i.e. probability of no isolation, denoted by P_{NI}). This parameter is crucial since an isolated node does not contribute to the network and cannot communicate with any other node. Note that two physical neighbors (i.e. nodes that are placed within the coverage range of each other) will only be able to establish a link if both of them have sufficient available timeslots. The second target parameter is the probability that a network is *K*-connected, that is, the probability that any node in the network can communicate with any other node through *K* mutually independent paths. This probability is denoted by $P_{K\text{-}con}$. In other words, a mesh network is *K*-connected if any node can reach any other node even if any group of *K*-1 nodes are dropped from the network [85]. This property expresses robustness of a mesh network in the presence of node or link failures, which may be due to battery depletion, device malfunctioning, interference, etc.

In order to determine P_{NI} and $P_{K\text{-}con}$, we assume that nodes follow a uniformly random spatial distribution over a two-dimensional area. The node spatial distribution assumed captures the characteristics of many real scenarios (e.g. smart homes), where nodes are

not deployed under a predetermined, regular pattern. For more structured scenarios (e.g. grid topologies), we recommend using ad-hoc developed models.

We use the following input parameters for our model: the area where nodes are distributed (denoted A), the total number of nodes in the network (denoted N), the coverage area of a node (denoted a), and N_{Slot}. For the analysis, we assume that all network nodes are homogeneously configured with the same value for N_{Slot}, *connInterval*, and timeslot duration.

The next two subsections provide the analytical models for calculating P_{NI} and $P_{K\text{-}con}$, respectively.

4.4.1. Probability of no isolation

In order to determine P_{NI}, we first calculate the probability that a node is placed in the coverage range of another node, denoted q, which can be obtained as $q=a/A$.

Let d_{phy} be the number of physical neighbors of a node. P_{NI} can be determined as the probability that a node can connect with at least one of its physical neighbors (i.e. the node and at least one of its physical neighbors have an available timeslot for communication). P_{NI} can be thus expressed as per (4-1):

$$P_{NI} = \sum_{j=1}^{N-1} P\left(d_{phy} = j\right) \cdot P\left(conn | d_{phy} = j\right) \tag{4-1}$$

where $P(d_{phy}=j)$ denotes the probability of a node having j physical neighbors, and $P(conn|d_{phy}=j)$ is the probability that the node is not isolated when the node has j physical neighbors. These terms are calculated through (4-2) and (4-3). In the latter, n_{conn} and n_{iso} denote the number of combinations in which a node with j physical neighbors is connected, and isolated, respectively, and are obtained through (4-4) and (4-5).

$$P\left(d_{phy} = j\right) = \binom{N-1}{j} \cdot q^i \cdot (1-q)^{N-1-j} \tag{4-2}$$

$$P\left(conn | d_{phy} = j\right) = \frac{n_{conn}}{n_{conn} + n_{iso}} \tag{4-3}$$

$$n_{conn} = \sum_{h=1}^{N_{Slot}} \binom{j}{h} \cdot P\left(conn | d_{phy} = 1\right)^h \tag{4-4}$$

$$n_{iso} = \left(1 - p\left(conn \middle| d_{phy} = 1\right)\right)^j \tag{4-5}$$

$P(conn|d_{phy}=1)$, that is, the probability of a node x being connected to its only physical neighbor y, can be computed by adding two different terms, U and V (see (4-

6)), because the calculation has to be done differently when y has less than N_{Slot} physical neighbors (see (4-7)) from when it has at least N_{Slot} physical neighbors (see (4-8)), respectively. In the first case, there exist available timeslots for establishing Link Layer connections between node y and all its own physical neighbors. However, in the second case, y can only establish Link Layer connections with up to N_{Slot} of its physical neighbors. We assume that all neighbors have the same priority of becoming connected with a node. Then, $P(conn|d_{phy}=1)$ can be obtained by using (4-6)-(4-9).

$$p\left(conn\middle|d_{phy}=1\right)=U+V \tag{4-6}$$

$$U=\sum_{i=1}^{N_{Slot}-1}\binom{N-2}{i}\cdot q^{i}\cdot(1-q)^{N-2-i} \tag{4-7}$$

$$V=\sum_{i=N_{Slot}}^{N-2}\beta(i)\cdot\binom{N-2}{i}\cdot q^{i}\cdot(1-q)^{N-2-i} \tag{4-8}$$

In (4-8), parameter $\beta(i)$ denotes the probability that x can establish a connection with y. We next describe how parameter $\beta(i)$ is obtained. Assume that the number of physical neighbors of y is at least N_{Slot}, and y has N_{Slot} free timeslots. There may be up to N_{Slot} different opportunities for x to establish a connection with y. Each time a physical neighbor of y connects to y, it reserves one of y's free timeslots. However, other timeslots remain free, and x has other chances to connect to y, except when N_{Slot} nodes have connected with y. Let $\beta(i)_r$ denote the probability of establishing a connection between x and y when y has r connected neighbors, and thus N_{Slot} -r timeslots are available for connecting with y. Then, $\beta(i)$ can be approximated by using (4-9), where $\beta(i)_0$=0.

$$\beta(i)=\sum_{r=1}^{N_{Slot}}(1-\beta(i)_{r-1})\cdot\frac{(N_{Slot}-r-1)}{(i-r)} \tag{4-9}$$

Finally, P_{NI} can be obtained by plugging (4-2)-(4-9) into (4-1).

4.4.2. Probability of *K*-connected network

This section provides the analytical model for calculating the probability that a network is K-connected. Let d denote the degree of a node, i.e. the number of simultaneous connections a node has with its corresponding neighbors (note that for a given node, $d \le d_{phy}$). Let d_{min} be the minimum node degree among the nodes in the network. Based on a property of geometric random graphs [86], if the graph has a high number of nodes, the probability that a network is K-connected, $P_{K\text{-}con}$, can be obtained by using (4-10). An assumption in (4-10) is absence of a limit on the number of neighbors for a node, whereas in 6BLEMesh, a node may have a maximum of N_{Slot}

connected neighbors. However, the resulting model is accurate for practical N_{Slot} values, i.e. the range of N_{Slot} values that enable an almost 1-connected 6BLEMesh (see Section 4.4.3.2).

$$P_{K-con} = P(d_{min} \geq K) \tag{4-10}$$

Note that $d_{min} \geq K$ requires all nodes to have at least K neighbors. On the other hand, the maximum number of connected neighbors a node can have is N_{Slot}. Therefore, (4-10) can be developed as shown in (4-11).

$$P(d_{min} \geq K) = P(d \geq K)^N = \left(\sum_{i=K}^{N_{Slot}} P(d = i) \right)^N \tag{4-11}$$

The term $P(d=i)$ in (4-11) can be obtained through (4-12), which is further developed in (4-13).

$$P(d = i) = \sum_{j=i}^{N-1} P(d_{phy} = j) \cdot P(d = i | d_{phy} = j) \tag{4-12}$$

where $P(d_{phy}=j)$ denotes the probability of a node having j physical neighbors, which can be obtained as per (4-2), and $P(d=i|d_{phy}=j)$ is the probability that the node has i connected neighbors out of j physical neighbors. Accordingly, the calculation of $P(d=i)$ is carried out in (4-13),

$$\begin{aligned} P(d = i) = \sum_{j=i}^{N-1} \binom{N-1}{j} \cdot q^j \cdot (1-q)^{N-1-j} \cdot \\ \cdot \binom{j}{i} \cdot P(d = 1)^i \cdot (1 - P(d = 1))^{j-i} \end{aligned} \tag{4-13}$$

where $P(d=1)$ can be approximated by $P(conn|dphy=1)$, and it can be computed by using (4-6)-(4-9).

4.4.3. Evaluation

Next we evaluate, and validate by simulation, the analytical models presented in section 4.4. We use Matlab to simulate uniformly random node distributions, and obtain P_{NI} and $P_{K\text{-}con}$ from them. Our simulation code is publicly available [87]. In the simulation, N=100 nodes are distributed randomly in a square area of A=40000 m^2. In order to address the border effect problem, we use the toroidal distance [10].

The simulation is executed for different values of N_{Slot} and average node degree (N_{Deg}). N_{Deg} is the expected number of physical neighbors of a node, which is obtained as $N_{Deg} = q \cdot N$ (thus, $N_{Deg} > 0$). For each specific value of N_{Slot} and N_{Deg}, 1000 different

node distributions are generated. In each distribution, 100 sets of connections between neighboring nodes are randomly established. Thus, 100000 individual simulations have been run for each combination of N_{Slot} and N_{Deg} values.

4.4.3.1. Probability of no isolation

Figure 26 plots P_{NI} analytical and simulation results for N_{Slot} =2 and N_{Slot} =10, and for N_{Deg} between 1 and 15. As shown in the figure, the analytical model is accurate. Differences between analytical and simulation results are due to the assumption that the neighbors of a node are independent. The differences tend to decrease as N_{Slot} increases, since connectivity opportunities then also increase. Note that P_{NI} depends on (4-6), which is obtained as $U + V$. Term V uses $\beta(i)$, which is calculated as an approximation in (4-9). As N_{Slot} increases, the overall influence of V in (4-6) decreases, therefore accuracy of the model improves. As expected, P_{NI} increases with N_{Slot} and N_{Deg} (with stronger impact of the latter). Greater values for both parameters lead to a greater amount of connectivity opportunities between a node and its possible neighbors. A node is almost surely connected to at least one neighbor for N_{Slot} =10 and $N_{Deg} \geq 6$. However, a low N_{Slot} setting does not allow to reach near-one P_{NI} for the range of N_{Deg} values considered. Results for N_{Slot} greater than 10 have not been depicted since further P_{NI} increase is negligible for such N_{Slot} values.

4.4.3.2. Probability of *K*-connected network

Figure 27 and Figure 28 illustrate analytical and simulation K-connectivity results, for N_{Deg} between 1 and 15, and for N_{Slot} values of 10 and 15. As expected, K-connectivity increases with N_{Slot}, and N_{Deg}. Achieving an almost-1-connected network requires $N_{Deg} \geq 10$ (and thus, $N_{Slot} \geq 10$). $N_{Deg} \geq 10$ leads to an asymptotic 1-connectivity increase, while significant benefits in terms of 2- or 3-connectivity are achieved.

Figure 29 depicts the K-connectivity model error, defined as the difference between simulation and analytical results. As shown in Figure 29, the model is accurate for the range of scenarios considered, with a maximum error of 0.03. The main reasons for a non-zero error are two approximations made in the model for the sake of analytical tractability: i) calculating $P(d=i|d_{phy}=j)$ in (4-13) by assuming that the connected neighbors of a node are independent, and ii) the approximation of $P(d=1)$ by $P(conn|d_{phy}=1)$ for (4-13). The model error absolute value decreases for low and high N_{Deg} values, whereby connectivity opportunities are scarce and abundant, respectively, which reduces the impact of the approximations.

The developed analytical model allows to plan or evaluate a 6BLEMesh. If three of the four input parameters of the model (i.e. A, N, a, N_{Slot}) are known, it is possible to determine the fourth one for a given connectivity target. For example, assume a smart home with A=100 m^2, a=60 m^2, and N_{Slot}=10. In order to achieve 95% 1-, 2- or 3-connectivity, N_{Deg} needs to be at least 8, 10 or 14, respectively (Figure 27). Thus, the

number of nodes *N* needed for the deployment is at least 14, 17 or 24, respectively. For the same settings, *N*=12 yields a 1-, 2- or 3-connectivity of 86%, 45% or 4%, respectively.

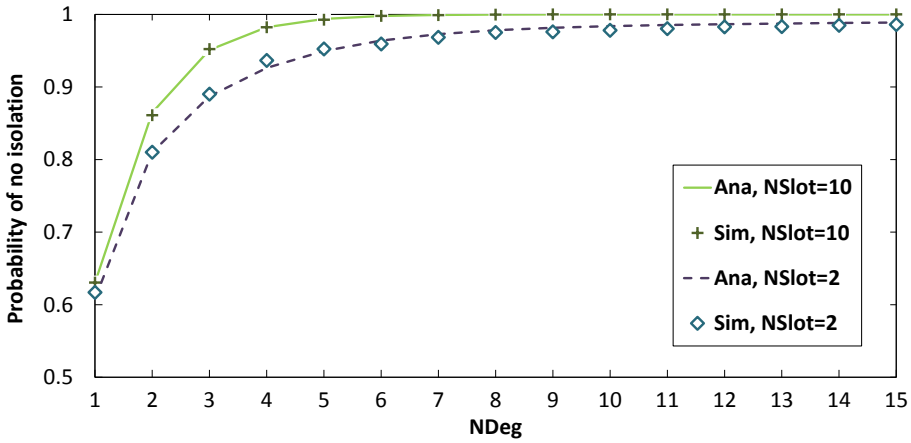


Figure 26: Probability of no isolation for different values of N_{Deg} and N_{Slot}.

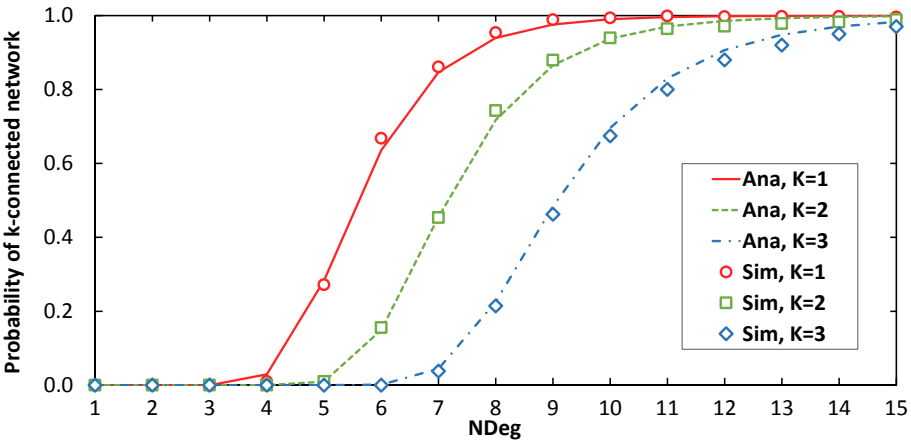


Figure 27: Probability of K-connectivity for N_{Slot} = 10.

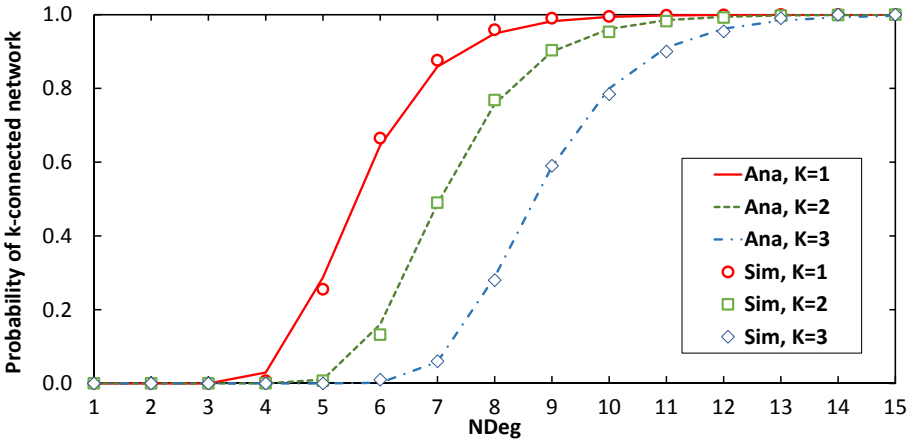


Figure 28: Probability of K-connectivity for N_{Slot} = 15.

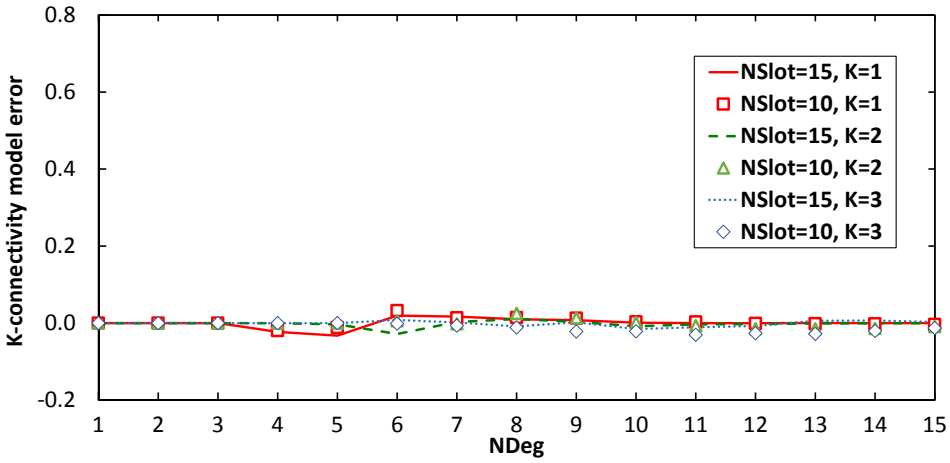


Figure 29: K-connectivity model error as a function of N_{Deg}.

4.5. Prototype implementation and performance evaluation

In order to validate and evaluate 6BLEMesh, we have implemented 6BLEMesh prototypes on a number of popular commercial hardware platforms, namely: Raspberry PI 3 (Figure 30), Nordic Semiconductor nRF51 family (Figure 31), and Texas Instruments CC2650 (Figure 32).

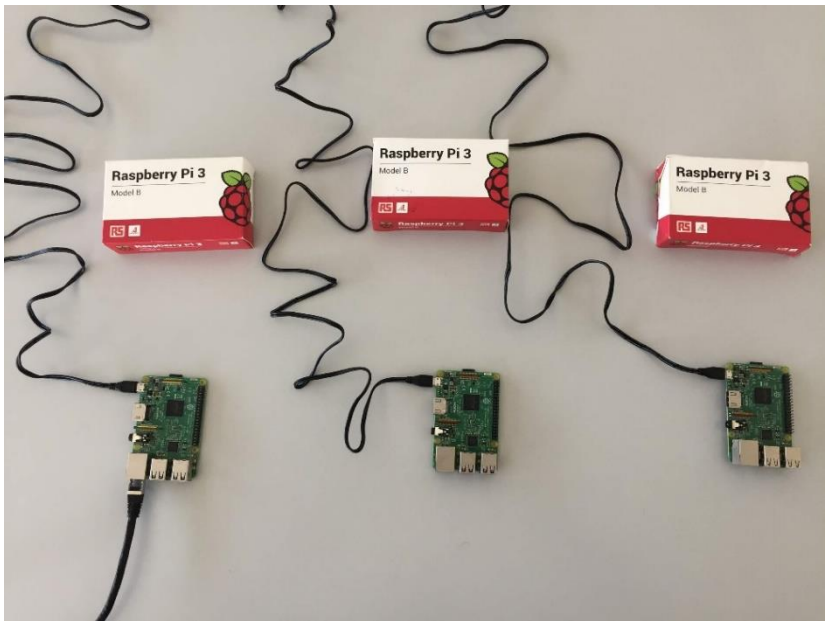


Figure 30: Raspberry Pi 3 devices used for implementing 6BLEMesh.

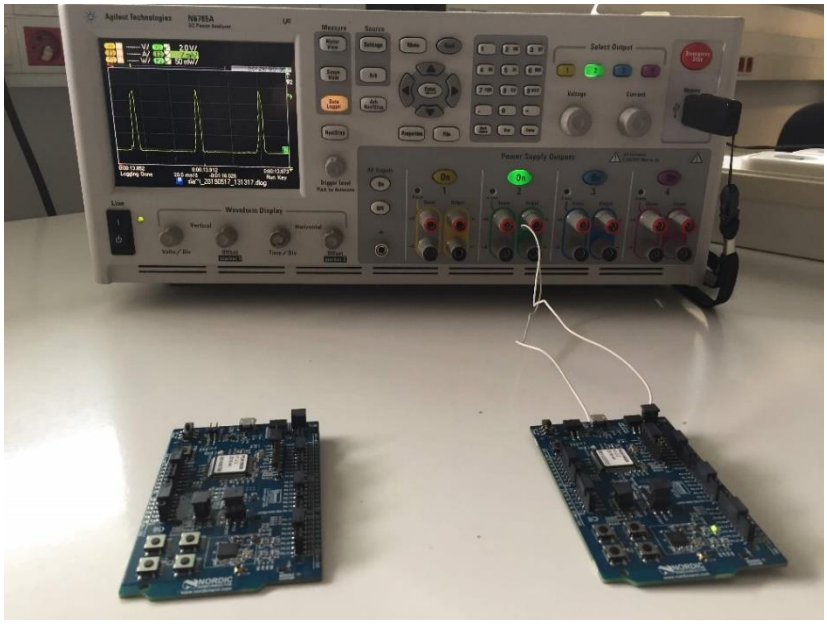


Figure 31: Nordic Semiconductor nRF51 devices used for implementing 6BLEMesh.

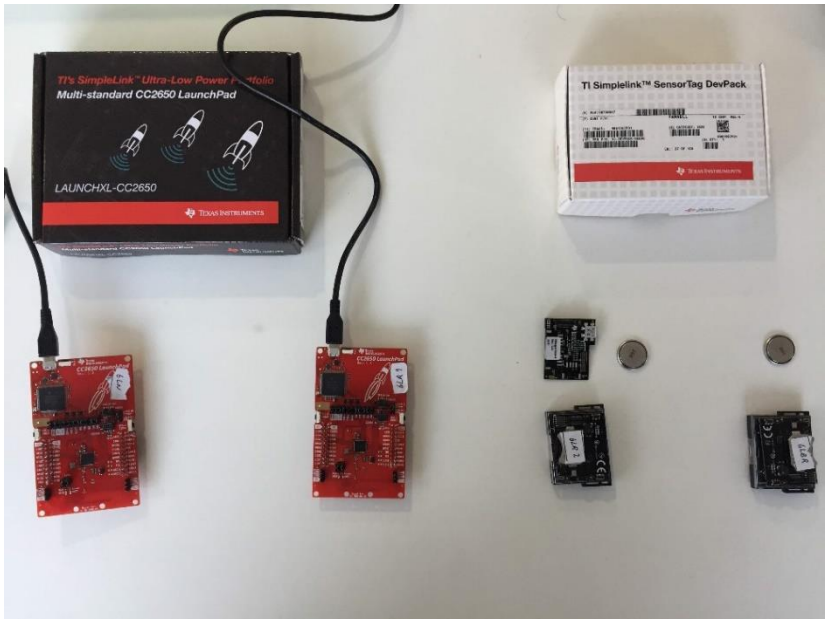


Figure 32: Texas Instrument devices used for implementing 6BLEMesh.

We first implemented 6BLEMesh on Raspberry Pi 3 devices (see Figure 30). Although we implemented all required functionality, we found a driver side issue by which the node was not able to handle more than one BLE connection simultaneously. Then, we turned to use Nordic Semiconductor nRF51 and Texas Instruments CC2650 family, both of which support more than one connection simultaneously.

While we have also evaluated the current consumption of 6LNs on both Nordic Semiconductor nRF51 and Texas Instruments CC2650 devices (see section 4.5.4), we have

used CC2650 LaunchPad (in short, LaunchPad), and CC2650 SensorTag (in short, SensorTag) devices from Texas Instruments to develop a 6BLEMesh prototype testbed (see Figure 32). A LaunchPad device is used as 6LN and 6LR, while SensorTag acts as 6LBR. The topology we used is depicted in Figure 33. Evaluations presented in this section are based on this topology. We have studied one-hop, one-way latency, Round Trip Time (RTT) and energy consumption by means of experiments. Nodes were placed almost 2 meters far from each other, using Bluetooth 4.1, and a transmit power of 0 dBm. We used a packet size that carries an 18-byte user payload, thus it does not exceed 27 bytes at the BLE Link Layer, and it fits into one Physical Layer PDU.

Figure 33: Network topology used in the evaluation of the 6BLEMesh implementation.

4.5.1. Latency

We measure one-hop latency as the time since the command for sending a packet is executed at the sender until the time at which the packet is delivered to a sender's one-hop neighbor. The situation is different when the sender is also the packet source or the sender just relays the packet. Both situations are considered in this subsection where, regarding the former, we studied the latency from the 6LN to the 6LR and, regarding the latter, we have evaluated the latency between the 6LR and the 6LBR.

From an analytical point of view, the IPv6 packet may be sent at any time during current interval of *connInterval* duration. Thus, the expected time to actually send a IPv6 packet from the sender node is *connInterval*/2, plus the packet transmission time. However, if occasionally the IPv6 packet cannot be delivered during the current interval, it will be delivered in subsequent connection events.

Figure 34 shows one-hop latency when the sender (the 6LN in Figure 33) is also the IPv6 packet source, with the 6LR as a receiver, for 100 consecutive IPv6 packets, with an inter-packet time of 1 s, and *connInterval* set to 125 ms. As shown in Figure 34, the first packet cannot be sent in the current *connInterval*, and it is actually sent at the beginning of the second one, after 132.813 ms. This time is equal to connInterval plus the CPU clock period of the 6LN (i.e. a LaunchPad device in our testbed), which is 7.813 ms. For each subsequent packet sent, latency decreases from the previous one by one or two clock times, as the actual packet generation and handling is delayed by that time. The time until the next connection event decreases steadily for a cycle of 11-12 packets, until a packet needs to wait again for a full *connInterval* for the next

connection event. The described behavior repeats again thereafter, leading to the sawtooth shape shown in Figure 34.

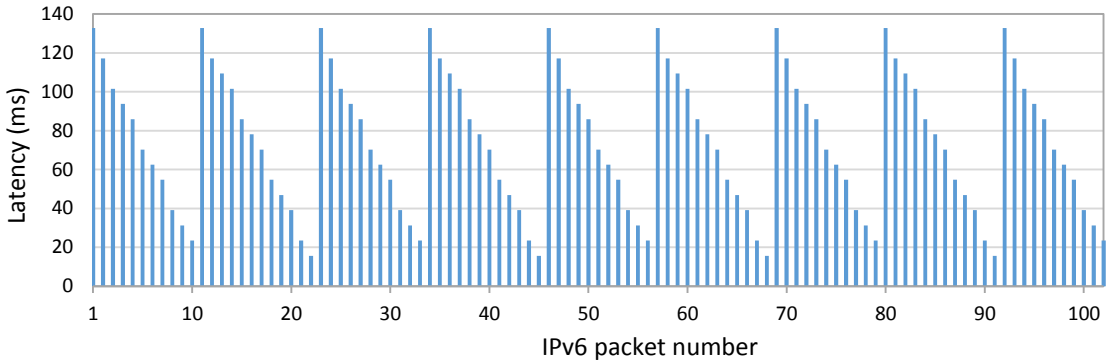


Figure 34: One-hop IPv6 packet delivery latency when the sender (the 6LN in our testbed) is also the packet source (for connInterval = 125 ms).

The previous study was carried out on a network where the sender was also the IPv6 packet source. We next also study the latency between two neighboring nodes where the sender is not the IPv6 packet source; instead, the sender (the 6LR in Figure 33) only receives the IPv6 packet from a previous node, and relays the IPv6 packet to the next node (the 6LBR). Figure 35 illustrates the obtained latency results, for the same *connInterval* setting (i.e. 125 ms) as in the previous experiment.

There are three remarkable observations from Figure 35. First, most IPv6 packets are delivered at the beginning of the next connection interval (i.e. the next connection event), leading to a latency of about 125 ms, in some cases with slight deviations that are a multiple of 7.183 ms (i.e. the CPU clock period). Secondly, there are a few IPv6 packets that are not delivered during the first next connection interval and are transmitted during the subsequent one, duplicating one-hop latency. Thirdly, latency is quite constant in Figure 35 since the 6LR does not need to generate the IPv6 packet. Thus, the received IPv6 packet is typically sent in the same position of the connection interval wherein it is sent.

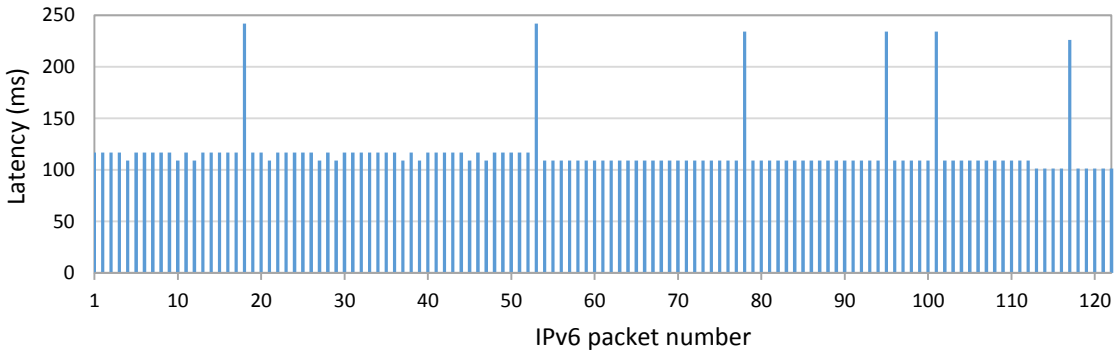


Figure 35: One-hop IPv6 packet delivery latency when the sender (the 6LR in our testbed) only relays the received packet to the 6LBR (for connInterval = 125 ms).

We made additional experiments for other *connInterval* settings (between 50 ms and 4 s), finding similar behaviors as the ones observed for *connInterval*=125 ms.

4.5.2. Round Trip Time measurement

We now measure the RTT on the topology depicted in Figure 33. We run an application on the 6LN that repeatedly sends a message intended for the 6LBR. The 6LR relays packets received from the 6LN to the 6LBR. While typically the 6LBR would route the packet towards the Internet, for our measurement purposes we set the 6LBR to return such messages back to the 6LR immediately. The 6LR forwards again the packets received from the 6LBR back to the 6LN. We have logged the sending time of each packet from the 6LN and the reception time of the same packet by the 6LN as the RTT.

Figure 36 shows the main RTT statistics obtained from the mentioned topology for various *connInterval* settings from 50 ms to 4 s. *connInterval* values lower than 50 ms did not allow stable operation for the devices used in the experiments. Results show that, when *connInterval* is set to 1 s, the minimum and maximum measured RTT values are approximately equal to 2 s and 4 s, respectively. These results correspond to 2 and 4 *connInterval* periods, respectively, which is consistent with the fact that one-hop and one-way latency results can take values from almost zero up to *connInterval*.

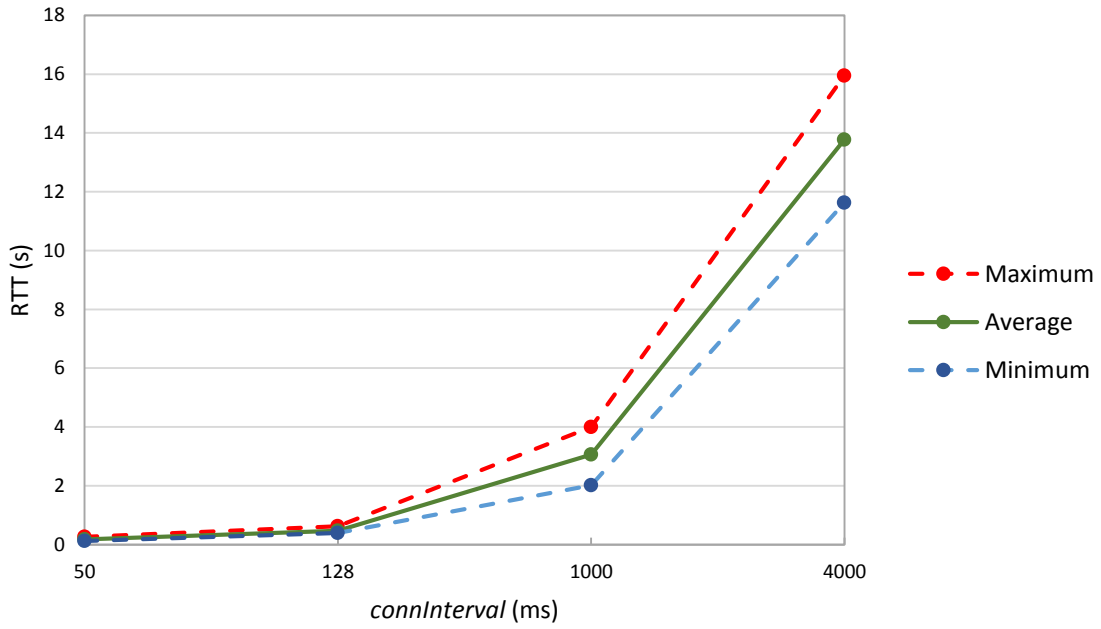


Figure 36: Measured RTT results.

However, for *connInterval* of 4 s, the packet sending interval (of 1 s) is lower than *connInterval*. For this setting, the minimum and the average RTT latency increase, which we attribute to node queuing delay. An additional comment is that we observed

packet drops, in general, when the time between consecutive packets exceeded *connInterval*, due to memory overflow events. The average measured RTT, when *connInterval* is set to 1 s and 4 s, is 3.0 s and 13.8 s, respectively.

4.5.3. Current consumption characterization of 6LN, 6LR and 6LBR

In this section, we characterize the current consumption of the 6BLEMesh node roles shown in Figure 33 (i.e. 6LN, 6LR and 6LBR), in different phases of their lifecycle. Such phases are: i) initialization, ii) node discovery, and iii) established Link Layer connection.

Figure 37 illustrates the current consumption of a 6LN when it is turned on. In the figure, marker 1 indicates the instant at which the node is powered, whereas marker 2 indicates the time when the 6LN starts node discovery. This process takes 573.2 ms and consumes 2.5 mA in average.

Figure 37: Current consumption of a 6LN during initialization

Figure 38 shows the current consumption of the 6LN, once a Link Layer connection has been established, during a data interval of 1 s, for a *connInterval* of 125 ms. There are 8 visible current consumption peaks between markers 1 and 2 that correspond to active parts of 8 consecutive connection intervals. At the end of this subsection, we discuss and compare the current consumption of the considered 6LN, 6LR and 6LBR during a data interval of 1 s.

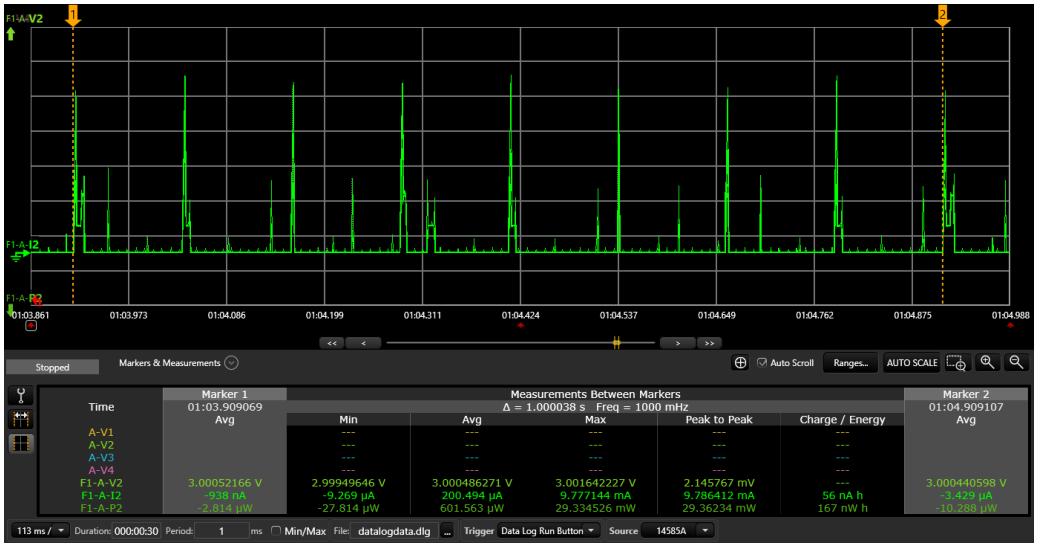


Figure 38: Current consumption of a 6LN during an established connection, for connInterval =125 ms and Data Interval = 1 s

Figure 39 depicts the initialization process of a 6LR. The process takes 528.8 ms (relatively similar to that of the 6LN), and it consumes slightly greater current, of 3.05 mA, in average.

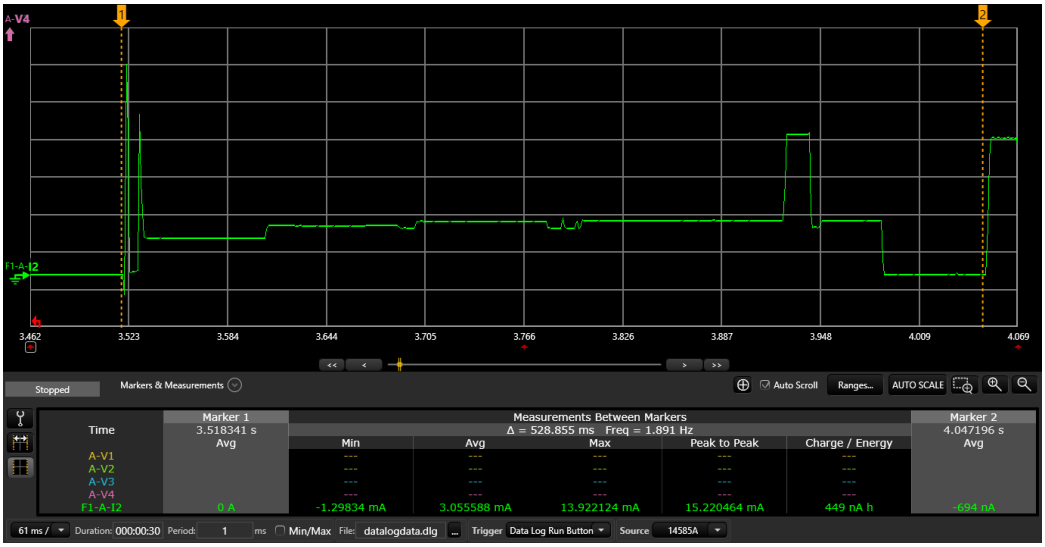


Figure 39: Current consumption pattern of a 6LR during initialization

The 6LR in our prototype testbed needs to establish a connection with the 6LN on one side, and with the 6LBR on the other side. Since the 6LR is assumed to be the master in both connections, it needs to keep scanning until both connections are established. Figure 40 shows the current consumption of the 6LR during node discovery. Given that the 6LR participates in two simultaneous connections, there are two main peak current consumption intervals per *connInterval*. For a given *connInterval*, the second current peak corresponds a connection which is already

established, while the first one corresponds to connection scanning activities intended to establish a new connection (in which it will be the master). Figure 41 shows the current consumption of the device once both connections are established.

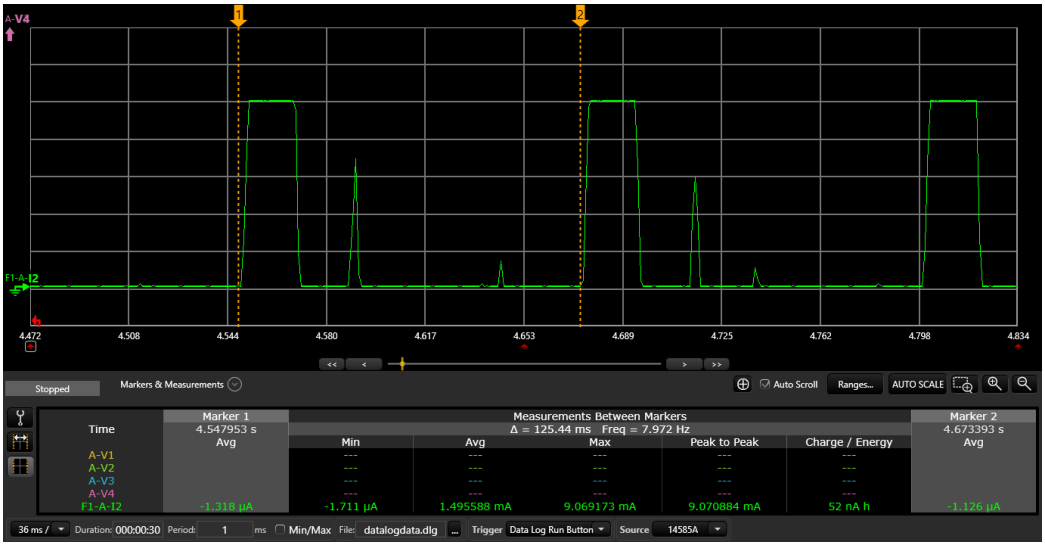


Figure 40: Current consumption pattern of the 6LR, once a connection has already been established, and during node discovery before establishing a second connection

Figure 41 depicts a 6LR connection interval (between markers 1 and 2) in which packets carrying no user data are sent. The average current consumption of the 6LR during one connection interval is 212.0 µA. In Figure 42, a data packet is transmitted via the first connection, while only empty packets are exchanged via the second connection. The average current consumption of the 6LR during one connection interval is 536.3 µA.

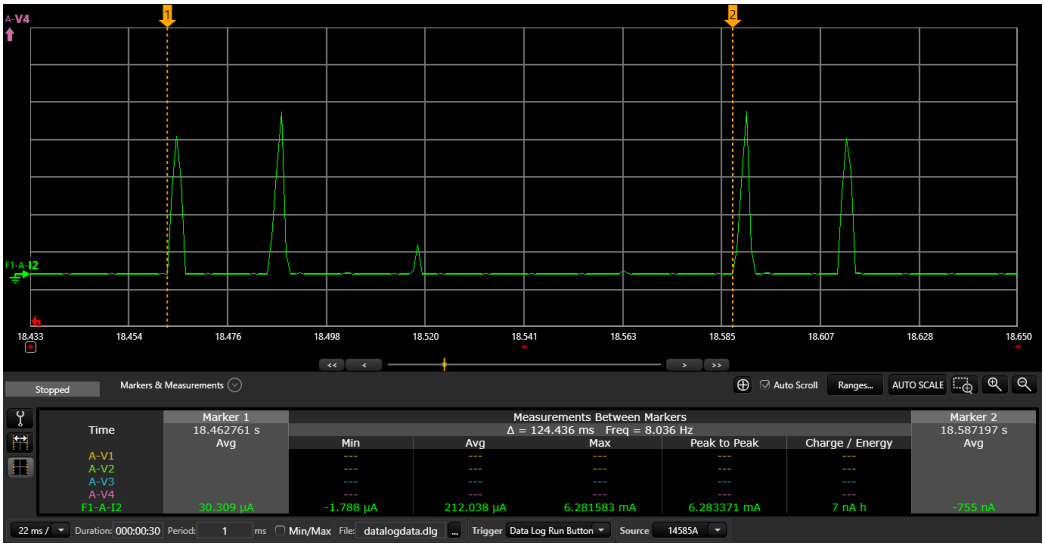


Figure 41: Current consumption pattern of the 6LR in a common connection interval without user data transmission

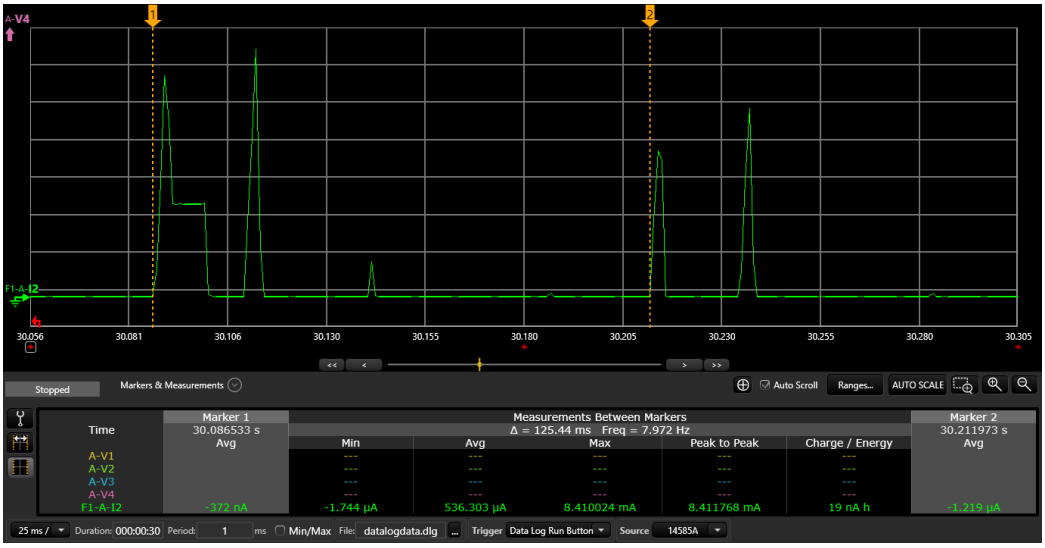


Figure 42: Current consumption pattern of the 6LR in a connection interval with one packet transmission carrying user data only in the first connection

Figure 43 depicts the 6LR current consumption during one data interval of 1 s. During the data interval, there are 8 connection intervals, each one showing two current consumption peaks (one peak for each established connection). The current consumption measured during one second is 323.7 µA .

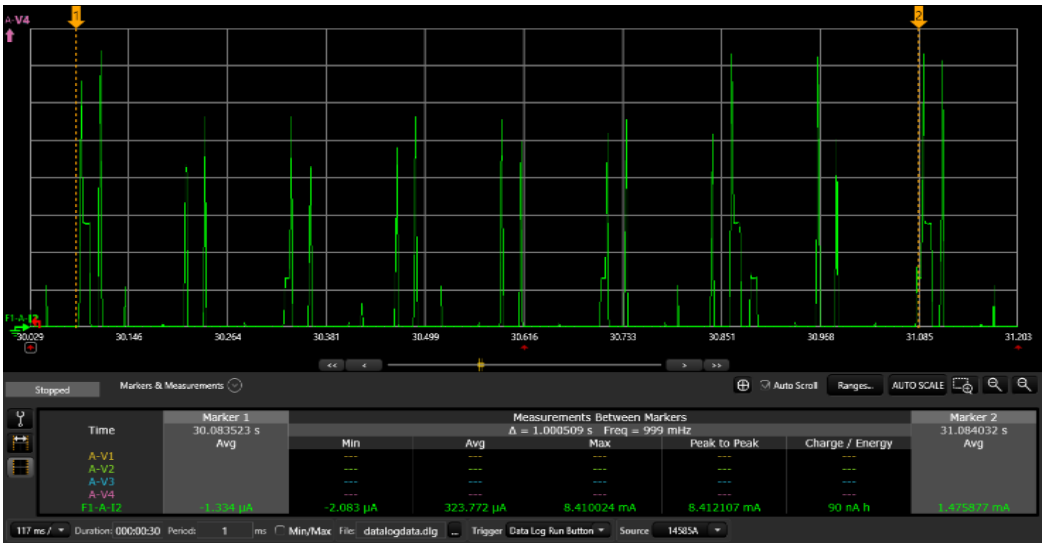


Figure 43: Current consumption pattern of a 6LR (for connInterval=125 ms and Data Interval = 1 s)

Figure 44 depicts the current consumption of the 6LBR during initialization. This process takes 476.6 ms, i.e. less than that of the 6LN and the 6LR, but the average current consumption is 4.1 mA, which is higher than that of the 6LN and the 6LR. These differences are due to the implementation differences between the LaunchPad and SensorTag platforms.

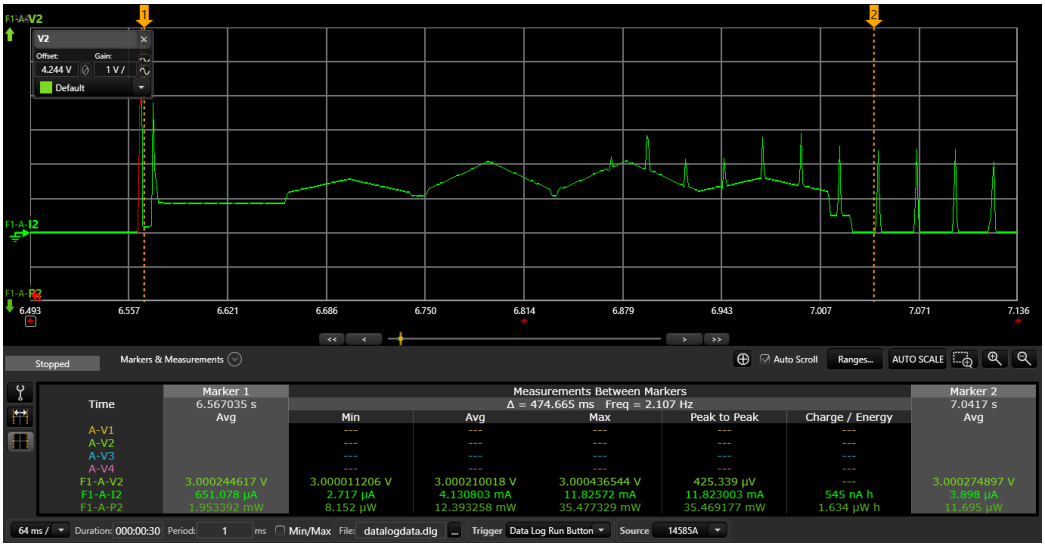


Figure 44: Current consumption pattern of the 6LBR during initialization

After initialization, like the other nodes, the 6LBR starts node discovery until it finds a neighbor (the 6LR) which establishes a connection with it. Figure 45 illustrates the time interval where the 6LBR transitions from node discovery (in which it performs scanning activities) until it runs the first connection event.

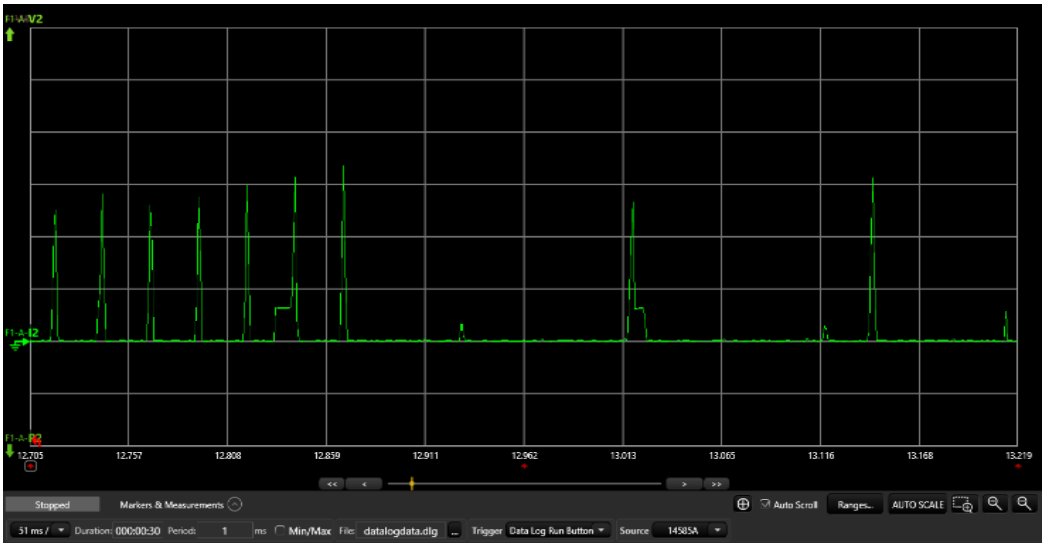


Figure 45: Current consumption model in node discovery and connection establishment in 6LBR

The 6LBR is typically a relatively powerful device (e.g. a cell phone), therefore its energy consumption is not typically critical. However, for the sake of completeness, Figure 46 illustrates the current consumption of the 6LBR considered in our experiments, where the node just receives a packet from the 6LR and returns it back as soon as possible. The average current consumed by the 6LBR during one data interval during the described operation is 186.6 µA.

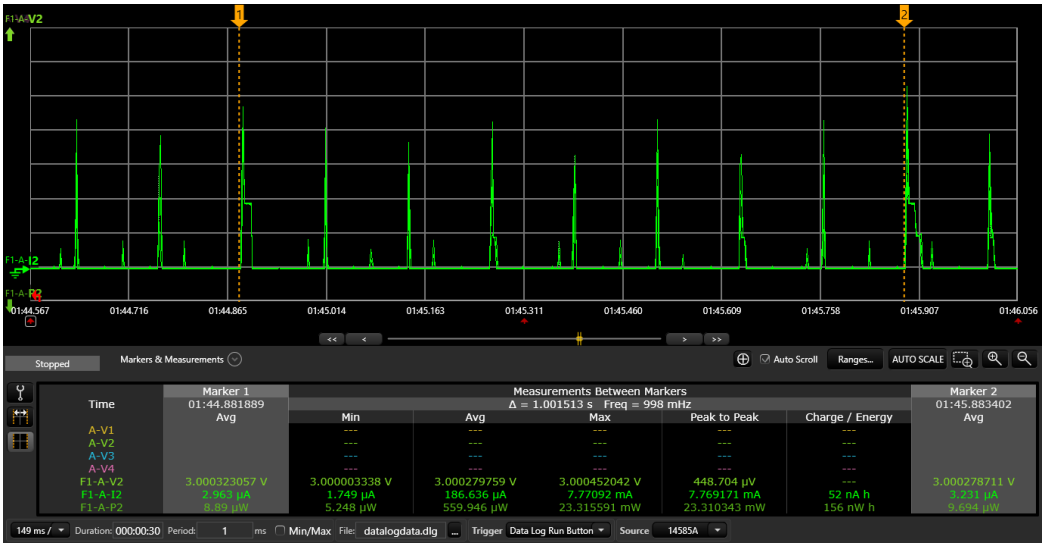


Figure 46: Current consumption of a 6LBR, for connInterval =125 ms and Data Interval = 1s

As shown in Figure 45, the current consumption pattern of the 6LBR is relatively similar to the 6LN one. In contrast, the current consumption pattern of the 6LR is fundamentally different, since it handles two connections simultaneously.

As a conclusion from this subsection, the average current consumption of 6LN, 6LR and 6LBR during one data interval (i.e. 1 s in this study) is 200.9 µA, 323.8 µA and 186.6 µA respectively.

4.5.4. Current consumption of a 6LN

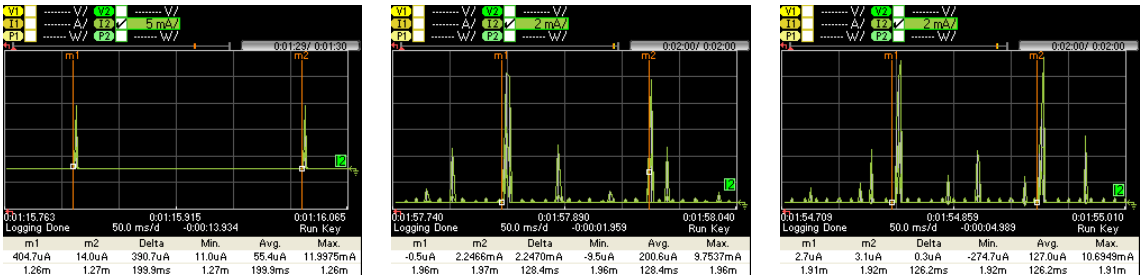
Since the 6LN is expected to be an energy-constrained device, we next focus on it, and evaluate its power consumption with greater detail, for three popular commercial BLE device platforms: i) PCA10028, ii) LaunchPad, and iii) SensorTag. We measure the current consumption of the aforementioned devices by using the Agilent N6705A Power Analyzer. The transmit power was set to 0 dBm for all devices.

First, we study the current consumption of these devices in two situations: i) in a connection interval without actual data transmission (i.e. with an empty packet exchange intended to keep the BLE connection alive), and ii) in a connection interval wherein the device transmits one data packet every connection interval. These measurements allow to derive a power consumption model. Then, we study the power consumption efficiency that corresponds to different data packet transmission intervals. Finally, we study the device lifetime of the considered device types for different data transmission intervals.

4.5.4.1. Current consumption during a connection interval without data transmission

The 6LN will often participate in connections where not all connection intervals include the exchange of BLE data units carrying IPv6 packets. Note that, to maintain link-level connectivity and synchronization, both endpoints of a BLE connection will always exchange BLE data units every *connInterval*. Accordingly, we have measured the power consumption of connection intervals without actual data transmission.

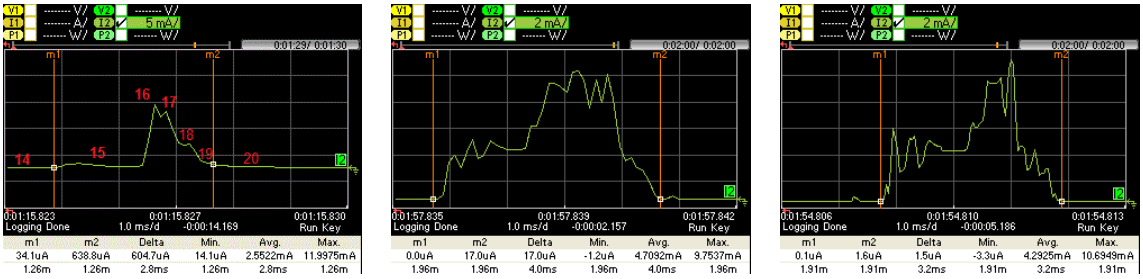
Figure 47 shows the current consumption of a connection event with no data packet transmission for a) PCA10028, b) LaunchPad, and c) SensorTag. We perform current consumption measurements for the device in the slave role. The slave is typically in sleep mode. When a new connection event starts, the slave wakes up, it receives one packet from the master, it replies by sending another packet to the master, and it returns to sleep mode.



a) PCA 10028 *b) CC2650 LaunchPad* *c) CC2650 SensorTag*

Figure 47: Current consumption of one connection interval without data transmission.

Figure 48 illustrates the active part of the connection interval shown in Figure 47. We identified 7 current consumption states, namely: sleep mode, wake up, packet reception from the master, packet response to the master, post-processing, radio off and cool down.



a) PCA 10028 *b) CC2650 LaunchPad* *c) CC2650 SensorTag*

Figure 48: Current consumption of the active part of a connection interval without data transmission.

Figure 48 provides the characteristics of each state. Screenshots from the power analyzer showing the details of each corresponding state are presented in Appendix B. Figure 48 shows how the hardware platforms considered follow different approaches that exhibit different current consumption patterns, due to BLE implementation differences.

Table 6: State characterization of a connection interval without data transmission.

State #	Description		Duration Value (ms)				Current Consumption Value (mA)		
		Variable	PCA10028	LaunchPad	SensorTag	**Variable**	PCA10028	LaunchPad	SensorTag
14	Sleep mode	$T_{Sleep\text{-}ND}$	-	-	-	$I_{Sleep\text{-}ND}$	0.015	0.014	0.019
15	Wake up	$T_{Wake\text{-}up\_ND}$	1.7	1.6	1.5	$I_{Wake\text{-}up\_ND}$	0.689	2.769	3.391
16	Packet reception from the master	$T_{Receive\text{-}ND}$	0.206	1.2	0.702	$I_{Receive\text{-}ND}$	5.908	7.321	6.904
17	Packet response to the master	$T_{Response\text{-}ND}$	0.290	0.610	0.198	$I_{Response\text{-}ND}$	10.032	7.376	8.829
18	Post Processing	$T_{Post\_Proc\text{-}ND}$	0.183	0.290	0.504	$I_{Post\_Proc\text{-}ND}$	5.625	3.120	3.303
19	Radio off	$T_{Radio\_off\text{-}ND}$	0.397	0.305	0.267	$I_{Radio\_off\text{-}ND}$	2.835	1.583	1.033
20	Cool down	$T_{Cool\_Down\text{-}ND}$	28.5	0.710	0.183	$I_{Cool\_Down\text{-}ND}$	0.046	0.046	0.027

Based on the current consumption characterization shown in Table 6, it is possible to determine the average current consumption of a slave device in a BLE connection without actual data packet transmission (denoted $I_{NoData\text{-}CI}$). This variable can be obtained by using (4-14).

$$I_{NoData-CI} = \left(\sum_{i=14}^{20} T_i \cdot I_i \right) \tag{4-14}$$

All values of T_i and I_i represent the state durations and corresponding current consumptions shown in Table 6, except for the sleep duration (T_{14} which is denoted $T_{Sleep\text{-}ND}$). The latter can be computed by using (4-15) and (4-16), where T_{CI} refers to *connInterval* and $T_{ND\text{-}active}$ denotes the sum of the active time parts within *connInterval*.

$$T_{Sleep-ND} = T_{CI} - T_{ND-active} \tag{4-15}$$

$$T_{ND-active} = \left(\sum_{i=15}^{20} T_i \right) \tag{4-16}$$

The average current consumption of a slave, without data transmission, for the different devices considered, and for different *connInterval* settings, is depicted in Figure 49. As expected, the average current consumption decreases with *connInterval*, since the influence of sleep intervals increases. For example, when *connInterval* is set to 50 ms and 4 s, there are 20 and 0.25 complete connection intervals during 1 second, respectively. That is why the current consumption when *connInterval* is set to 50 ms is dramatically higher than when it is set to 4 s. Note that, in state 15 (see Table 6), during its wake up process, PCA10028 consumes significantly lower current than

LaunchPad and SensorTag, while state 16 is shorter for PCA10028. In consequence, the average current consumption by PCA10028 is lower than that of the other considered devices.

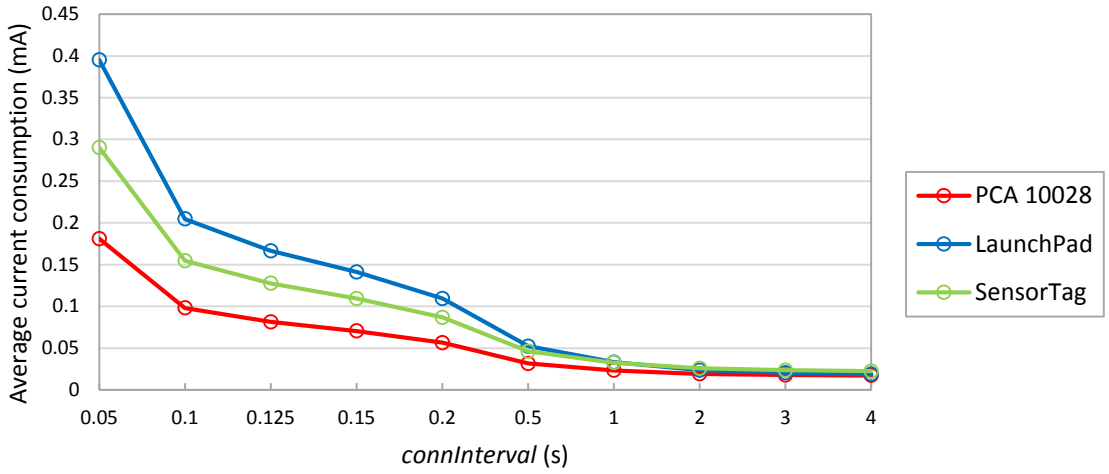


Figure 49: Average current consumption for different connInterval settings, without data transmission.

Figure 50 depicts the energy consumption of each device type in the slave role, during one complete connection interval, without data transmission. For *connInterval* lower than 1 s, LaunchPad consumes more energy per *connInterval*, while otherwise SensorTag consumes a greater amount of energy per *connInterval*. This happens because the current consumption during sleep mode in SensorTag is higher than the LaunchPad one.

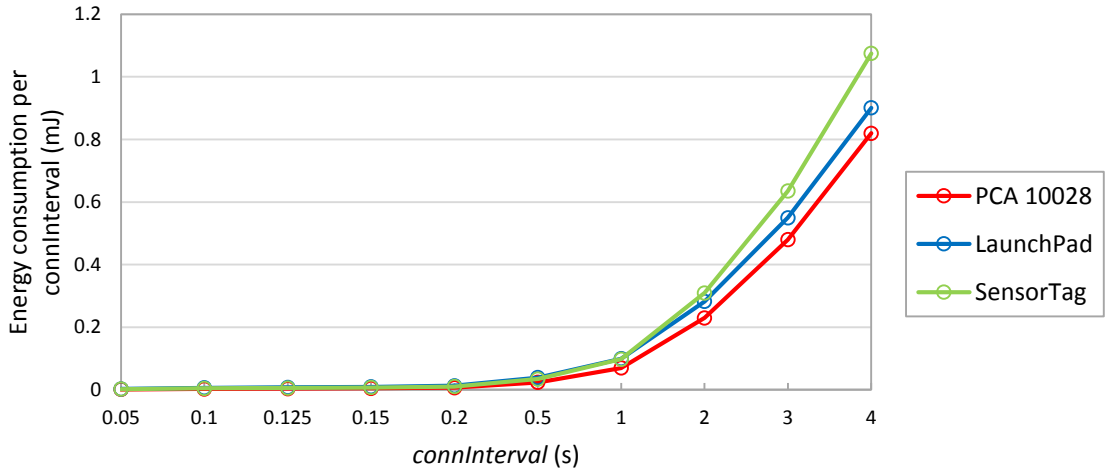
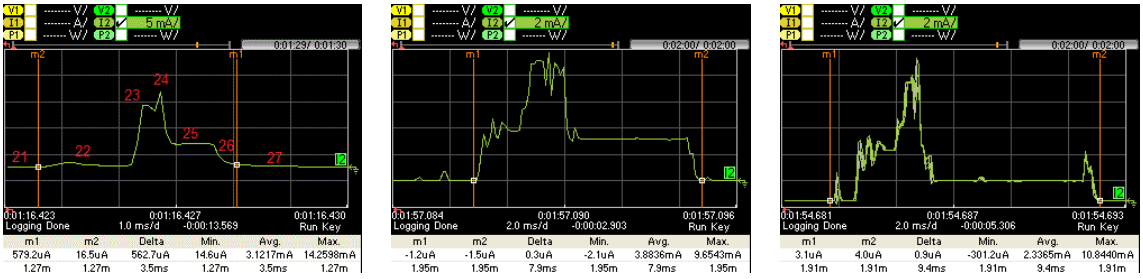


Figure 50: Average energy consumption during a connection interval, without data transmission.

4.5.4.2. Current consumption during a connection interval, with data transmission

When there is a data exchange between two connected devices in our experiment, the packets corresponding to states 16 and 17 in Table 6 carry data. Figure 51 depicts the current consumption pattern for such states, when data transmission takes place. The characteristics of such states (in terms of duration and current consumption) are further detailed in Table 7. The main differences between Table 6 and Table 7 are that devices in states 23 and 24 consume more energy, compared with states 16 and 17, and state 25 takes longer time than state 18 due to the processing of the received data. Screenshots from the power analyzer showing the details of each corresponding state are presented in Appendix C.



a) PCA 10028 *b) CC2650 LaunchPad* *c) CC2650 SensorTag*

Figure 51: Current consumption of the active part of a connection event with data transmission.

Table 7: State characterization of a connection interval, with data transmission.

State #	Description	Duration Value (ms)				Current Consumption Value (mA)			
		Variable	PCA10028	LaunchPad	SensorTag	**Variable**	PCA10028	LaunchPad	SensorTag
21	Sleep mode	$T_{Sleep\text{-}WD}$	-	-	-	$I_{Sleep\text{-}WD}$	0.015	0.014	0.019
22	Wake up	$T_{Wake\text{-}up\_WD}$	1.7	1.2	2.1	$I_{Wake\text{-}up\_WD}$	0.394	3.003	2.463
23	Packet reception from the master	$T_{Receive\text{-}WD}$	0.305	1.2	0.723	$I_{Receive\text{-}WD}$	7.099	6.494	7.043
24	Packet response to the master	$T_{Response\text{-}WD}$	0.305	0.923	0.671	$I_{Response\text{-}WD}$	10.764	6.747	4.913
25	Post Processing	$T_{Post\_Proc\text{-}WD}$	0.717	3.9	5.3	$I_{Post\_Proc\text{-}WD}$	4.699	3.127	1.503
26	Radio off	$T_{Radio\_off\text{-}WD}$	0.414	0.679	0.572	$I_{Radio\_off\text{-}WD}$	2.595	1.557	1.599
27	Cool down	$T_{Cool\_Down\text{-}WD}$	23.6	0.610	0.259	$I_{Cool\_Down\text{-}WD}$	0.043	0.048	0.010

We next calculate the average current consumption of a slave device in a BLE connection with actual data packet transmission (denoted $I_{Data\text{-}CI}$). This variable can be obtained by using (4-17).

$$I_{Data-CI} = \left(\sum_{i=21}^{27} T_i \cdot I_i \right) \quad (4\text{-}17)$$

The values of T_i and I_i in (4-17) represent the state durations and corresponding current consumptions shown in Table 7, except for the sleep time duration (T_{21}, which is denoted $T_{Sleep\text{-}WD}$). T_{21} can be determined based on (4-18) and (4-19), where $T_{WD\text{-}active}$ refers to the sum of active time parts within *connInterval* when an IPv6 packet is sent.

$$T_{Sleep-WD} = T_{CI} - T_{WD-active} \quad (4\text{-}18)$$

$$T_{WD-active} = \left(\sum_{i=22}^{27} T_i \right) \quad (4\text{-}19)$$

The average current consumption of a slave, when a data packet is actually transmitted, for the considered devices, and for different *connInterval* settings, is depicted in Figure 52.

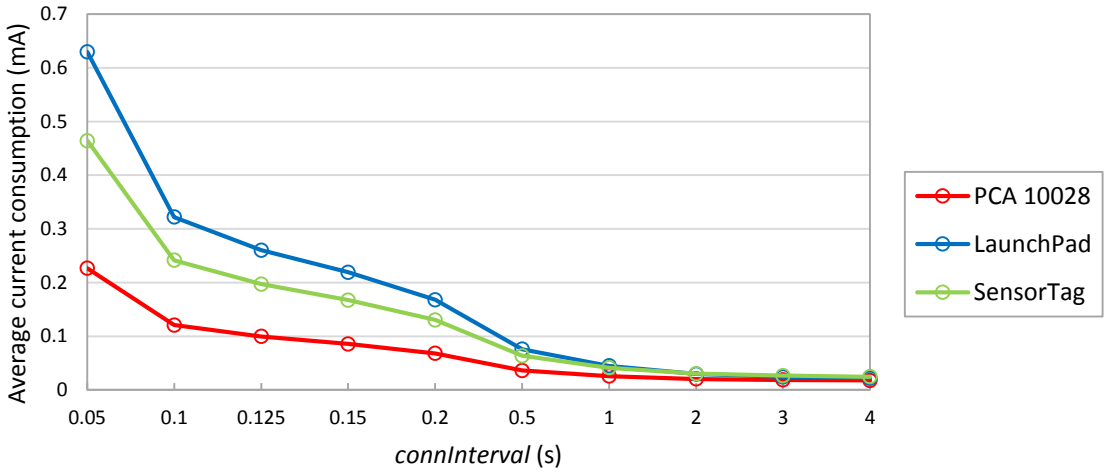


Figure 52: Average current consumption as a function of connInterval (with data transmission).

Figure 53 depicts the energy consumption per connection interval. Similarly to Figure 50, for *connInterval* lower than 1 s, LaunchPad consumes more current, while otherwise SensorTag shows a greater current consumption. Curves in Figure 53 are similar to those in Figure 50, although with higher values. The highest energy consumptions per connection interval shown in Figure 53 are 1.17 mJ, 1.04 mJ and 0.84 mJ, for SensorTag, LaunchPad and PCA10028, respectively. However, in Figure 49, these values are 1.07 mJ, 0.90 mJ and 0.81 mJ, respectively.

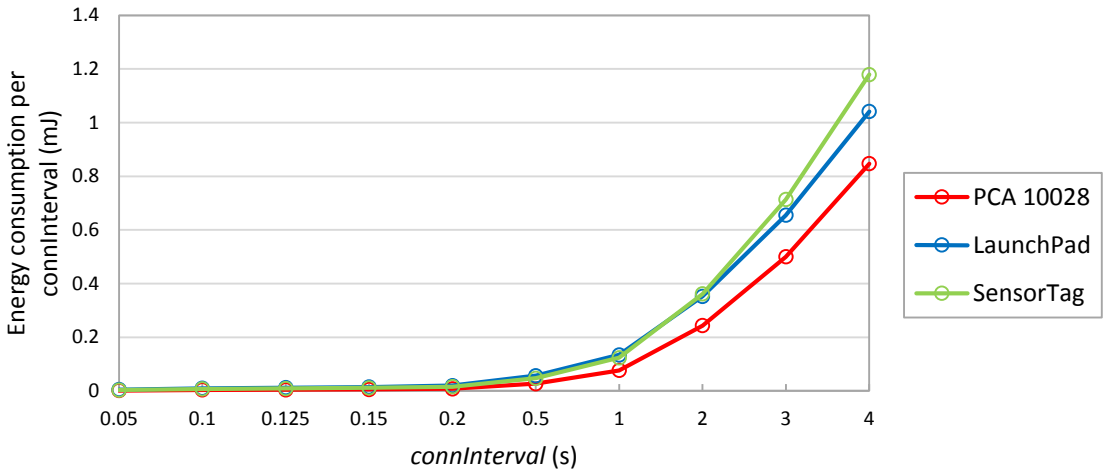


Figure 53: Energy consumption per connection interval (with data transmission).

4.5.4.3. Current consumption of the different hardware platforms: a comparison

Device current consumption during connection intervals with and without data transmission were characterized in detail in 4.5.4.1 and 4.5.4.2, respectively. Figure 54 compares the average current consumption of the active parts of connection intervals with and without data. Note that, during the remaining time, the node is in sleep mode. (Thus, average current consumption during a complete connection interval will vary depending on *connInterval*.) As shown in Figure 54, current consumption of LaunchPad is greater than that of SensorTag and PCA10028. PCA10028 even consumes less current than LaunchPad and SensorTag in absence of data transmission. Besides, the current consumption to transmit an IPv6 data packet with PCA10028 is significantly lower than that of LaunchPad and SensorTag.

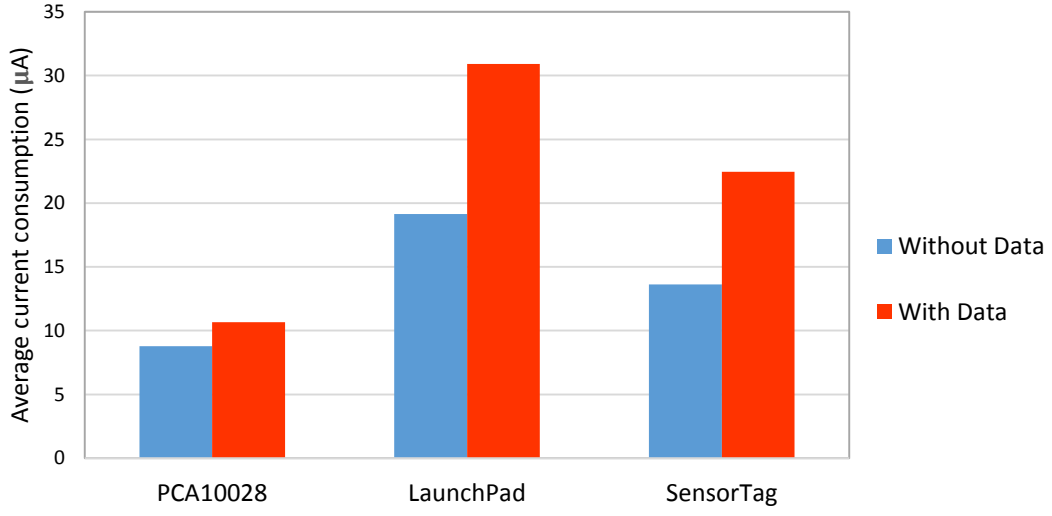


Figure 54: Average current consumption of active parts in connection events with and without data packets, for different device hardware platforms.

4.5.4.4. Energy efficiency

In this section, we study the energy efficiency of data delivery, based on the device states characterization shown in Table 6 and Table 7. The energy efficiency per conveyed byte of user data through a BLE connection is modeled by using (4-20).

$$\mathcal{E}_{Byte} = \frac{E_{DataPacket}}{L_{Payload}} \tag{4-20}$$

where $E_{DataPacket}$ is the energy that is consumed to deliver one data packet and $L_{Payload}$ is the user payload length of that data packet (in our prototype, 18 bytes). $E_{DataPacket}$ can be obtained by using (4-21).

$$E_{DataPacket} = I_{Avg} * V * T_{DI} \tag{4-21}$$

Within a data interval (T_{DI}), there may be connection intervals without actual data transmission, along with one data connection interval. We take this into consideration in order to compute the average current consumption, denoted I_{Avg}. The voltage assumed is 3 V, as required by the BLE devices used in our prototype implementation. I_{Avg} can be obtained as shown in (4-22).

$$I_{Avg} = [\left(1 - \frac{T_{CI}}{T_{DI}}\right) * I_{NoData-CI} + \left(\frac{T_{CI}}{T_{DI}}\right) * I_{Data-CI}] \tag{4-22}$$

$I_{NoData-CI}$ and $I_{Data-CI}$ can be obtained by using (4-14) and (4-17), respectively.

By using the previous equation, the energy that is consumed per delivered data byte, for different data intervals, and different *connInterval* settings, is depicted in Figure 55. The figure shows that the power consumed to deliver one byte increases with the data interval. That is because, with or without data to be sent, connections keep their periodic empty packet transmission, plus the energy consumption in sleep mode intervals, thus consume energy while data is not actually transmitted. The worst energy efficiency obtained is 6.59 mJ/B which is found for LaunchPad when the data interval is 100 s and *connInterval* is 50 ms. The best energy efficiency is 11 µJ/B, which is found for PCA10028, when the data interval is 4 s and *connInterval* is 4 s.

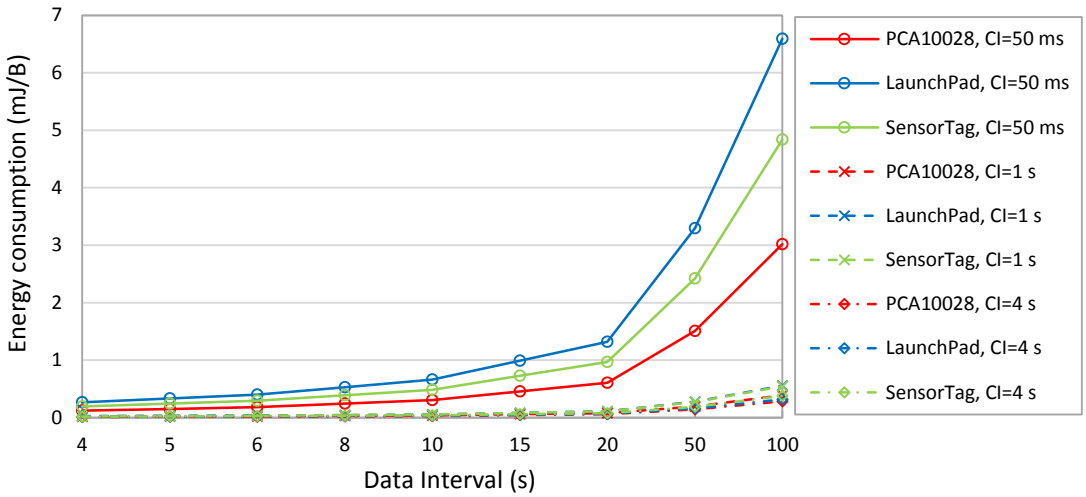


Figure 55: Energy consumption per delivered data byte, for different data intervals, connInterval (CI) settings, and different BLE device platforms.

4.5.4.5. Device lifetime

The device lifetime is an important performance parameter for a battery-operated device. In addition to the capacity of the energy source, device lifetime is affected by many BLE and application settings that determine the average current consumption, including *connInterval*, and data interval. In order to determine the device lifetime of our devices, we assume a simple 235 mAh battery that is supported by all three considered devices. Figure 56 shows the minimum and maximum device lifetime for different *connInterval* settings, and for different devices. In the figure, maximum lifetime curves correspond to the case where there is no data payload transmitted in any connection event. Minimum lifetime corresponds to $T_{CI}=T_{DI}$ in our study, where we assume that all connection intervals involve the exchange of one user data packet.

Figure 56 confirms that the lifetime of PCA10028 is greater than that of LaunchPad and SensorTag. Moreover, SensorTag lifetime is greater than LaunchPad one when *connInterval* is lower than 1 s, due to the lower sleep mode current consumption of SensorTag discussed earlier. The maximum lifetime is 420 days, 296 days and 300 days for PCA10028, LaunchPad and SensorTag, respectively, when *connInterval* is set to 1 s. The overall maximum lifetime is 573 days, which happens for PCA10028 when *connInterval* is set to 4 s, while the minimum lifetime is lower than 16 days, which happens for LaunchPad when *connInterval* is set to 50 ms.

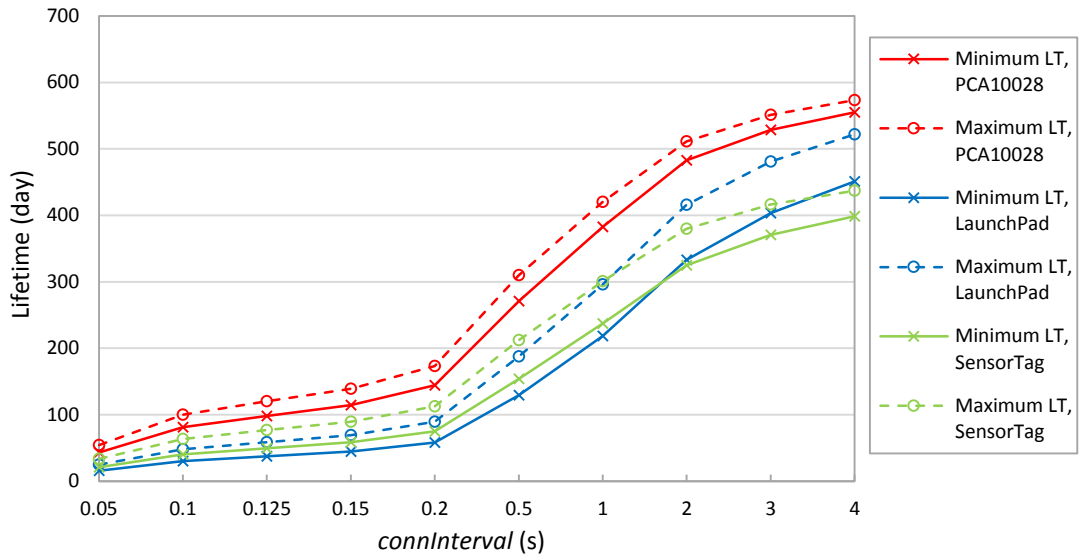


Figure 56: Device lifetime for the different devices considered. Minimum and maximum lifetimes correspond to data interval equal to connInterval, and data interval → ∞, respectively.

Figure 57 illustrates the device lifetime for different *connInterval* and data interval settings, and for the considered devices. Figure 57 shows that, while lifetime values tend to increase with data interval, the curves are almost horizontal for data intervals greater than *connInterval*. This confirms that the data interval setting has no significant impact on the device lifetime, in contrast with the *connInterval* setting, which is assumed to be smaller than or equal to the data interval in this study.

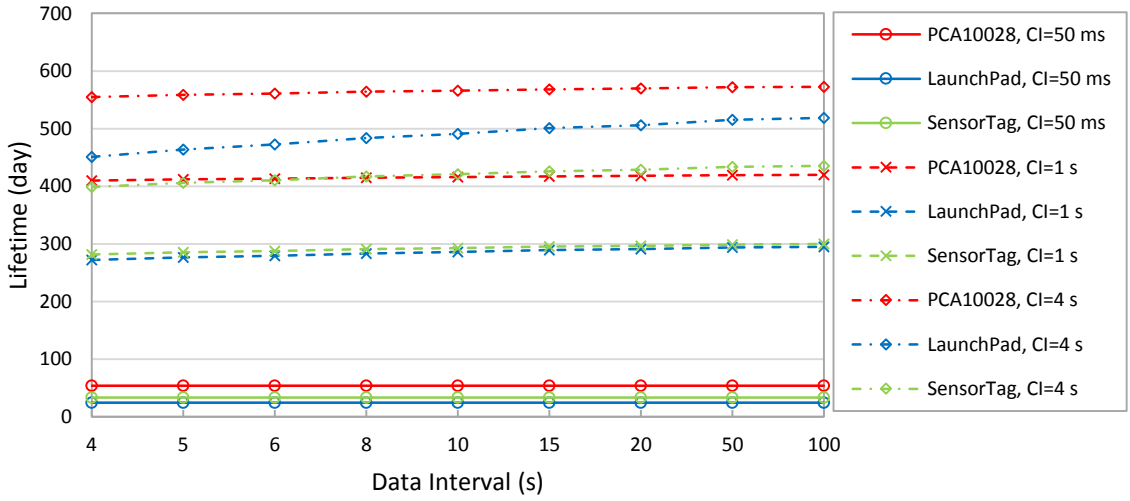


Figure 57: Device lifetime in terms of different connInterval and data intervals.

5. Comparison: Bluetooth Mesh Vs 6BLEMesh

As shown in the previous two chapters, Bluetooth Mesh and 6BLEMesh follow different approaches to enable BLE mesh networks. This chapter compares Bluetooth Mesh and 6BLEMesh, in terms of the following performance metrics and features: protocol stack (section 5.1), protocol encapsulation overhead (section 5.2), end-to-end latency (section 5.3), energy consumption (section 5.4), message transmission count (section 5.5), end-to-end reliability (section 5.6), variable topology robustness (section 5.7), and Internet connectivity (section 5.8). Finally, section 5.9 concludes the comparison in terms of design goals, performance, and main remarks.

5.1. Protocol stack

The protocol stacks of Bluetooth Mesh and 6BLEMesh have been introduced in sections 3.1 and 4.3.1, respectively. Figure 58 depicts a comparison of the BLE protocol stack, the Bluetooth Mesh protocol stack (for both advertising and GATT bearers), and the 6BLEMesh protocol stack.

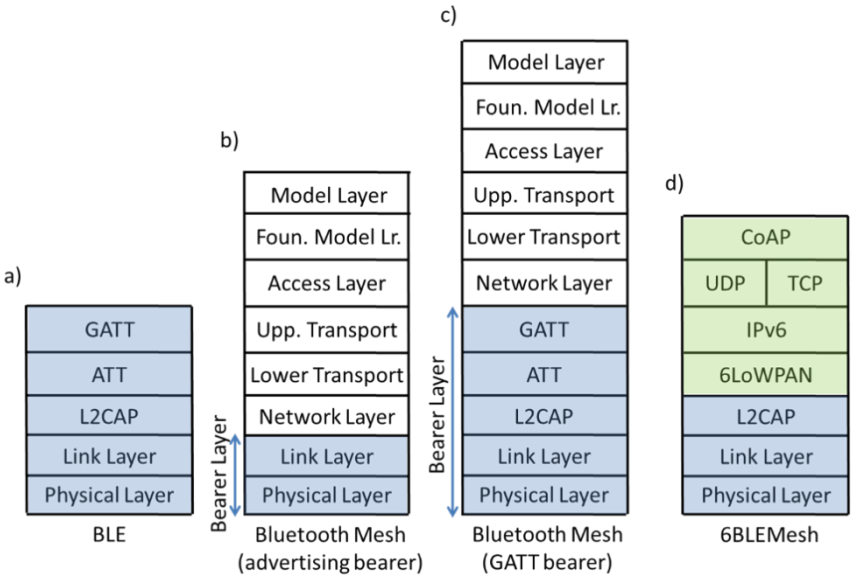


Figure 58: Comparing BLE, Bluetooth MESH and 6BLEMesh protocol stacks.

5.2. Protocol encapsulation overhead

We define the protocol encapsulation overhead of either Bluetooth Mesh or 6BLEMesh as the total header and footer overhead added by all protocol stack layers to a user data payload before transmission. We assume that the user data payload fits into a single Physical Layer data unit.

The minimum protocol encapsulation overhead of Bluetooth Mesh (29 bytes) is slightly greater than that of 6BLEMesh (25 bytes). Only the lowest protocol stack layer (which contributes 8 bytes to the protocol encapsulation overhead) is shared by both BLE mesh network approaches. In 6BLEMesh, header compression is crucial to produce a 7-byte compressed IPv6/UDP header (in contrast with a 48-byte uncompressed one). In Bluetooth Mesh, the 9-byte Network layer header is the greatest contributor to protocol encapsulation overhead. In addition, this header includes 4 bytes used for security purposes, such as identifying the keys used to protect a message, and protect against replay attacks.

5.3. End-to-end latency

We next study and discuss the end-to-end latency of packet transmission over a multihop path in Bluetooth Mesh and in 6BLEMesh. This performance parameter is particularly critical for applications where a human expects a quick reaction to an action (e.g. turning on a lightbulb after pressing a button on a remote control). Such applications are typical in smart home, a major target domain for BLE mesh networking. In these scenarios, interaction is often considered real-time when latency is below 500 ms [88] .

Figure 59 depicts the whole range of average per-hop latency values for Bluetooth Mesh and 6BLEMesh, assuming negligible processing time and ideal channel conditions. As it can be seen, both solutions offer flexibility for determining latency performance. The relevant involved parameters and mechanisms are described next.

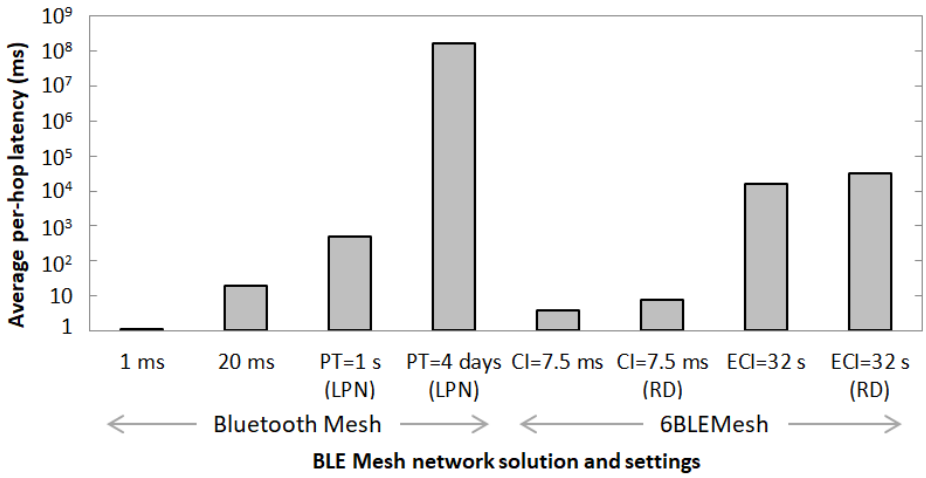


Figure 59: Theoretical average per-hop latency of Bluetooth Mesh and 6BLEMesh, for their whole ranges of main settings. PT, CI, ECI and RD stand for PollTimeout, connInterval, equivalent connInterval and Route Discovery, respectively.

In Bluetooth Mesh, each hop contributes at least the time required to transmit a packet via the advertising channels. This time depends on how BLE is implemented or configured, and may fall between 1 ms and 20 ms.

In Bluetooth Mesh, when the next hop is a LPN, the latter will only be able to receive data packets after polling its FN, which happens every *PollTimeout*. Therefore, in this case, the last hop will contribute a random, uniformly distributed, additional delay of up to *PollTimeout*. The minimum and maximum possible values for this parameter are 1 second and 4 days, respectively. Remarkably, the minimum *PollTimeout* value does not ensure real-time interaction when the destination node is a LPN.

In 6BLEMesh, the time required to deliver a packet from one node to its next hop (regardless of the role of the latter) is a uniformly distributed random variable up to connInterval. The minimum value allowed for this parameter is 7.5 ms, therefore real-time communication is possible in 6BLEMesh even when the destination node is a 6LN. The maximum connInterval setting is 4 s, although a BLE slave is allowed to skip a number of consecutive communication opportunities, leading to a maximum equivalent connInterval value of 32 s.

In 6BLEMesh, route discovery may be an additional contributor to end-to-end packet latency. If a reactive routing protocol is used, route discovery delay can be estimated as twice the one-way end-to-end delay. Proactive routing does not add a route discovery delay.

5.4. Energy consumption

We now study the energy consumption performance of Bluetooth Mesh and 6BLEMesh. We determine the theoretical lifetime of a battery-powered Bluetooth Mesh LPN and a 6BLEMesh 6LN, based on current consumption measurements carried out on PCA10028 platform, and a battery capacity of 235 mAh. We assume that the battery-powered device transmits a data message periodically. The data message payload size is 8 bytes (therefore, it fits into a single Physical Layer data unit) and the transmit power is 0 dBm.

Current consumption of a LPN in Bluetooth Mesh and a 6LN in 6BLEmesh are depicted in Figure 60 as a function of DI. Figure 60 highlights that current consumption strongly depends on network configuration parameters of Bluetooth Mesh (i.e. *PollTimeout* and *ReceiveWindow*) and 6BLEMesh (i.e. *connInterval*). 6BLEMesh, with an equivalent *connInterval* setting of 32 s exhibits the lowest current consumption, while the Bluetooth Mesh (for *PollTimeout* = 1 s and *ReceiveWindow* = 255 ms) has the highest current consumption among the considered options.

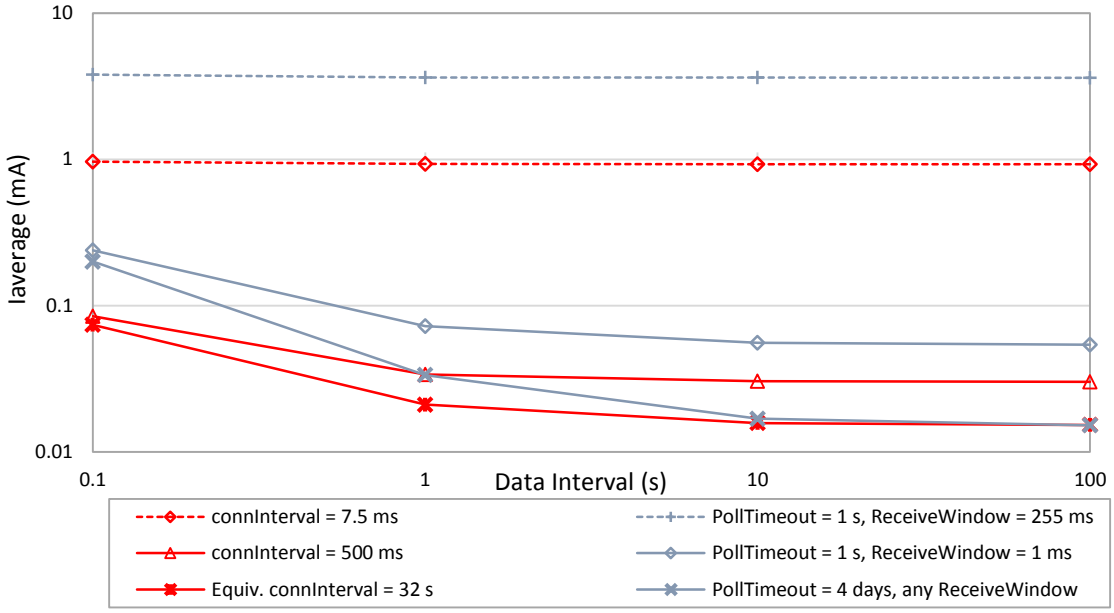


Figure 60: Energy consumption of a battery-operated device for Bluetooth Mesh and 6BLEMesh, as a function of Data Interval.

Figure 61 depicts the corresponding lifetime for each option shown in Figure 60. The DI only influences device lifetime for relatively infrequent link maintenance interactions. In such conditions, device lifetime increases asymptotically with DI. The maximum achievable device lifetime is 644 days.

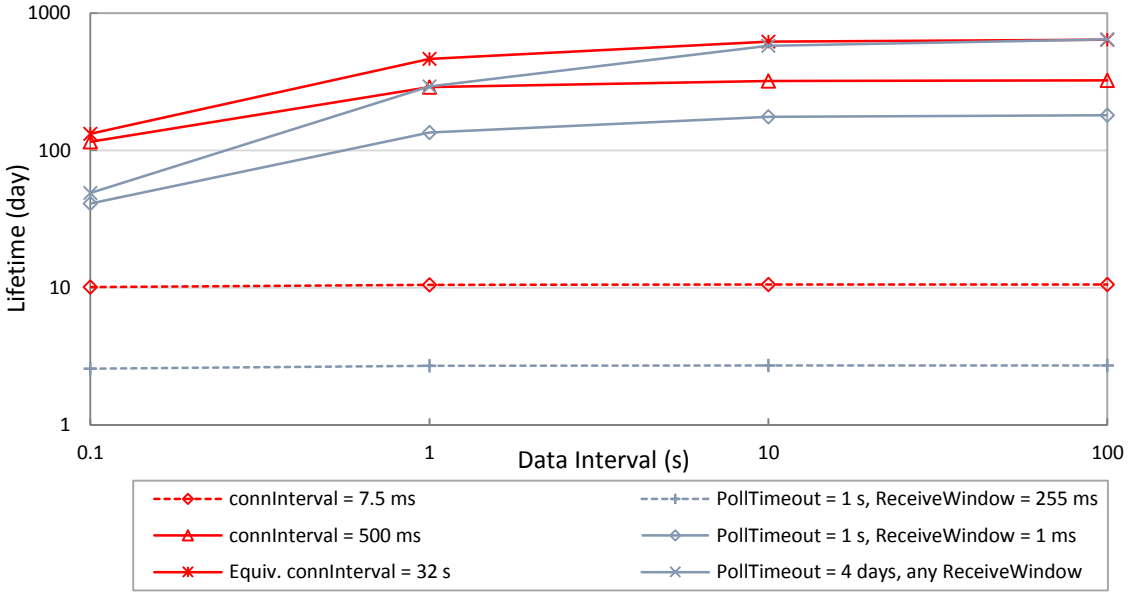


Figure 61: Lifetime of a battery-operated device for Bluetooth Mesh and 6BLEMesh, as a function of the Data Interval.

Reducing communication latency (by means of either decreasing *PollTimeout* or *connInterval*) decreases device lifetime. It is possible to achieve a 325-day device lifetime with 6BLEMesh, while keeping single-hop latency below the real-time threshold of 500 ms (i.e. for *connInterval* ≤ 500 ms). In contrast, in Bluetooth Mesh, the lowest assured latency is 1 s (for *PollTimeout* = 1 s), whereas the greatest device lifetime in this case is only 181 days. Remarkably, the operations carried out by the device in 6BLEMesh every *connInterval* (which include one receive and one transmit interval) consume less than 25% of the energy consumed in a poll action in Bluetooth Mesh (which includes three transmit intervals and one longer receive interval). Therefore, 6BLEMesh is able to provide low latency while offering energy-efficient operation at the same time.

5.5. Message transmission count

Equation (5-1) computes the message count in Bluetooth Mesh, denoted M_{Total_BM}. There are 3 main contributors to message count of different message types, which can be determined by using Equations (5-2) to (5-4). The factor of 3 that is present in (5-1) is due to the assumption that each advertising message is sent through the three BLE advertising channels. Let N indicate the total number of nodes in the network.

$$M_{Total\_BM} = 3 * \left(M_{Flooding} + M_{Polling} + M_{Heartbeat}\right) \quad (5\text{-}1)$$

$$M_{Flooding} = \left(\frac{1}{T_{Data}} * (N - l) * N_{in\ (TTL-1)\ radius}\right) \quad (5\text{-}2)$$

$$M_{Polling} = \left(l * \frac{1_{Poll\ message}}{T_{Poll}} \right) \quad (5\text{-}3)$$

$$M_{Heartbeat} = \left(N * (N - l) * \frac{1_{HeartBeat}}{T_{HeartBeat}} \right) \quad (5\text{-}4)$$

Equation (5-2) determines the number of data packets that are propagated throughout the network. We assume that all nodes send data packets periodically, every T_{Data}. $N_{in\ (TTL-1)\ radius}$ indicates the average number of nodes that forward data packets within a radius of *TTL-1* hops from the sender. We assume that there are l LPNs in the network. Note that these nodes will not perform forwarding tasks, while we assume that the rest of nodes will do so.

Equation (5-3) computes the number of polling messages generated by LPNs, as part of the friendship relationship with their FNs. We assume that all LPNs participate in friendship relationships. Each LPN sends one polling message every T_{poll}.

Equation (5-4) calculates the total number of Heartbeat messages that are sent, at every Heartbeat interval, $T_{HeartBeat}$. Heartbeat messages are relayed by all nodes except for leaf nodes.

Figure 62 depicts the contribution of each component of (5-1) in the total message count in Bluetooth Mesh, assuming the same value (of 1 s) for T_{Data}, T_{Poll} and $T_{Heartbeat}$. Note that we obtain $N_{in\ (TTL-1)\ radius}$ by simulation, and results shown correspond to the average from the network scenarios defined in Table 8.

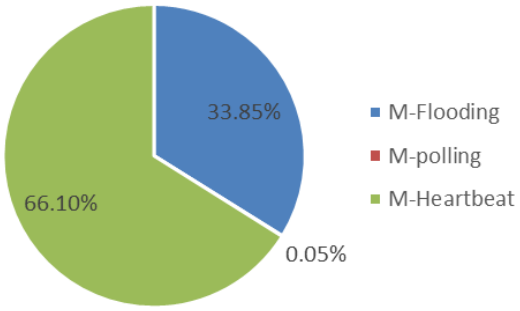


Figure 62: Contribution of each part of equation (5-1) to the total message count in Bluetooth Mesh, for an equal setting (of 1 s) for all interval parameters.

This study provides insights on the contribution of each component to the total number of messages, although in a real network, T_{Poll} and $T_{Heartbeat}$ are usually tens of times greater than the data interval. Figure 62 shows that 66.1% of the messages correspond to Heartbeat messages (as they are broadcast network-wide), while just 33.85% of messages are data messages (since they flooding is limited to the hopwise distance between source and destination). On the other hand, packets corresponding to the LPN polling operations is negligible.

Figure 63-64 show the message count as a function of T_{Data} and $T_{Heartbeat}$ parameters, for N equal to 69, 275 and 619 nodes, which are randomly distributed in 50*50 m^2, 100*100 m^2 and 150*150 m^2 areas, respectively. There are 7, 28 and 62 LPNs in Figure 63, Figure 64 and Figure 65, respectively. The same node density is kept in these different scenarios. *PollTimeout* is set to 30 s.

As shown in Figures 62-64, the total message count first tends to decrease quickly as the Heartbeat interval increases. Then, it becomes flat, especially in the cases with lower DI, because by increasing the Heartbeat interval, $M_{Flooding}$ becomes the main contributor to message transmission, instead of $M_{Heartbeat}$.

As expected, the curves in Figures 62-64 follow similar patterns, but with different values. For example, for a Heartbeat Interval of 400 s, the total message count is between 100 message/s and 10000 message/s in Figure 63, while it is between 1000 message/s and 100000 message/s in Figure 64. A similar trend can be observed when comparing Figure 64 and Figure 65. In fact, as previously mentioned, $M_{Flooding}$ becomes dominant for high Heartbeat interval values, thus the total message count increases by a factor in the order of N^2.

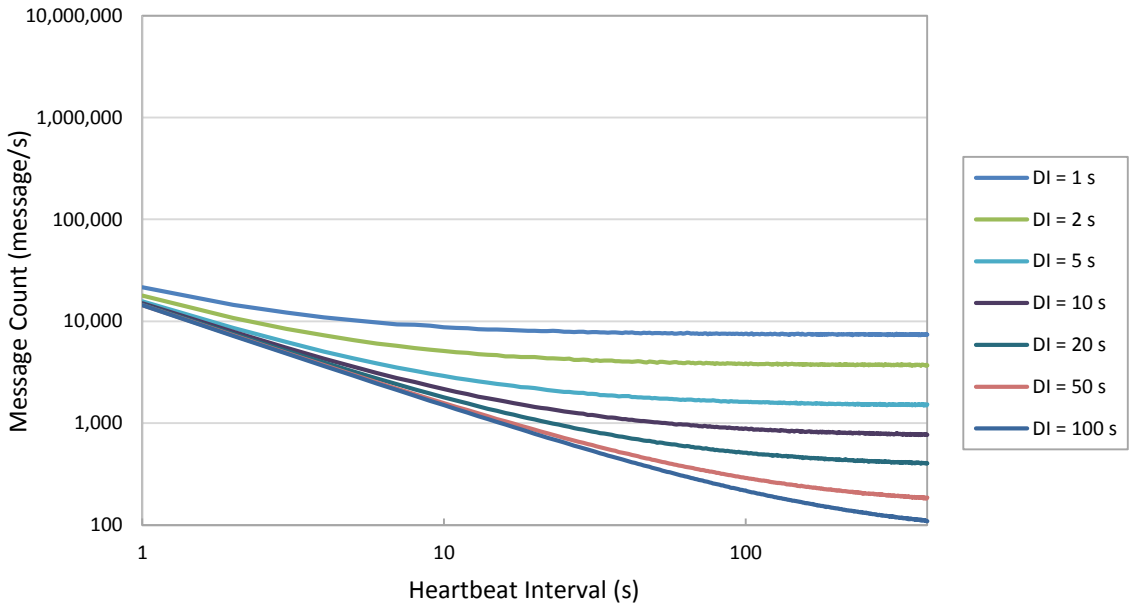


Figure 63: Total message count in Bluetooth Mesh as a function of Data interval and Heartbeat interval. Network scenario parameters: N=69, A=50·50 m^2 and l=7.

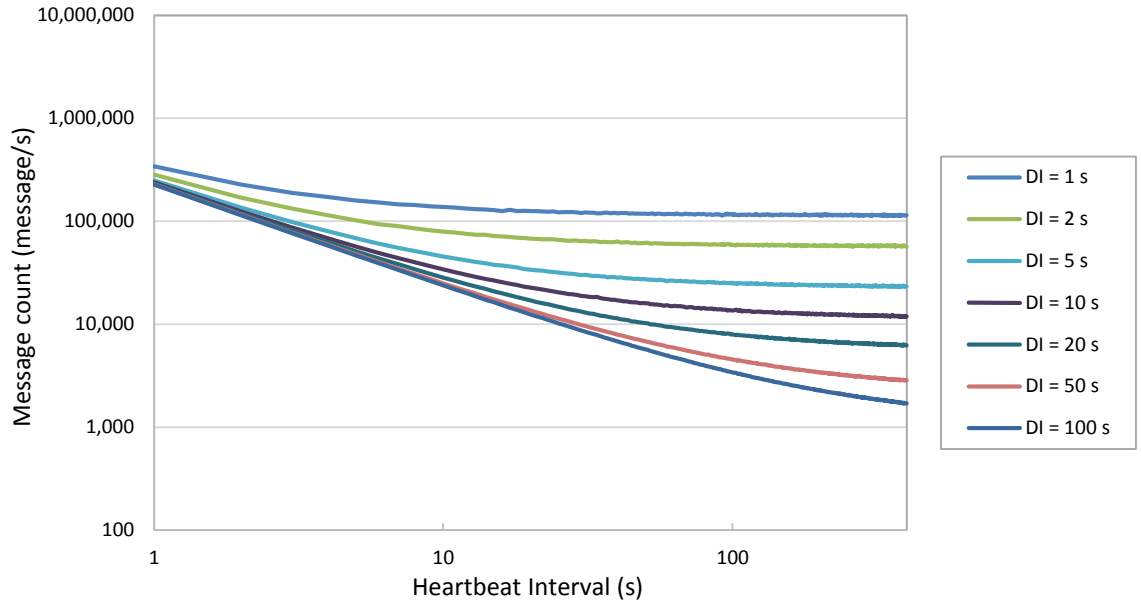


Figure 64: Total message count in Bluetooth Mesh as a function of Data interval and Heartbeat interval. Network scenario parameters: N=275, A=100·100 m² and l=28.

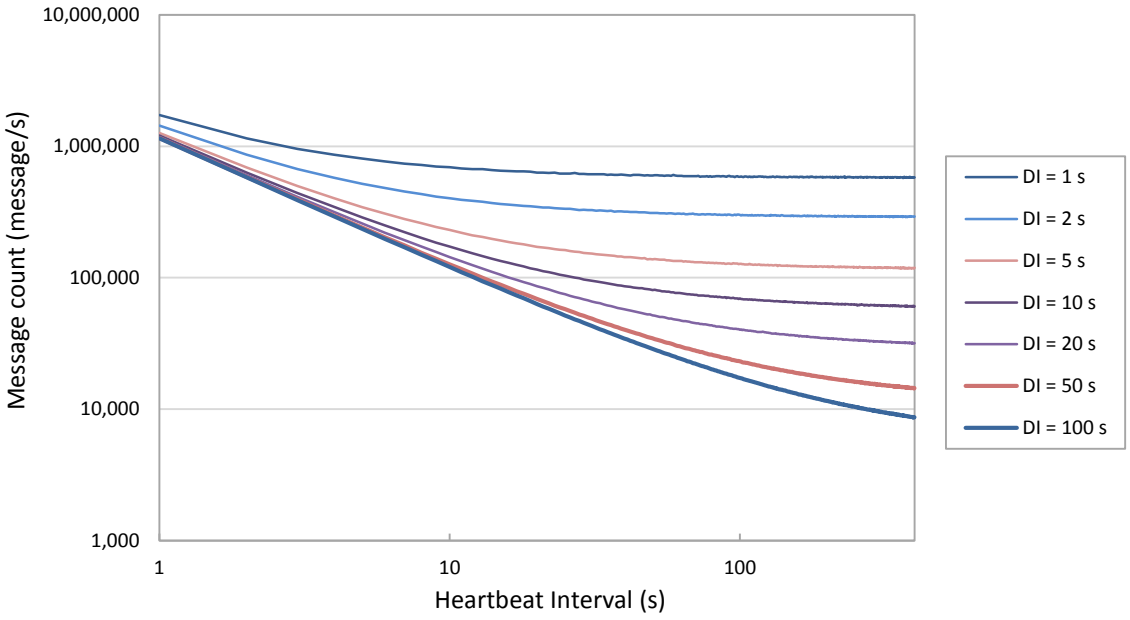


Figure 65: Total message count in Bluetooth Mesh as a function of Data interval and Heartbeat interval. Network scenario parameters: N=619, A=150·150 m² and l=62.

Equation (5-5) computes the message count in 6BLEMesh, which is denoted $M_{Total\_6BLEMesh}$. There are three contributors to the total message count in 6BLEMesh. They are described next, and can be computed by using (5-6) to (5-8).

$$M_{Total\_6BLEMesh} = \left(M_{MData} + M_{Link} + M_{Routing}\right) \quad (5\text{-}5)$$

$$M_{Data} = \left(\frac{1}{T_{Data}} * N * N_{hop}\right) \quad (5\text{-}6)$$

$$M_{Link} = \left(\frac{2}{T_{ConnInterval}} * N_{link}\right) \quad (5\text{-}7)$$

$$M_{Routing} = \left(\frac{DIO}{\mathrm{I}} * N\right) + \left(\frac{\mathrm{DAO}}{T_{DAO}} * (N-1)\right) \quad (5\text{-}8)$$

M_{Data} indicates message count to deliver one data packet through the shortest end-to-end route (provided by the routing protocol). N_{hop} denotes the number of hops of the end-to-end route.

M_{Link} indicates the messages periodically exchanged between a master and a slave in each established Link Layer connection in order to maintain the connection. N_{Link} denotes the total number of established connections in the network. Note that there are two message transmissions per link (one by the master, and one by the slave) every *connInterval*.

$M_{Routing}$ corresponds to routing protocol messages, assuming that RPL is used as the routing protocol. T_{DAO} denotes the time between two consecutive Destination Advertisement Object (DAO) messages and I is the time between two consecutive DODAG Information Object (DIO) messages. In RPL, DAOs allow their senders to receive data packets, whereas DIOs are the main control messages used to create and maintain the network topology.

Figure 66 shows the contribution of each total message count component in (5-5), with the same interval setting (of 1 s) for T_{Data}, $T_{connInterval}$, T_{DAO} and I (note that, in a real network setting, T_{DAO} usually is tens of times greater than $T_{connInterval}$). N_{Link} is obtained by simulation, where nodes are randomly deployed. Results correspond to the average from simulating the network scenarios defined in Table 8. Link connectivity maintenance has the main contribution to the total message count. With the considered settings, M_{Link} contributes the highest percentage of messages, as it depends on the number of links in the network.

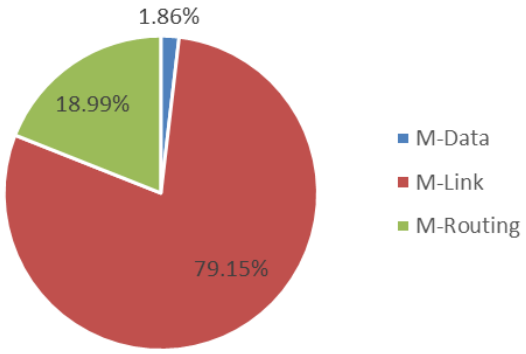


Figure 66: Contribution of each part of Equation (5-5) to the total message count in 6BLEMesh, for an equal setting (of 1 s) for all interval parameters.

Figure 67-68 show the message count in 6BLEMesh, as a function of T_{Data} and *connInterval*, for the same scenario settings assumed for the Bluetooth Mesh message count results shown in Figures 62 to 64. Figures 66-68 follow a decreasing pattern, as a function of *connInterval*. The maximum message count in Figure 67 is more than 60,000 message/s, which is found for *connInterval* = 10 ms, while for the same *connInterval* setting, in Figure 68 and Figure 69, the maximum message count is greater than 270,000 message/s and 630,000 message/s, respectively. In all three figures, the total message count dramatically decreases with *connInterval*. For *connInterval* greater than 100 ms, curves tend to be flat as *connInterval* increases. On the other hand, as expected, the relative impact of T_{Data} increases with *connInterval*.

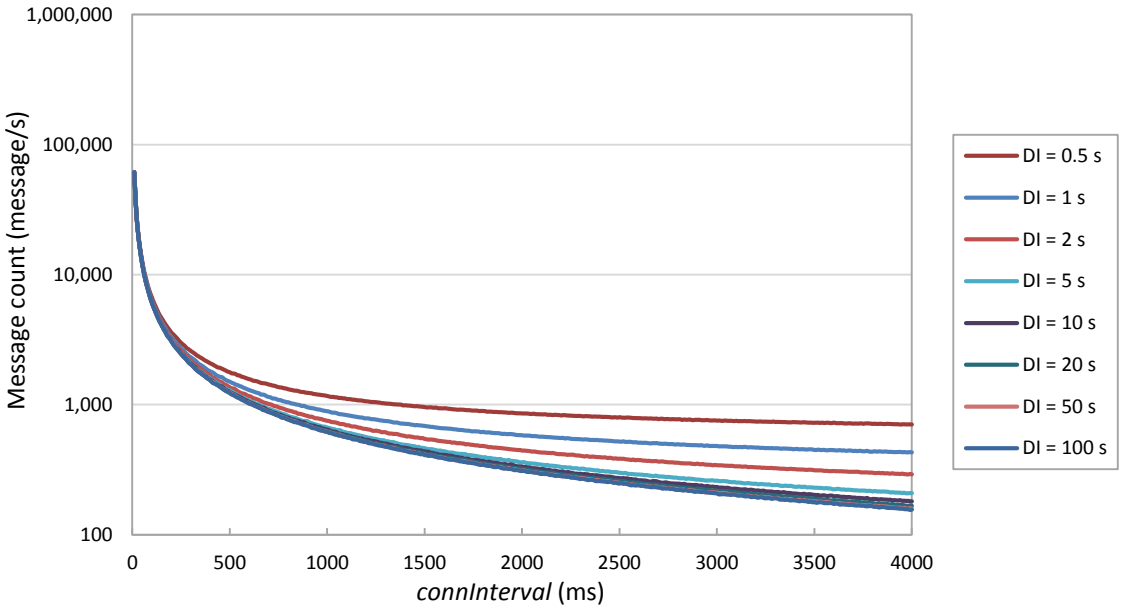


Figure 67: Total message count in 6BLEMesh, as a function of data interval and connInterval. Network scenario parameters: N=69 and A=50·50 m².

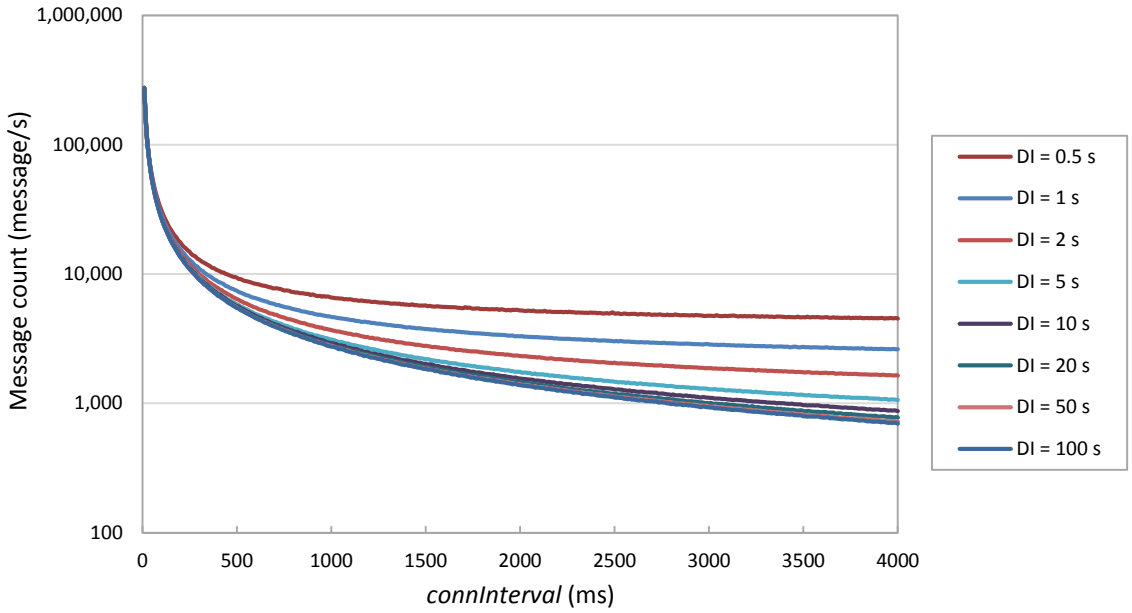


Figure 68: Total message count in 6BLEMesh approach as a function of data interval and connInterval. Network scenario parameters: N=275 and A=100·100 m².

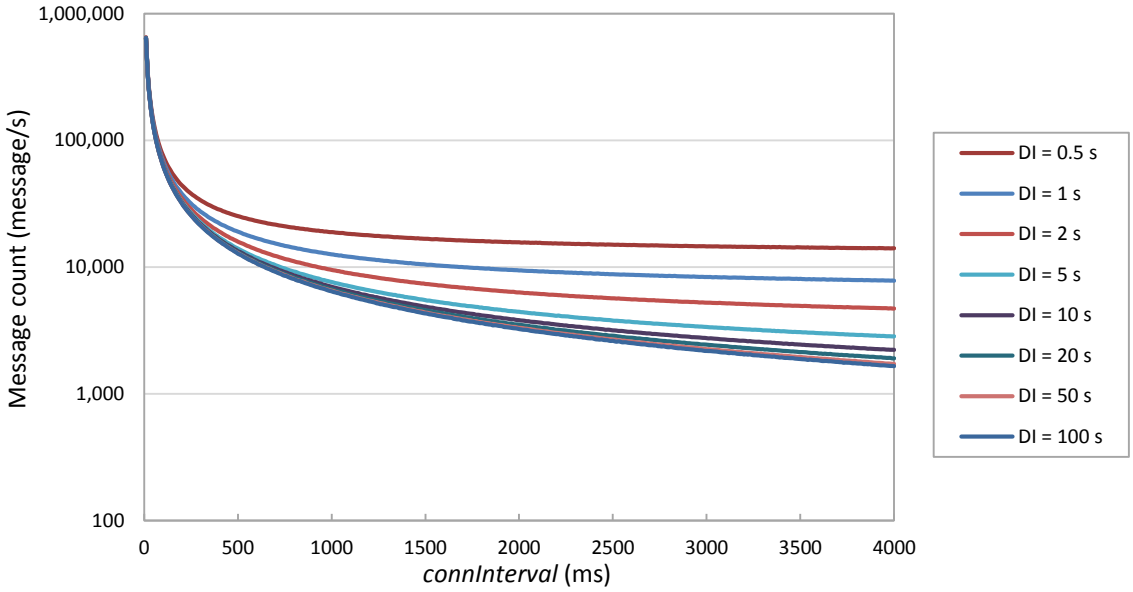


Figure 69: Total message count in 6BLEMesh approach as a function of data interval and connInterval. Network scenario parameters: N=619 and A=150·150 m².

We now evaluate by simulation, and compare, the message count (i.e. total number of message transmissions) of Bluetooth Mesh and 6BLEMesh, for a range of network sizes, node densities and protocol parameters. Table 8 classifies network size and node density into network scenarios where coverage area of a node is 400 m2. We assume connected, steady-state static networks without message losses, where each node sends a data message every data interval to a randomly chosen destination node. We also assume that,

in 6BLEMesh, shortest path routes are found at network initialization and maintained thereafter by the Point-to-Point extension of RPL (P2P-RPL) [89].

Table 8: A range of scenarios with different network size and densities.

Scenario Size	Area (m^2)	Node Degree	Scenario Name	Number of Nodes
Small	25·25	7	S 7	11
		11	S11	18
		15	S15	24
Medium	50·50	7	M 7	44
		11	M11	69
		15	M15	94
Large	100·100	7	L 7	175
		11	L11	275
		15	L15	375
Very large	150·150	7	VL 7	394
		11	VL11	619
		15	VL15	844

Figure 70 depicts samples of Small, Medium, Large, and Very large network sizes, where the average node degree is set to 11.

In order to characterize the different scenarios under consideration, Figure 71 depicts the average shortest hop distance between all nodes for each network scenario indicated in Table 8. This is an important parameter in order to understand the message count of Bluetooth Mesh networks and 6BLEMesh networks. Figure 71 shows that, for a given scenario area, as node degree decreases, the average hop distance increases since the number of available nodes decreases.

Figure 72 and Figure 73 show the total message count for the considered network scenarios, as a function of T_{Data}, for Bluetooth Mesh and 6BLEMesh, respectively. For Bluetooth Mesh, we have assumed $T_{Heartbeat}$=300 s and T_{Poll} = 30 s. For 6BLEMesh, we have assumed *connInterval* = 1 s.

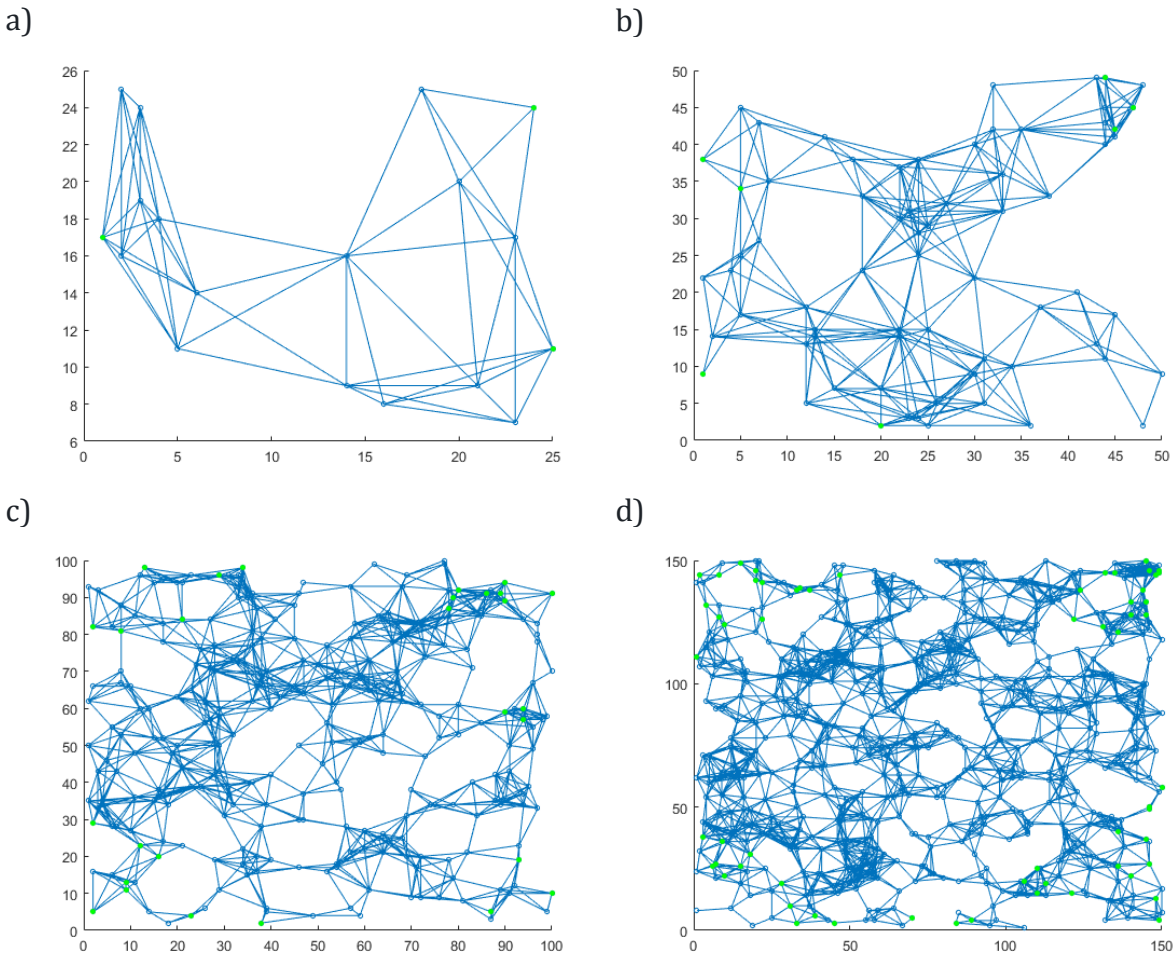


Figure 70: sample networks of a) Small, b) Medium, c) Large, and d) Very large networks, for a node degree of 11.

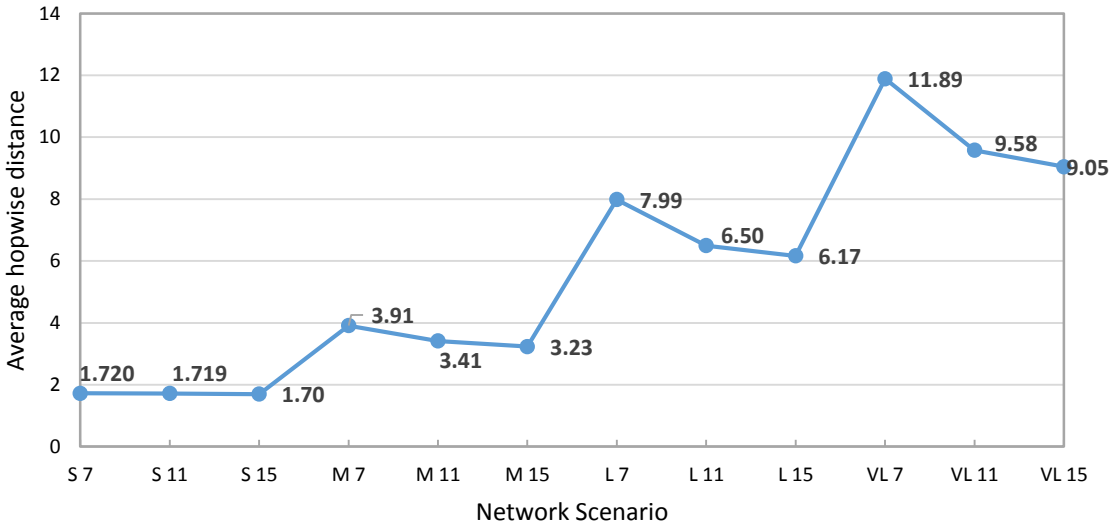


Figure 71: Average hopwise distance between all nodes in different network size.

Figure 72 shows that, in Bluetooth Mesh, the contribution of data message transmission, due to its controlled flooding approach, is dominant. Increasing T_{Data} by a factor of 100 (from T_{Data} = 1 s to T_{Data} = 100 s) decreases the total message count by a factor of 60.

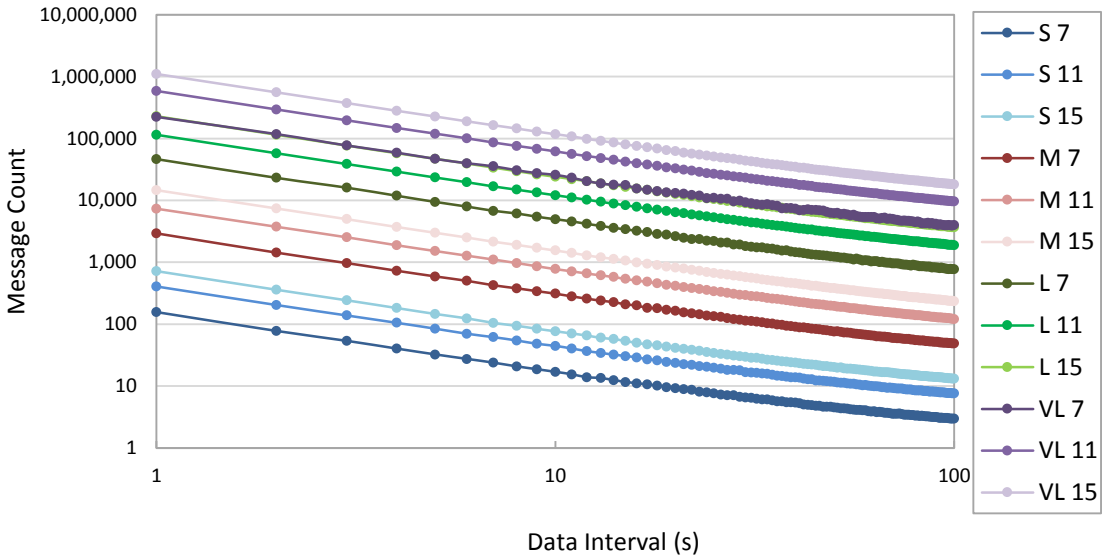


Figure 72: Message count in Bluetooth Mesh as function of Data Interval, for the scenarios shown in Table 8, and for $T_{Heartbeat}$ = 300 s and T_{Poll} = 30 s.

Figure 73 illustrates that T_{Data} does not significantly impact the total message count in 6BLEMesh networks. In fact, curves in Figure 73 are almost constant as a function of T_{Data}, except for the smallest T_{Data} values considered. This contribution will be even lower for lower *connInterval* settings.

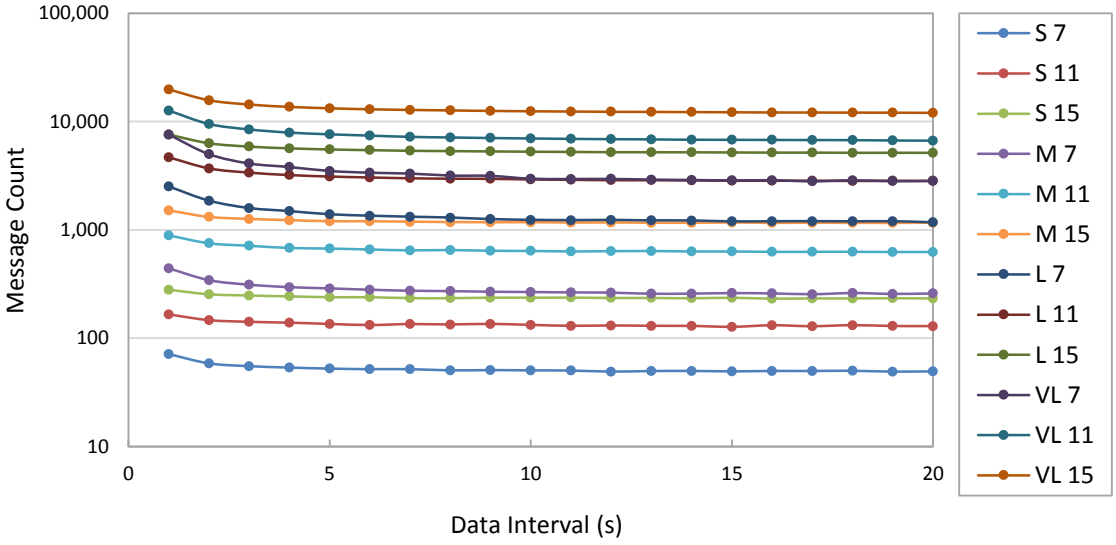


Figure 73: Message count of the 6BLEMesh network as function of Data Interval, for the scenarios shown in Table 8, and for connInterval = 1 s.

Next we compare the message count in Bluetooth Mesh and 6BLEMesh networks, for each network scenario defined in Table 8, for a range of values for data interval, Heartbeat interval, and *connInterval*. Note that, regarding the Heartbeat interval parameter, the greatest value that we consider is 1024 s, as it is a large value that is still reasonable to use in a practical setup. Regarding *connInterval*, we cover its whole range of possible values, including the equivalent *connInterval* setting considered in section 5.4.

Figure 74 illustrates the average value of the total message transmission count in the whole network per time unit. Each individual result has been obtained over 100 different topologies where nodes are randomly distributed over a square area.

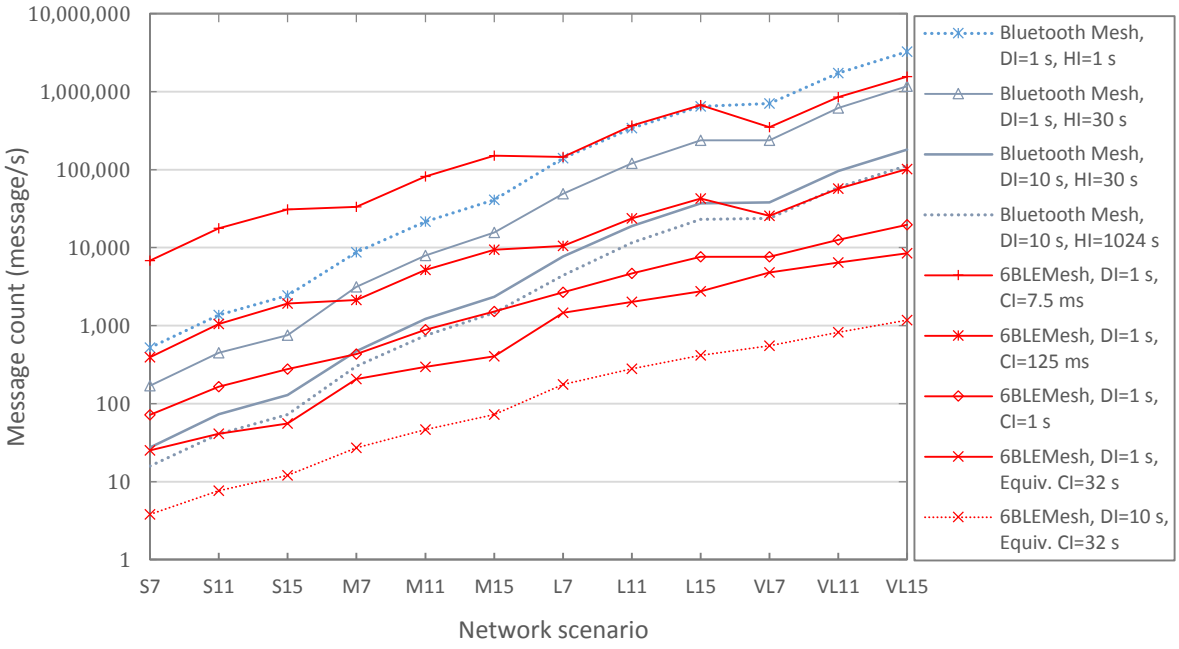


Figure 74: Message count of Bluetooth Mesh and 6BLEMesh, for the scenarios shown in Table 8, and for various parameter settings. CI and HI stand for connInterval and Heartbeat interval, respectively.

The message transmission count in a BLE mesh network comprises two main components: i) data traffic, and ii) network maintenance traffic. In Bluetooth Mesh, the latter corresponds to Heartbeat and polling messages, whereas in 6BLEMesh it comprises link maintenance and routing messages. We next analyze how data and network maintenance traffic contribute to the total message count for both BLE mesh networking approaches. Bluetooth Mesh presents a greater number of data message transmissions than 6BLEMesh, since Bluetooth mesh uses (controlled) flooding, whereas the latter uses single-path routing. Furthermore, in Bluetooth Mesh, each message transmission is carried out thrice (i.e. once per advertising channel).

Regarding network maintenance traffic, in Bluetooth Mesh each node sends Heartbeat messages (which are forwarded network-wide) periodically. Therefore, the total rate of Heartbeat message transmissions is a function of N^2, where N denotes the number of

network nodes. In contrast, in 6BLEMesh, each node sends one message (which is not relayed) per connected neighbor every *connInterval*. Routing traffic is negligible in comparison, as in steady state and with default P2P-RPL settings, the time between consecutive routing protocol messages sent by a node to a neighbor is in the order of hours. Therefore, the 6BLEMesh network maintenance message rate depends on the number of network links, N_{links}, which is smaller than N^2.

The described features of data and network maintenance traffic yield a message transmission count of Bluetooth Mesh that scales worse with network size and node density (i.e. a greater slope in Figure 74) than 6BLEMesh. However, if *connInterval* is set to low values (e.g. to achieve low end-to- end latency), the message count in 6BLEMesh is in several scenarios greater than that of Bluetooth Mesh, for the same data message period. This occurs mainly in networks with smaller size or node density.

Figures 74-76 depict the message count for Bluetooth Mesh and 6BLEMesh, for different settings in each case, and for DI equal to 1 s, 10 s and 100 s, respectively. In these figures, we have evaluated a large range of parameter settings for Heartbeat interval and *connInterval*, i.e. the main Bluetooth Mesh and 6BLEMesh parameters considering their impact on message count. As shown in these figures, message count tends to increase with network size, while it decreases with DI, HI and *connInterval*. However, in some cases, a network scenario with N_{Deg} of 7 leads to a lower message count than a smaller network size with N_{Deg} of 15.

Figure 75 shows that the message count in Bluetooth Mesh is almost constant in terms of HI, when DI is set to 1 s. However, Figure 76 illustrates how the message count tends to show greater difference for different HI settings when DI is set to 10 s. This phenomenon becomes amplified when DI is set to 100 s, as shown in Figure 77. The reason is that, for DI = 1 s, the $M_{Flooding}$ component is very high, compared to $M_{Heartbeat}$, in (5-1). As DI increases, the relative contribution of to $M_{Heartbeat}$ becomes more relevant.

Figure 75-76 also show how, for a given DI setting, network parameter settings cause up to thousand-fold difference in message count for 6BLEMesh, but significantly smaller difference for Bluetooth Mesh. This is mostly due to the impact of *connInterval* on the message count in 6BLEMesh, as this parameter may take values from 7.5 ms, leading to very high message count, up to 32 s (when considering its highest equivalent value), which reduces the message count. In fact, for the settings of DI considered, 6BLEMesh shows both the maximum and the minimum message count, depending on the *connInterval* setting.

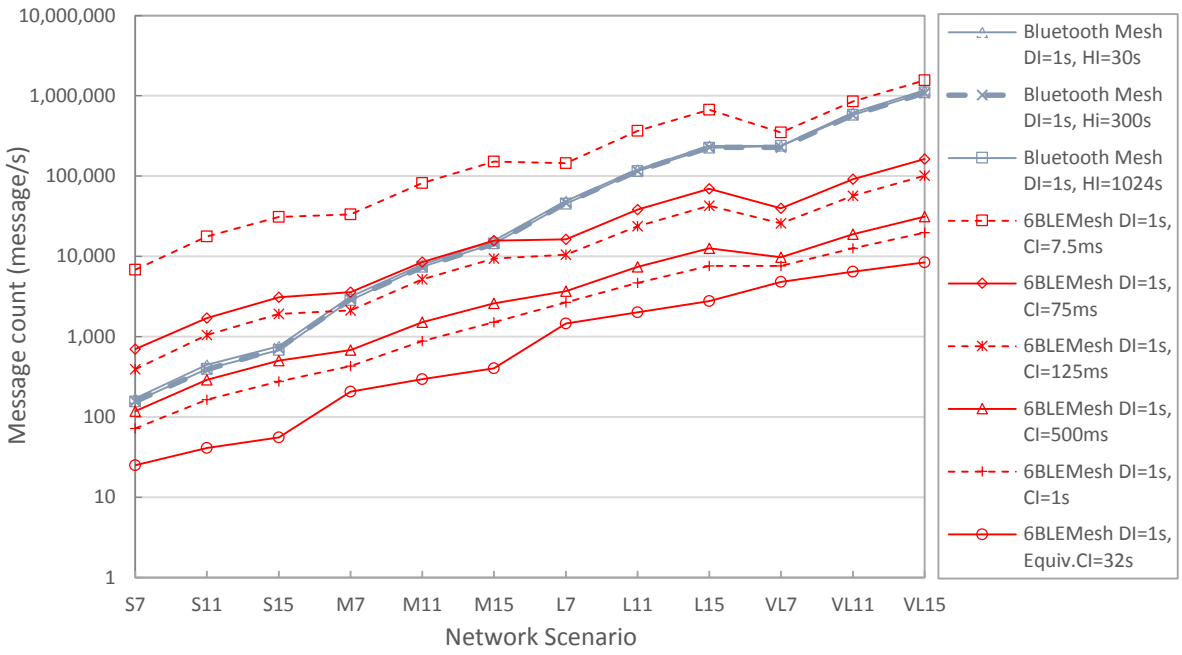


Figure 75: Comparing the message count in Bluetooth Mesh and 6BLEMesh for different network sizes when Data Interval = 1 s.

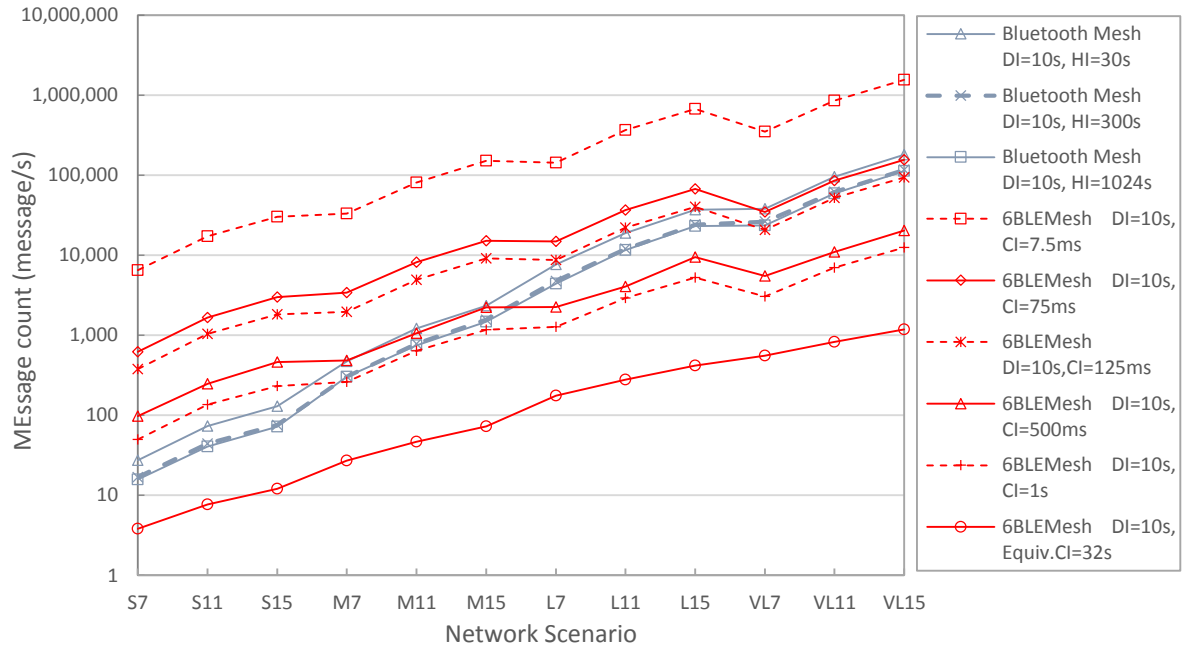


Figure 76: Comparing message count in Bluetooth Mesh and 6BLEMesh based approaches in different network sizes when Data Interval = 10 s.

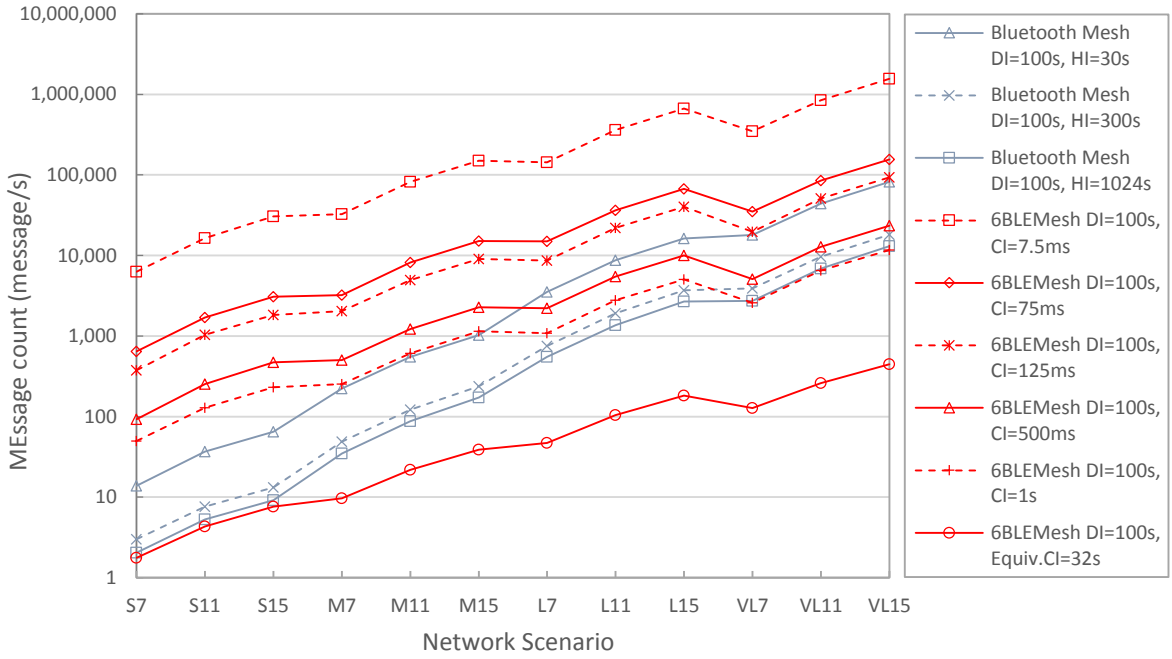
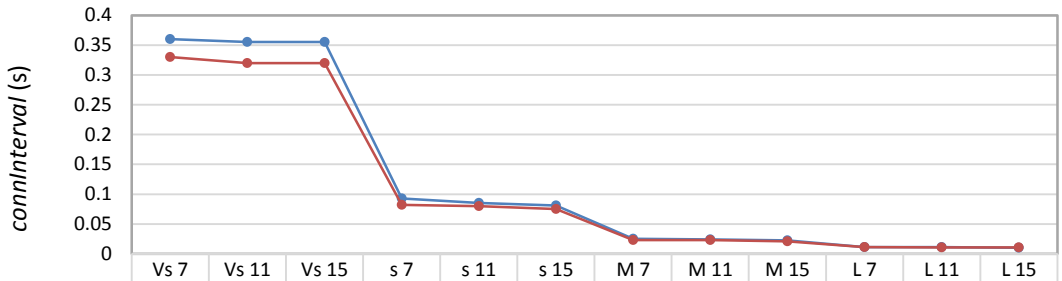


Figure 77: Comparing message count in Bluetooth Mesh and 6BLEMesh based approaches in different network sizes when Data Interval = 100 s.

Another useful study is determining, for each network topology, which is the *connInterval* setting that leads both Bluetooth Mesh and 6BLEMesh to the same message count. We call such setting *critical connInterval*. We assumed HB=30 s and 1024 s for all network topologies. Figure 78-79 illustrate the *critical connInterval* value for DI settings of 1 s, 10 s, and 100 s, respectively. As shown in Figure 78-79, regardless of the DI setting, *critical connInterval* decreases with the network size increase. Another observation is that the effect of the HI setting on *critical connInterval* increases with DI and decreases with the network size increases. This happens because, as discussed earlier, the relative effect of Heartbeat messages on the total message count increases with DI, as $M_{Flooding}$ decreases with DI.

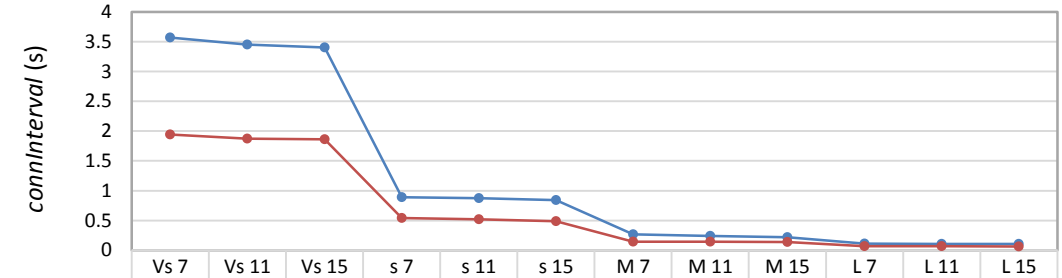
Figure 79 shows how *connInterval* can be set to equal the found *critical connInterval* values while keeping a *connSlaveLatency* value of 0 (for HI=1024 s in Bluetooth Mesh). Similarly, as shown in Figure *80*, for HI=30 s, DI should not be higher than 100 s to set *connInterval* to *critical connInterval* while keeping *connSlaveLatency* set to 0.

Finally, Figures 77-79 illustrate that, for a given network size, *critical connInterval* is relatively constant, although with a tendency to decrease with N_{Deg}, since Bluetooth Mesh offers worse scalability than 6BLEMesh.



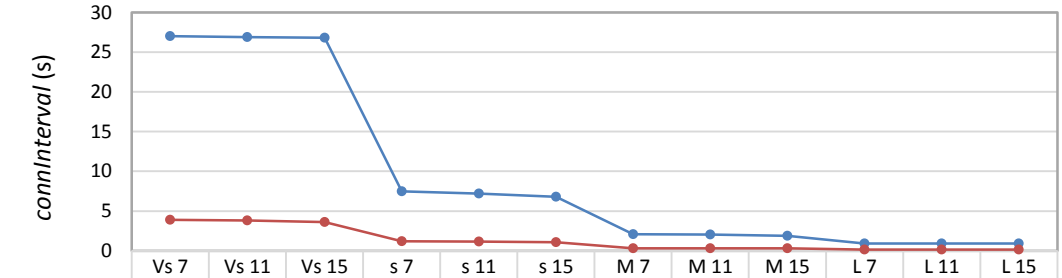
	Vs 7	Vs 11	Vs 15	s 7	s 11	s 15	M 7	M 11	M 15	L 7	L 11	L 15
HI= 1024 s	0.36	0.355	0.355	0.093	0.085	0.081	0.025	0.024	0.0225	0.011	0.011	0.01
HI= 30 s	0.33	0.32	0.32	0.082	0.08	0.075	0.023	0.023	0.021	0.011	0.0105	0.0105

Figure 78: critical connInterval for Data Interval = 1 s.



	Vs 7	Vs 11	Vs 15	s 7	s 11	s 15	M 7	M 11	M 15	L 7	L 11	L 15
HI= 1024 s	3.57	3.45	3.4	0.892	0.874	0.842	0.265	0.241	0.219	0.112	0.108	0.104
HI= 30 s	1.94	1.87	1.86	0.54	0.52	0.49	0.143	0.142	0.141	0.068	0.067	0.065

Figure 79: critical connInterval for Data Interval = 10 s.



	Vs 7	Vs 11	Vs 15	s 7	s 11	s 15	M 7	M 11	M 15	L 7	L 11	L 15
HI= 1024 s	27	26.9	26.8	7.5	7.2	6.8	2.1	2.05	1.9	0.92	0.91	0.91
HI= 30 s	3.9	3.8	3.6	1.2	1.15	1.1	0.33	0.33	0.32	0.14	0.14	0.14

Figure 80: critical connInterval for Data Interval = 100 s.

5.6. End-to-end reliability

Links in a BLE mesh network are prone to suffering bit errors, due to phenomena such as radio signal fading or interference, among others. Bluetooth Mesh and 6BLEMesh support different mechanisms intended to tackle this problem.

Bluetooth Mesh's flooding offers path diversity to each packet transmission. In contrast, 6BLEMesh typically uses single-path routing for unicast communication. Both

Bluetooth Mesh and 6BLEMesh support frequency diversity in different ways. In Bluetooth Mesh, each message is typically sent via the 3 advertising channels in parallel. In 6BLEMesh, Link layer retries are performed (if needed) over a frequency channel that is updated every connInterval, as long as the Link layer connection remains open. The maximum number of consecutive Link layer retries in a Link layer connection, denoted R, is configurable.

In order to illustrate the performance of Bluetooth Mesh and 6BLEMesh in the presence of bit errors, Figure 81 depicts the end-to-end packet delivery probability of both approaches for a network comprising M independent end-to-end paths of equal characteristics between a source and a destination, of N end-to-end uncorrelated hops each, and for a link delivery probability p of 0.6. As shown in Figure 81, Bluetooth Mesh requires path diversity in order to achieve high packet delivery performance, especially for long end-to-end paths. In contrast, 6BLEMesh approaches ideal packet delivery probability, as long as R is set to a high enough value (e.g. a 99% packet delivery probability is achieved for a 10-hop path for $R \geq 7$ and p=0.6), at the expense of latency increase.

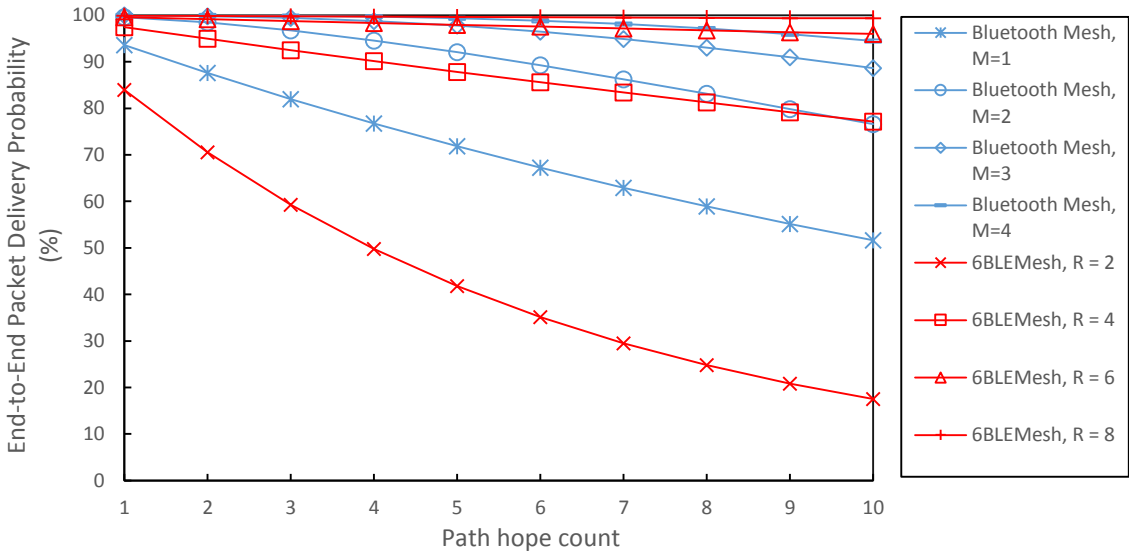


Figure 81: Packet delivery probability for Bluetooth Mesh and 6BLEMesh for a network comprising M independent N-hop paths between two endpoints and R retries, for p=0.6.

5.7. Variable topology robustness

A BLE mesh network exhibits a variable topology for several reasons, including node mobility or node failure. If a path being used in 6BLEMesh for end-to-end communication fails, an alternative path (if any) is only used after detection of the problem. In addition, some routing protocols may need to reactively discover an alternative path. In consequence, in 6BLEMesh, a topology change prevents end-to-end connectivity during significant time, typically in the order of at least several seconds. Instead, the flooding,

multipath approach in Bluetooth Mesh allows continuous end-to-end packet delivery, as long as an alternative path between the two communicating endpoints exists.

5.8. Internet connectivity

While 6BLEMesh naturally supports IPv6-based Internet connectivity, the Bluetooth Mesh standard does not. Therefore, connectivity of Bluetooth Mesh devices with the Internet requires a protocol translation gateway between the Bluetooth Mesh network and the Internet. The protocol translation gateway transforms message formats received on one interface to those used on the other one, and vice versa. While this solution is feasible, it limits application development scalability (since applications on the Bluetooth Mesh network side need to be designed specifically for Bluetooth Mesh communication services, and cannot be used over other technologies), it introduces issues of protocol consistency at both sides of the protocol translation gateway, and it precludes use of well-known IP-based tools and protocols for end-to-end connectivity, security and management.

5.9. Comparison: design goals, performance and main remarks

The different characteristics and performance of Bluetooth Mesh and 6BLEMesh are due to their respective design goals.

The main application domain for Bluetooth Mesh is smart home. In this domain, small network diameter (often up to 4 hops [90]) and good path diversity are expected. In such conditions, Bluetooth Mesh's flooding performs reasonably well in terms of message transmission count and link corruption robustness, while avoiding connectivity gaps due to topology changes. In contrast, 6BLEMesh was not created for a particular application area. 6BLEMesh follows a generic approach based on unicast routing on top of typically persistent Link layer connections. Thus, in 6BLEMesh, message transmission count and link corruption resiliency scale better with network size and density than Bluetooth Mesh, at the penalty of connectivity gaps after route failures.

Finally, note that intrinsic Internet connectivity support was not considered for Bluetooth Mesh, whereas it was a fundamental goal for 6BLEMesh leading to the IPv6-centric design of the latter.

Concluding the comparison, Bluetooth Mesh and 6BLEMesh offer fundamentally different BLE mesh networking solutions. Their performance depends significantly on their parameter configuration. Nevertheless, the following conclusions can be obtained. Bluetooth Mesh exhibits slightly greater protocol encapsulation overhead than 6BLEmesh. Both Bluetooth Mesh and 6BLEMesh offer flexibility to configure per-hop latency. For a given latency target, 6BLEMesh offers lower energy consumption. In terms of message transmission count, both solutions may offer relatively similar performance for small networks; however, 6BLEMesh scales better with network size and density.

6BLEMesh approaches ideal packet delivery probability in the presence of bit errors for most parameter settings (at the expense of latency increase), whereas Bluetooth Mesh requires path diversity to achieve similar performance. Bluetooth Mesh does not suffer the connectivity gaps experimented by 6BLEMesh due to topology changes. Finally, 6BLEMesh naturally supports IP-based Internet connectivity, whereas Bluetooth Mesh requires a protocol translation gateway.

6. Conclusions and future work

In this chapter, we provide the main conclusions from this PhD thesis, along with a number of future work directions. The chapter is organized in two sections. Section 6.1 provides the conclusions, emphasizing our main findings, innovations and contributions. Section 6.2 points out future work directions, offering a perspective of activities that may be carried out in order to expand scientific knowledge in the area.

6.1. Conclusions

This PhD thesis was motivated by the lack of standardized solutions for BLE mesh networking, along with the existence of gaps in the state of the art at a research level. During the timespan of this PhD thesis, the field has expanded and has become more mature, with two main standardization initiatives (Bluetooth Mesh, produced by the Bluetooth SIG, and 6BLEMesh, a proposal we are leading, which is being standardized by the IETF).

The contributions of this PhD thesis have mainly been made in four areas: i) surveying and creating a taxonomy of BLE mesh networking solutions, ii) evaluating the Bluetooth Mesh standard, iii) designing, evaluating and standardizing an IPv6-based BLE mesh network solution (i.e. 6BLEMesh), and iv) comparing the performance and characteristics of Bluetooth Mesh and 6BLEMesh.

This section provides the main conclusions from this PhD thesis. The section is organized into four subsections, which focus on the four areas mentioned in the previous paragraph, respectively.

6.1.1. Survey and taxonomy of BLE mesh networking solutions

During the initial stages of this PhD thesis, we made a comprehensive survey on existing BLE mesh networking proposals (see Chapter 2). We provided a taxonomy for BLE mesh network solutions, whereby we divided the solutions into three main categories, namely: academic solutions, proprietary solutions, and standardized solutions. We further classified the academic solutions into flooding-based, and routing-based, with the latter also divided into static-routing-based and dynamic-routing-based. After reviewing the BLE mesh techniques considered, we critically discussed their main advantages and drawbacks, and we presented open issues in areas that had not been deeply considered for BLE mesh network solutions. These included security, the effects of address assignment on privacy and routing performance, multicast, and interoperability.

6.1.2. Bluetooth Mesh evaluation

When the Bluetooth Mesh standard was published, we started analyzing and evaluating it in terms of several performance parameters.

In Chapter 3, we introduced a model to measure and predict the energy consumption of the energy-constrained devices in Bluetooth Mesh, that is, LPNs. To this end, we performed real current consumption measurements on a Nordic PCA10028 board. We identified the main device states relevant in terms of current consumption, and measured the time duration and average current consumption of each state. Based on this characterization, we created an analytical model, and we used it to predict useful performance parameters, such as device average current consumption, device lifetime and energy efficiency, considering the impact of the most relevant Bluetooth Mesh parameters, i.e. *PollTimeout* and *ReceiveWindow*, as well as the application parameters, DI.

We showed how current consumption decreases with *PollTimeout*, and increases with *ReceiveWindow.* We also found that the current consumption of data packet transmission tends to be negligible for DI values greater than 10 s. Moreover, we studied the effect of each parameter setting on total node lifetime and energy efficiency. We found a maximum device lifetime of 18 months, for a simple 235 mAh battery.

We also evaluated Bluetooth Mesh in terms of other performance parameters, in the context of a comparison with 6BLEMesh. Such parameters comprise protocol stack, protocol encapsulation overhead, end-to-end latency, energy consumption, message

transmission count, end-to-end reliability, and Internet connectivity. The results and findings in this area can be found in Chapter 5 (see also subsection 6.1.4).

6.1.3. 6BLEMesh design, evaluation and standardization

One of our main contributions is creating a new IPv6-based BLE mesh networking standard, called 6BLEMesh, which operates over established BLE Link Layer connections. To this end, we have extended RFC 7668, which specified IPv6 over star topology networks, by: i) restoring the 6LR node role, ii) adapting header compression to the mesh topology, iii) adding the requirement for a routing mechanism, and iv) using IPSP for Link Layer connection establishment in the context of a mesh network. Chapter 4 provides the details of the design and evaluation of 6BLEMesh, whereas Chapter 5 compares 6BLEMesh with Bluetooth Mesh.

After defining the characteristics and properties of 6BLEMesh, we evaluated it in terms of connectivity, latency, RTT, current consumption, energy efficiency, and device lifetime. Next, we summarize the main results regarding these characteristics.

Network connectivity is an essential requirement. We developed an analytical model for 6BLEMesh, since it is based on BLE Link Layer connections, that takes a set of network and scenario characteristics as inputs, and provides two main results: i) the probability of no isolation of a node, and ii) how strong is the *k*-connectivity of the considered network. We evaluated our analytical model, and validated it by simulation. Evaluation results were provided for different node density values.

We then implemented the 6BLEMesh standard on three different popular commercial hardware platforms, namely: Raspberry PI 3, Nordic Semiconductor nRF51 and Texas Instruments CC2650. We also built a three-node testbed consisting of all node types (i.e. 6LN, 6LR and 6LBR) using the latter, and evaluated a number of performance parameters on this testbed, related with latency and energy consumption.

We evaluated one-way latency, and we analyzed two different scenarios: i) when the sender is also the IPv6 packet source, and ii) when the sender just relays a received IPv6 packet. We showed that the average one hop, one way latency is *connInterval*/2. We also measured a two-hop RTT in the three-node testbed. We found experimentally that the RTT varies between 2·*connInterval* and 4·*connInterval*.

Next, we characterized the current consumption patterns of the complete life cycle for different node types in the three-node testbed. The different stages considered include node initialization, neighbor discovery, connection establishment, waiting for more connections (if any), a complete connection interval period, and a complete data interval period.

We also evaluated the energy performance of 6LN functionality on three different platforms: PCA10028, CC2650 LaunchPad and CC2650 SensorTag. We presented a 6LN current consumption model for different *connInterval* settings. To this end, we

considered connection intervals with and without IPv6 packet transmission. We experimentally characterized each current consumption state in terms of its duration time and average current consumption value. We illustrated how the average current consumption of 6LN devices decreases by increasing *connInterval*.

We experimentally found that the PCA10028 platform exhibits the lowest average current consumption, compared with the two other considered platforms.

We also evaluated the energy efficiency per delivered byte of a 6LN. We reported that this performance parameter increases linearly with DI from 1 s until 20 s and then exponentially for DI values higher than 20 s.

Finally, we studied 6LN device lifetime for the different device platforms considered, based on the previously derived average current consumption models. We found that, in contrast with *connInterval,* DI does not significantly impact on the total node lifetime. Maximum node lifetime, with a simple 235 mAh battery, is 573 days, for a PCA10028 device, when *connInterva*l is 4 s (and *connSlaveLatency* is 0). The node lifetimes on LaunchPad and SensorTag platforms under the same configuration are 521 days and 437 days, respectively.

6.1.4. Comparing Bluetooth Mesh and 6BLEMesh

Finally, we compared Bluetooth Mesh and 6BLEMesh, in terms of protocol stack, protocol encapsulation overhead, end-to-end latency, energy consumption, message transmission count, end-to-end reliability, and Internet connectivity.

With regard to protocol encapsulation overhead, both solutions offer similar performance (29 and 25 bytes for Bluetooth Mesh and 6BLEMesh, respectively).

Both approaches offer a range of latency features that depend on parameter settings, among others. The per-hop latency in non-LPN Bluetooth Mesh nodes is between 1 ms and 20 ms. When the destination is an LPN, per-hop latency is up to *PollTimeout*, which may be set between 1 s and 4 days. In 6BLEMesh, per-hop latency is between 0 and (equivalent) *connInterval*, where the latter may be set to values from 7.5 ms until 32 s. The routing protocol may incur additional delay in 6BLEMesh.

We also conducted an energy consumption comparison, focused on energy-constrained devices, that is, LPNs in Bluetooth Mesh and 6LNs in 6BLEMesh. In 6BLEMesh, current consumption decreases with *connInterval*. In Bluetooth Mesh, it decreases with *PollTimeout*, and also as *ReceiveWindow* decreases. For a given latency target, an energy-constrained device consumes less energy in 6BLEMesh than in Bluetooth Mesh. In fact, the operations carried out by a device in 6BLEMesh every *connInterval* (which include one receive and one transmit interval) consume less than 25 percent of the energy consumed in a poll action in Bluetooth Mesh (which includes three transmit intervals and one longer receive interval).

We then analyzed in detail the message overhead, in terms of total amount of messages per time unit, of Bluetooth Mesh and 6BLEMesh, assuming static topologies. To this end, we computed by simulation the amount of data messages, polling messages and Heartbeat messages in Bluetooth Mesh, and data messages, link maintenance messages and routing messages in 6BLEMesh. In Bluetooth Mesh, the largest contribution to the total message count comes from data message (controlled) flooding and Heartbeat message dissemination. In 6BLEMesh, the main contribution to the message count is link connectivity maintenance. We showed the impact of different network settings on the total message count for both solutions. Finally, we conducted a comprehensive message count study by simulation, considering scenarios of different sizes and node densities, as well as the impact of network parameters (*connInterval* and HI), along with the main application parameter (DI). We found that both solutions may offer relatively similar performance for small networks. However, 6BLEMesh scales better with network size and density, since it uses single-path routing, and the message count of link connectivity maintenance is lower than that of Heartbeat messages in Bluetooth Mesh.

We also compared the end-to-end reliability, in the presence of link corruption, of Bluetooth Mesh and 6BLEMesh. We showed that Bluetooth Mesh requires path diversity in order to achieve high packet delivery performance, especially for long end-to-end paths. In contrast, 6BLEMesh approaches an ideal packet delivery probability, as long as the number of link layer retries is set to a high enough value, at the expense of a latency increase.

Regarding robustness of the solutions in the context of variable topologies, we determined that Bluetooth Mesh is robust to topology changes, as it does not suffer the connectivity gaps incurred by routing in 6BLEMesh.

Finally, Internet connectivity is naturally supported by 6BLEMesh, due to its IPv6-centric design, whereas 6BLEMesh requires use of a protocol translation gateway for Internet connectivity.

6.2. Future work

This section provides a number of future work directions that stem from this PhD thesis. The suggested future research topics are: routing in Bluetooth Mesh, large scale experimental 6BLEMesh evaluation, evaluating different routing protocols for 6BLEMesh, and analysis of trade-offs in BLE mesh networks with Bluetooth 5.x.

6.2.1. Routing in Bluetooth Mesh

The Bluetooth Mesh specification uses controlled flooding as the technique to enable end-to-end data delivery in a multihop topology. As shown in this PhD thesis, controlled flooding has advantages and drawbacks. However, the Bluetooth Mesh

specification states that adding routing functionality may be considered for future versions of this standard. Therefore, an interesting future research direction is designing and evaluating routing protocols for Bluetooth Mesh. Since this standard mainly uses advertising bearers, the scenario is different from one based on the use of established link layer connections between neighbors.

6.2.2. Medium- and large-scale experimental 6BLEMesh evaluation

In this PhD thesis, we have designed and standardized 6BLEMesh. We have developed a prototype implementation, which has been evaluated in a three-node network. Considering the promising opportunities of IPv6-based BLE mesh networking, it will be interesting to experimentally evaluate 6BLEMesh in medium- and large-scale testbeds. Note that BLE mesh networking has applicability in areas such as smart home, smart factories, etc. Therefore, it will be important to determine the 6BLEMesh performance that can be expected in such scenarios.

6.2.3. Evaluating different routing protocols for 6BLEMesh

RPL is the IP-based routing protocol standardized by the IETF for IoT environments. Therefore, it is reasonable to consider it as the main routing protocol candidate for 6BLEMesh. However, RPL has often been criticized, and it has not been selected for relevant IP-based IoT protocol stacks, such as Thread. Accordingly, a future research work direction would be evaluating different routing protocols for 6BLEMesh. Such routing protocols could be based on RIP (as in Thread), AODV variants, or new protocols designed from scratch.

6.2.4. Analysis of trade-offs in BLE mesh networks with Bluetooth 5.x

For the first time in the BLE specifications, Bluetooth 5.0 introduced a variety of supported bit rates, ranging from 125 kbit/s to 2 Mbit/s. A BLE mesh network composed of Bluetooth 5.x devices offers additional flexibility options compared to Bluetooth 4.x networks. In the latter, a single bit rate (i.e. 1 Mbit/s) and thus a single link range is possible. However, in the former, different link ranges, along with different bit rates, are possible. In Bluetooth 5.x BLE mesh networks, path hop count between two endpoints may be low when using low bit rate links, since the link range will be greater. A low path hop count might appear to reduce end-to-end latency. However, the lower bit rate will contribute to increasing end-to-end latency. Also, a greater link range will increase network node density, which in some cases might have significant impact on performance. Analyzing all these trade-offs for different BLE mesh networking approaches is another promising future research work direction.

7. Contributions

Journal papers:

[I] S. M. Darroudi, C. Gomez, “Bluetooth Low Energy Mesh Networks: A Survey,” in *Sensors*, vol. 17, no. 7, pp. 1467, June **2017**.

[II] S. M. Darroudi and C. Gomez, "Modeling the Connectivity of Data-Channel-Based Bluetooth Low Energy Mesh Networks," in *IEEE Communications Letters*, vol. 22, no. 10, pp. 2124-2127, October **2018**.

[III] S. M. Darroudi, R. Caldera-Sànchez and C. Gomez, “Bluetooth Mesh Energy Consumption: A Model,” *Sensors*, vol. 19, no. 5, pp. 1238, March **2019**.

[IV] S. M. Darroudi, C. Gomez and J. Crowcroft, "Bluetooth Low Energy Mesh Networks: A Standards Perspective," in *IEEE Communications Magazine*, vol. 58, no. 4, pp. 95-101, April **2020**.

[V] S. M. Darroudi and C. Gomez, "Experimental Evaluation of 6BLEMesh: IPv6-based BLE Mesh Networks", submitted to Sensors, 2020 (in Major Revision).

Contributions to IETF standards:

[VI] C. Gomez, S. M. Darroudi, T. Savolainen and M. Spoerk, “IPv6 Mesh over BLUETOOTH(R) Low Energy using IPSP”, IETF Internet Draft, draft-ietf-6lo-blemesh-07, Dec. **2019** (work in progress).

Simulation and implementation code:

[VII] S. M. Darroudi, C. Gomez, 6BLEMesh connectivity simulation code:
https://sites.google.com/view/blemesh/home

[VIII] S. M. Darroudi, C. Gomez, Bluetooth Mesh message count simulation code:
https://sites.google.com/view/6blemesh/home

[IX] S. M. Darroudi, C. Gomez, 6BLEMesh message count simulation code:
https://sites.google.com/view/6blemesh/home

[X] S. M. Darroudi, C. Gomez, 6BLEMesh with Texas instruments CC2650 implementation code:
https://sites.google.com/view/6blemesh-implementation/home

Projects:

[XI] “ALLINONE” project, TEC2016-79988-P (funded by the Spanish Government).

[XII] “WINTERTIME” project PID2019-106808RA-I00 (funded by the Spanish Government).

[XIII] Grant 2017 SGR 376 (funded by Secretaria d'Universitats i Recerca del Departament d'Empresa i Coneixement de la Generalitat de Catalunya).

References

1. Kuor-Hsin, C. Bluetooth: a viable solution for IoT? *IEEE Wireless Communications* **2014**, *21*, 6-7.
2. Pie, Z.; Deng, Z.; Yang, B. In *Application-oriented wireless sensor network communication protocols and hardware platforms: A survey*,"in proc. of the" IEEE International Conference on Industrial Technology, 2008; Chengdu.
3. Haxhibeqiri, J.; De Poorter, E.; Moerman, I.; Hoebeke, J. A Survey of LoRaWAN for IoT: From Technology to Application. *Sensors* **2018**, *18*, 3995.
4. Casals, L.; Mir, B.; Aguilar, S.; Vidal, R.; Gomez, C. Modeling the Energy Performance of LoRaWAN. *Sensors* **2017**, *17*, 2364.
5. Rakić, A.; Popović, I.; Petruševski, I.; Begenišić, Đ.; Spajić, V.; Rakić, M. Key aspects of narrow band internet of things communication technology driving future IoT applications. In *25th Telecommunication Forum (TELFOR)*, IEEE: Belgrade, Serbia, 2017.
6. Gomez, C.; Veras, J.C.; Vidal, R.; Casals, L.; Paradells, J. A Sigfox Energy Consumption Model. *Sensors* **2019**, *19*, 681.
7. Tjensvold, J.M. Comparison of the IEEE 802.11, 802.15.1, 802.15.4 and 802.15.6 wireless standards. 2007.
8. Aguilar, S.; Vidal, R.; Gomez, C. Opportunistic Sensor Data Collection with Bluetooth Low Energy. *Sensors* **2017**, *17*, 159.
9. Gomez, C.; Oller, J.; Paradells, J. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors* **2012**, *12*, 11734-11753.
10. Gomez, C.; Paradells, J. Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine* **2010**, *48*, 92-101.
11. Group, T. Thread Usage of 6LoWPAN. Ver 2, Jul, 2015,
12. Gomez, C.; Paradells, J.; Eugenio Caballero, J. *Sensors Everywhere: Wireless Network Technologies and Solutions*. 978-84-934740-5-8: 2010.

13. Bluetooth SIG. *Specification of the Bluetooth System, Covered Core Package* Version 5.0,2016.
14. Bluetooth SIG. *Specification of the Bluetooth System, Covered Core Package* Version 4.0,2010.
15. Bluetooth SIG. *Specification of the Bluetooth System, Covered Core Package* Version 4.1,2013.
16. Bluetooth SIG. *Specification of the Bluetooth System, Covered Core Package* Version 4.2,2014.
17. Bluetooth SIG. *Specification of the Bluetooth System, Covered Core Package* Version 5.1,2019.
18. Bluetooth SIG. *Specification of the Bluetooth System, Covered Core Package* Version 5.2,2020.
19. Buetooth SIG. Bluetooth Consumer Electronics. Available online: https://www.bluetooth.com/what-is-bluetooth-technology/where-to-find-it/consumer-electronics (Accessed on: 20 June 2020),
20. Laird. Bluetooth Smart and Bluetooth Smart Ready. Available online: http://www.summitdata.com/blog/bluetooth-smart-bluetooth-smart-ready/ (Accessed on: 20 June 2020),
21. Bluetooth SIG. Adopted Specifications in Bluetooth 5. Available online: https://www.bluetooth.com/specifications/adopted-specifications (Accessed on: 20 June 2020),
22. Gogic, A.; Mujcic, A.; Ibric, S.; Suljanovic, N. Performance Analysis of Bluetooth Low Energy Mesh Routing Algorithm in Case of Disaster Prediction. *International Journal of Computer, Electrical, Automation, Control and Information Engineering* **2016**, *3*, 1075-1081.
23. Kim, H.-S.; Lee, J.; Jang, J.W. In *BLEmesh: A Wireless Mesh Network Protocol for Bluetooth Low Energy Devices*,"in proc. of the" International Conference on Future Internet of Things and Cloud (FiCloud), 3rd, 2015; IEEE: Rome.
24. Maharjan, B.K.; Witkowski, U.; Zandian, R. In *Tree network based on Bluetooth 4.0 for wireless sensor network applications*,"in proc. of the" European Embedded Design in Education and Research Conference (EDERC), 6th, 2014; IEEE: Milan.
25. Patti, G.; Leonardi, L.; Bello, L.L. In *A Bluetooth Low Energy real-time protocol for Industrial Wireless mesh Networks*,"in proc. of the" Annual Conference of the IEEE Industrial Electronics Society (IECON), 42nd, 2016; IEEE: Florence.
26. Mikhaylov, K.; Tervonen, J. In *Multihop data transfer service for Bluetooth Low Energy*,"in proc. of the" International Conference on ITS Telecommunications (ITST), 13th, 2013; IEEE: Tampere.
27. Sirur, S.; Juturu, P.; Gupta, H.P. In *A mesh network for mobile devices using Bluetooth low energy*,"in proc. of the" IEEE Sensors, 2015; Busan.
28. Reddy, Y.K.; Juturu, P.; Gupta, H.P.; Serikar, P.R.; Sirur, S.; Barak, S.; Kimy, B. In *Demo Abstract: A Connection Oriented Mesh Network for Mobile Devices using Bluetooth Low Energy*,"in proc. of the" ACM Conference on Embedded Networked Sensor Systems, 13th, 2015; ACM: Seoul, pp 453-454.
29. Guo, Z.; Harris, I.G.; Tsaur, L.-f. In *An on-demand scatternet formation and multi-hop routing protocol for BLE-based wireless sensor networks*,"in proc. of the" IEEE

Wireless Communications and Networking Conference (WCNC), 2015; IEEE: New Orleans.
30. Balogh, A.; Imre, S.; Lendvai, K. In *Service Mediation in multihop Bluetooth Low Energy networks based on NDN approach*,"in proc. of the" International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 23rd, 2015; IEEE: Bol.
31. Lee, T.; Lee, M.-S.; Kim, H.-S. In *A Synergistic Architecture for RPL over BLE*,"in proc. of the" Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 13th, 2016; IEEE: London.
32. CSRmesh. CSRmesh® Development Kit. Available online: http://www.csr.com/products/csrmesh-development-kit (Accessed on: 20 June 2020),
33. BLE-mesh.com. Available online: https://www.linkedin.com/company-beta/9264364/?pathWildcard=9264364 (Accessed on: 20 June 2020),
34. Nordic. Mesh networking platform uses Nordic Semiconductor. Available online: https://www.nordicsemi.com/eng/News/News-releases/Product-Related-News/Mesh-networking-platform-uses-Nordic-Semiconductor-nRF51822-SoCs-to-target-Internet-of-Things-applications (Accessed on: 20 June 2020),
35. NXP. NXP Bluetooth Smart Mesh. Available online: https://community.nxp.com/docs/DOC-329553 (Accessed on: 15 June 2017),
36. Silvair. Silvair Bluetooth mesh stack dedicated for lighting. Available online: https://www.silvair.com/#platform (Accessed on: 20 June 2020),
37. CYPRESS. Cypress Unveils New Bluetooth® Low Energy Modules and Bluetooth Smart Mesh Demo at CES. Available online: http://www.cypress.com/news/cypress-unveils-new-bluetooth-low-energy-modules-and-bluetooth-smart-mesh-demo-ces (Accessed on: 20 June 2020),
38. MESHTEK. ilumi MESHTEK homepage. Available online: https://ilumisolutions.com/ (Accessed on: 15 June 2017),
39. Estimote. Estimote homepage. Available online: http://estimote.com/?gclid=COKa0efjgNECFdW4GwodNccJeg (Accessed on: 20 June 2020),
40. WordPress.com. Telink Semiconductor (IoT Silicon), Tag:BLE mesh. Available online: https://telinkiotsilicon.wordpress.com/tag/ble-mesh/ (Accessed on: 20 June 2020),
41. Mindtree Mesh over Bluetooth low energy. Available online: https://www.mindtree.com/solutions/bluetooth-technology/blemesh (Accessed on: 20 June 2020),
42. Gogic, A.; Mujcic, A.; Ibric, S.; Suljanovic, N. In *Performance Analysis of Bluetooth Low Energy Mesh Routing Algorithm in Case of Disaster Prediction*,"in proc. of the" 18th International Conference on Wireless Communications, Mobile Computing and Networking, 2016; Vienna.
43. Levis, P.; Clausen, T.; Hui, J.; Gnawali, O.; Ko, J. The Trickle Algorithm. RFC 6206, Mar, 2011,
44. Scaglione, A.; Coates, M.; Gastpar, M.; Tsitsiklis, J.; Vetterli, M. Introduction to the Issue on Gossiping Algorithms Design and Applications. *IEEE Journal of Selected Topics in Signal Processing* **2011**, *5*, 645 - 648.

45. Vasseur, J.; Kim, M.; Pister, K.; Dejean, N.; Barthel, D. Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks. RFC 6551, March, 2012,
46. Winter, T.; Thubert, P.; Brandt, A.; Hui, J.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J.; Alexander, R. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March, 2012,
47. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. In *Networking named content*,"in proc. of the" Proceedings of the 5th international conference on Emerging networking experiments and technologies, 2009; Rome, pp 1-12.
48. Özsu, M.T.; Valduriez, P. Database Integration. In *Principles of Distributed Database Systems*, Springer: 2011; pp 131-170.
49. Hortelano, D.; Lopez, V.; Olivares, T. In *Poster Abstract: Improving the BLE Mesh Transmissions with User Collaboration in Smart Spaces Management*,"in proc. of the" IEEE International Conference on Information Processing in Sensor Networks (IPSN) (15th), 2016; IEEE: Vienna.
50. Zenker, P.; Krug, S.; Binhack, M. In *Evaluation of BLE Mesh capabilities: A case study based on CSRMesh*,"in proc. of the" International Conference on Ubiquitous and Future Networks (ICUFN), 8th, 2016; IEEE: Vienna.
51. Hortelano, D.; Olivares, T.; Ruiz, M.C.; Garrido-Hidalgo, C.; López, V. From Sensor Networks to Internet of Things. Bluetooth Low Energy, a Standard for This Evolution. *Sensors* **2017**, *17*, 372.
52. GitHub. nRF51-ble-bcast-mesh. Available online: https://github.com/NordicSemiconductor/nRF51-ble-bcast-mesh/ (Accessed on: 20 June 2020),
53. TelinkSemiconductor. TLSR8269F512 (BLE + IEEE802.15.4 Multi-Standard Wireless SoC). Available online: http://www.telink-semi.com/site/product_detail/65 (Accessed on: 15 June 2017),
54. Dierks, T.; Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, Aug, 2008,
55. NXP. *IEEE 802.15.4 Stack User Guide*;Version 2.6,2016.
56. Perkins, C.; Belding-Royer, E.; Das, S. Ad hoc On-Demand Distance Vector (AODV) Routing. July, 2003,
57. Maurya, P.K.; Sharma, G.; Sahu, V.; Roberts, A.; Srivastava, M. An Overview of AODV Routing Protocol. *International Journal of Modern Engineering Research (IJMER)* **2012**, *2*, 728-732.
58. Decuir, J. Bluetooth 4.0: Low Energy. In *Traning Document*, 2010.
59. Kim, H.-S.; Kumar, S.; Culler, D.E. Thread/OpenThread: A Compromise in Low-Power Wireless Multihop Network Architecture for the Internet of Things. *IEEE Communications Magazine* **2019**, *57*, 55-61.
60. Thread Technical Overview. Thread: 2015.
61. Han, S.N.; Cao, Q.H.; Alinia, B.; Crespi, N. Design, implementation, and evaluation of 6LoWPAN for home and building automation in the Internet of Things. In *12th International Conference of Computer Systems and Applications* IEEE: Morocco, 2015.
62. Thread Stack Fundamentals. Thread Technical white paper: 2015.

63. Thread. Thread Group website. Available online: https://www.threadgroup.org/technology/ourtechnology (Accessed on: 20 June 2020),
64. ANT+. ANT'S PRACTICAL MESH NETWORK SOLUTION SIMPLIFIES HOME AUTOMATION. Available online: https://www.thisisant.com/news/ants-practical-mesh-network-solution-simplifies-home-automation/ (Accessed on: 20 June 2020),
65. Hiertz, G.R.; Denteneer, D.; Max, S.; Taori, R.; Cardona, J.; Berlemann, L.; Walke, B. IEEE 802.11s: The WLAN Mesh Standard. *IEEE Wireless Communications* **2010** *17*, 104-111.
66. Šljivo, A.; Kerkhove, D.; Tian, L.; Famaey, J.; Munteanu, A.; Moerman, I.; Hoebeke, J.; De Poorter, E. Performance Evaluation of IEEE 802.11ah Networks With High-Throughput Bidirectional Traffic. *Sensors* **2018**, *18*, 325.
67. Buetooth SIG. Bluetooth® Smart Mesh Working Group annoncement. Available online: https://www.bluetooth.com/news/pressreleases/2015/02/24/bluetoothtechnology-adding-mesh-networking-to-spur-new-wave-of-innovation (Accessed on: 20 June 2020),
68. Buetooth SIG. Bluetooth SIG 2016 technology roadmap. Available online: https://www.bluetooth.com/news/pressreleases/2015/11/11/bluetooth-technology-to-gain-longer-range-faster-speed-mesh-networking-in-2016 (Accessed on: 20 June 2020),
69. Nordic Semiconductor. nRF5 SDK for Mesh. Available online: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF51422 (Accessed on: 20 June 2020),
70. Hui, J.W.; Culler, D.E. Extending IP to Low-Power, Wireless Personal Area Networks. *IEEE Internet Computing* **2008**, *12*, 37-45.
71. Nieminen, J.; Savolainen, T.; Isomaki, M.; Patil, B.; Shelby, Z.; Gomez, C. IPv6 over BLUETOOTH(R) Low Energy. RFC 7668, Oct, 2015,
72. Shelby, Z.; Bormann, C. *6LoWPAN: The Wireless Embedded Internet*. Wiley: 2009.
73. Nieminen, J.; Gomez, C.; Isomaki, M.; Savolainen, T.; Patil, B.; Shelby, Z.; Xi, M.; Oller, J. Networking solutions for connecting bluetooth low energy enabled machines to the internet of things. *IEEE Network* **2014**, *28*, 83 - 90.
74. Gomez, C.; Darroudi, S.M.; Savolainen, T.; Spoerk, M. IPv6 Mesh over BLUETOOTH(R) Low Energy using IPSP. March, 2017,
75. *Internet Protocol Support Profile V 1.0.0*;2014.
76. Thaler, D. Multi-Link Subnet Issues. RFC 4903, Jun, 2007,
77. Ko, J.; Terzis, A.; Dawson-Haggerty, S.; Culler, D.E.; Hui, J.W.; Levis, P. Connecting low-power and lossy networks to the internet. *IEEE Communications Magazine* **2011**, *49*, 96 - 101.
78. Malkin, G. RIP Version 2. RFC 2453, Nov, 1998,
79. Shelby, Z.; Chakrabarti, S.; Nordmark, E.; Bormann, C. Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). RFC 6775, Nov, 2012,
80. Hinden, R.; Deering, S. IP Version 6 Addressing Architecture. RFC 4291, Feb, 2006,
81. Hui, J.; Thubert, P. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282, Sep, 2011,

82. Narten, T.; Nordmark, E.; Simpson, W.; Soliman, H. Neighbor Discovery for IP version 6 (IPv6). RFC 4861, Sep, 2007,
83. Tsao, T.; Alexander, R.; Dohler, M.; Daza, V.; Lozano, A.; Richardson, M. A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs). RFC 7416, Jan, 2015,
84. Bettstetter, C. On the minimum node degree and connectivity of a wireless multihop network. In *3rd ACM international symposium on Mobile ad hoc networking & computing*, 2002; pp 80–91.
85. Ling, Q.; Tian, Z. Minimum Node Degree and k-Connectivity of a Wireless Multihop Network in Bounded Area. In *IEEE Global Telecommunications Conference*, IEEE: Washington, DC, USA, 2007.
86. Penrose, M.D. On k-connectivity for a geometric random graph. *Random Structures & Algorithms* **1999**, *15*, 145.
87. DC-BMN simulation code. Available online: https://sites.google.com/view/blemesh/home (Accessed on: 20 June 2020),
88. Brandt, A.; Buron, J.; Porcu, G. Home Automation Routing Requirements in Low-Power and Lossy Networks. RFC 5826, Apr, 2010,
89. Goyal, M.; Baccelli, E.; Philipp, M.; Brandt, A.; Martocci, J. Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks. RFC 6997, Aug, 2013,
90. Ferrari, G.; Medagliani, P.; Di Piazza, S.; Martalò, M. Wireless Sensor Networks: Performance Analysis in Indoor Scenarios. *EURASIP Journal on Wireless Communications and Networking* **2007**.

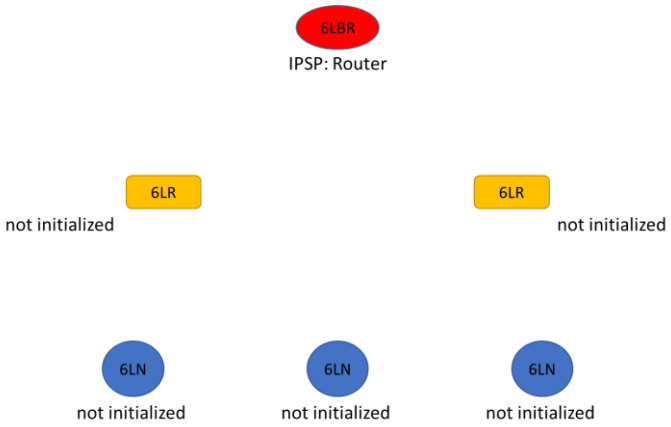
Appendix

A. Example of a 6BLEMesh network initialization procedure

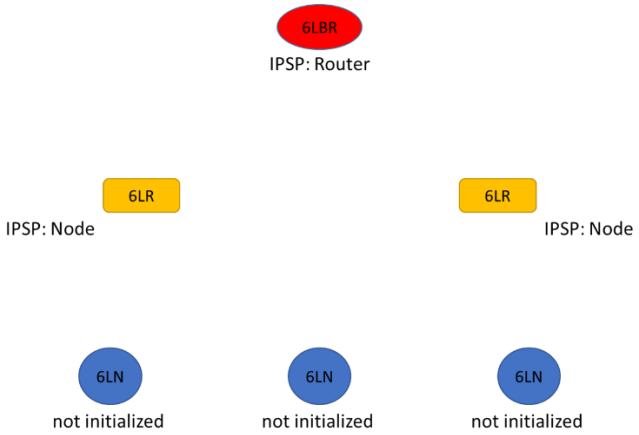
This appendix section offers an example of a procedure intended to enable 6BLEMesh network initialization, handling the establishment of Link Layer connections among neighboring nodes. Note that other sequences of events that may lead to the same final scenario are also possible.

At the beginning, the 6LBR starts running as an IPSP Router, whereas the rest of devices are not yet initialized (Step 1). Next, the 6LRs start running as IPSP Nodes, i.e., they use Bluetooth LE advertisement packets to announce their presence and support of IPv6 capabilities (Step 2). The 6LBR (already running as an IPSP Router) discovers the presence of the 6LRs and establishes one Bluetooth LE connection with each 6LR (Step 3). After establishment of those link layer connections (and after reception of Router Advertisements from the 6LBR), Step 4, the 6LRs start operating as routers, and also initiate the IPSP Router role (note: whether the IPSP Node role is kept running simultaneously is an implementation decision). Then, 6LNs start running the IPSP Node role (Step 5). Finally, the 6LRs discover presence of the 6LNs and establish connections with the latter (Step 6).

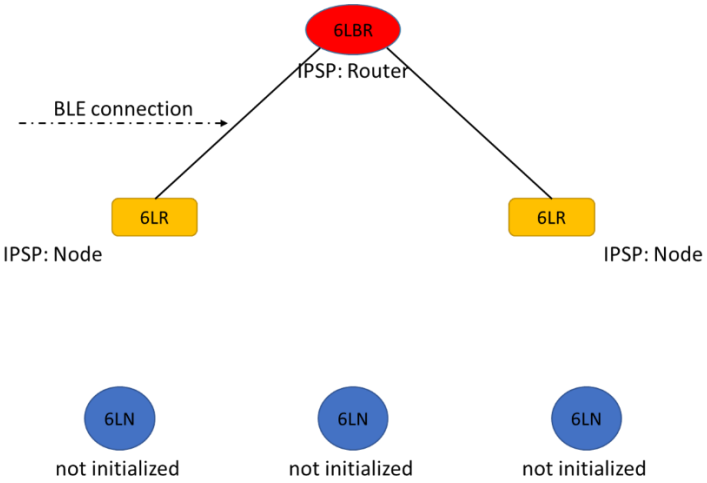
Step 1:



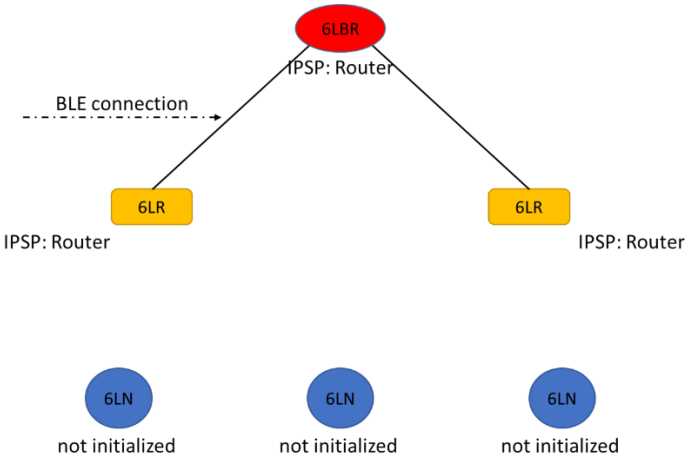
Step 2:



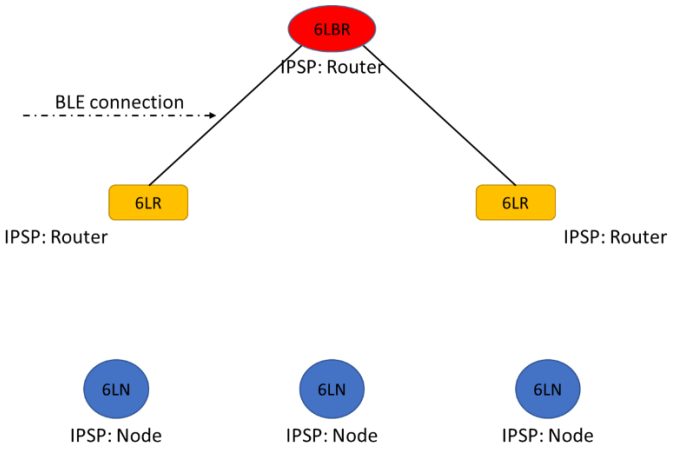
Step 3:



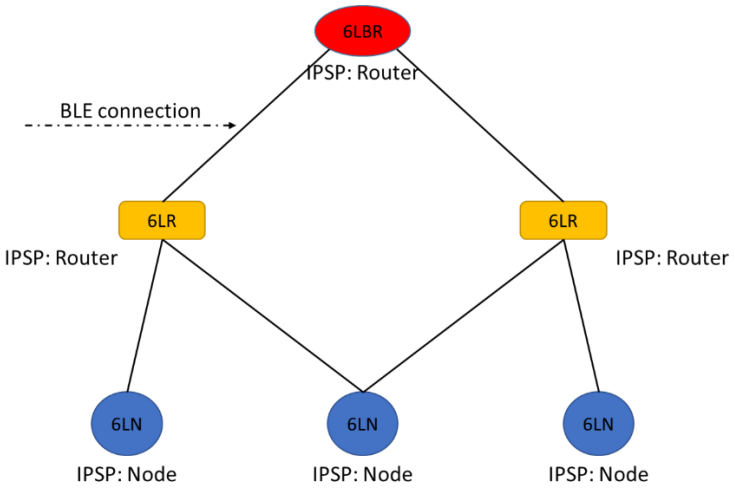
Step 4:



Step 5:

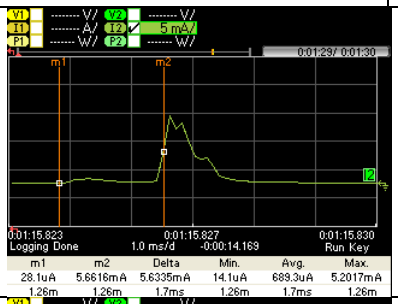
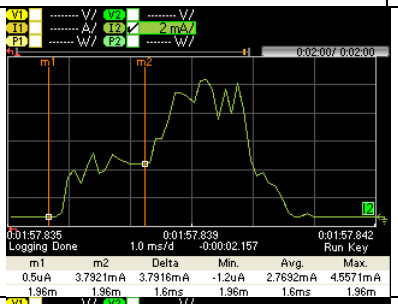
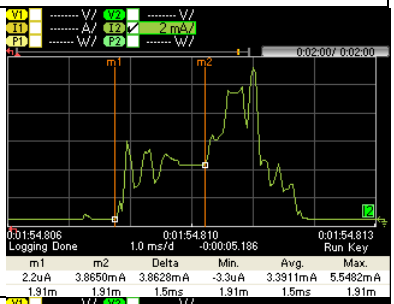
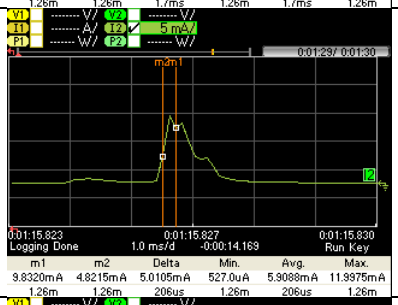
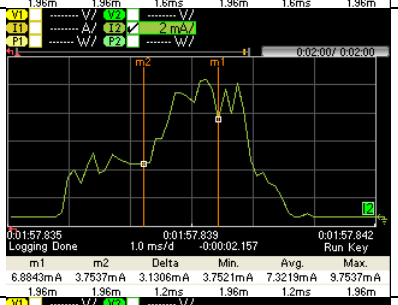
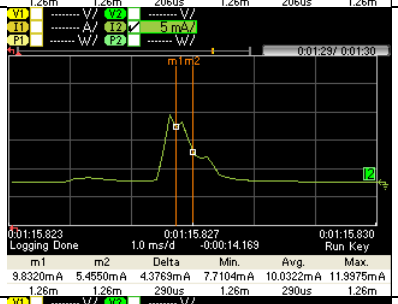
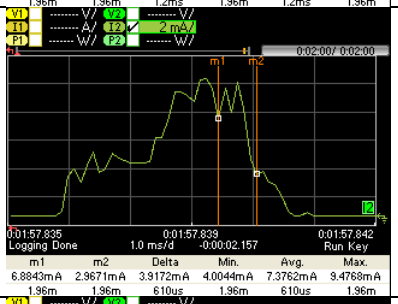
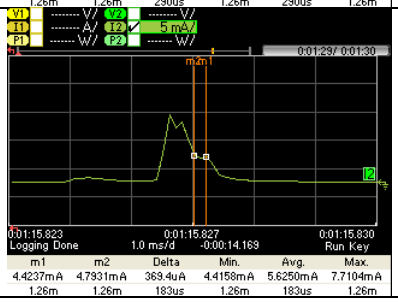
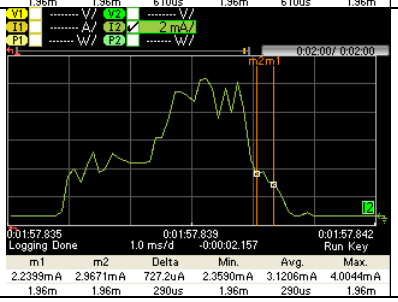
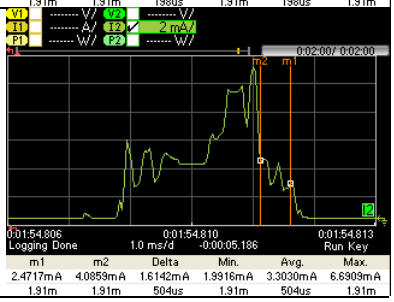


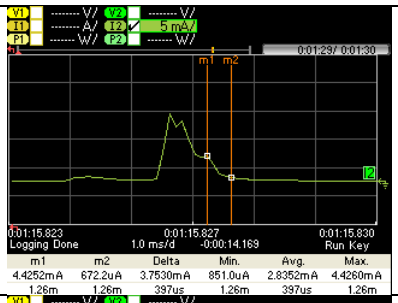
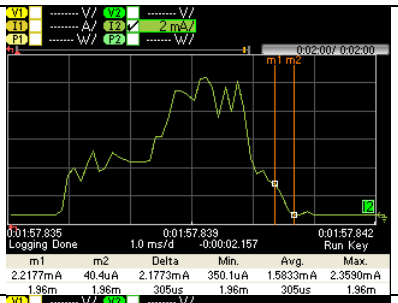
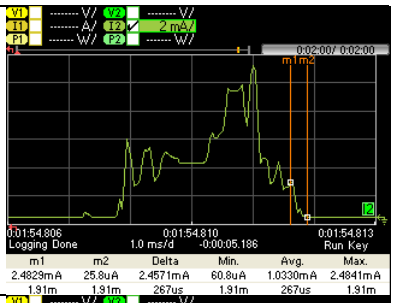
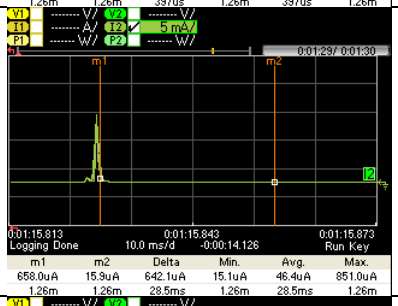
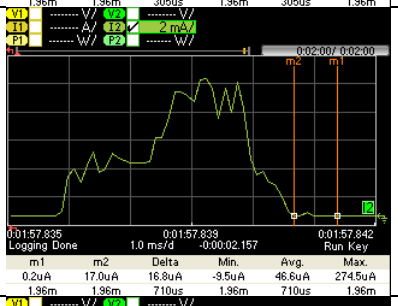
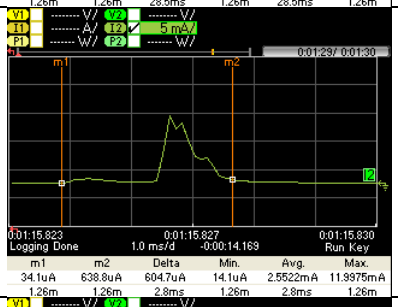
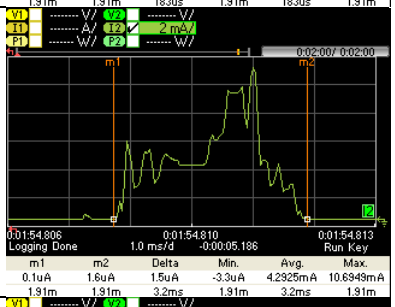
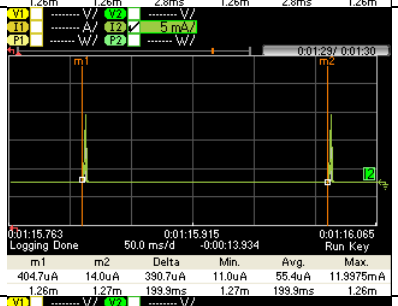
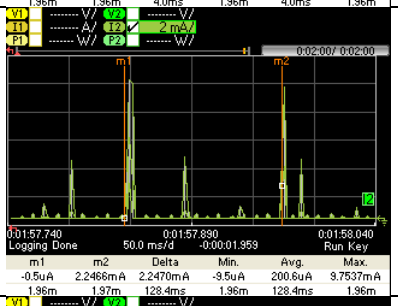
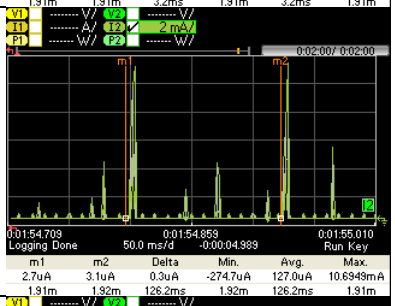
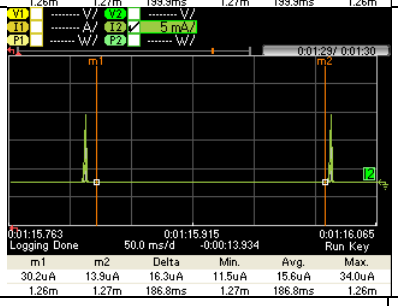
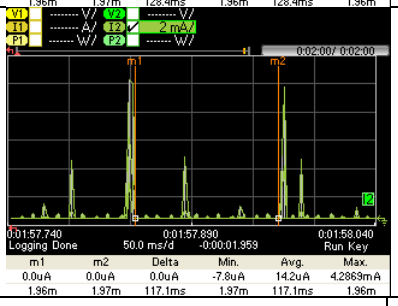
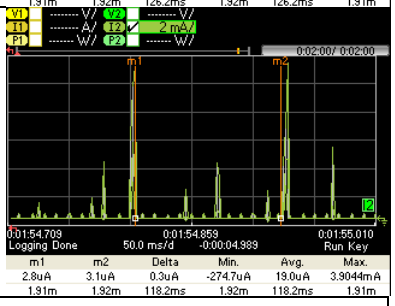
Step 6:


6LBR
IPSP: Router
BLE connection
6LR
6LR
IPSP: Router
IPSP: Router
6LN
6LN
6LN
IPSP: Node
IPSP: Node
IPSP: Node

B. Current consumption pattern of the active part of a connection event (without data)

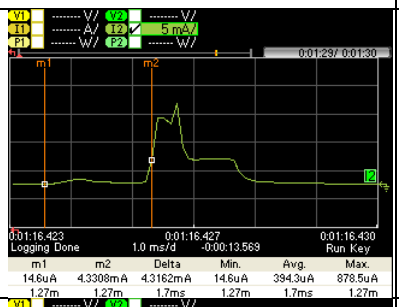
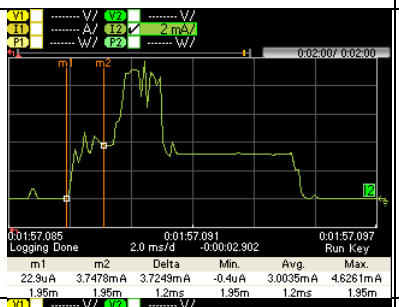
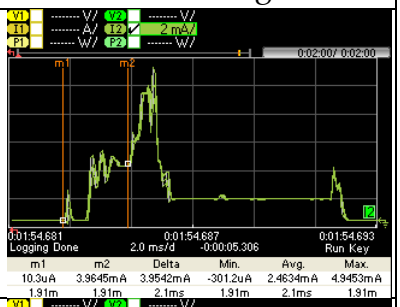
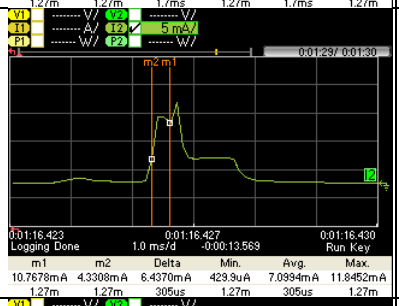
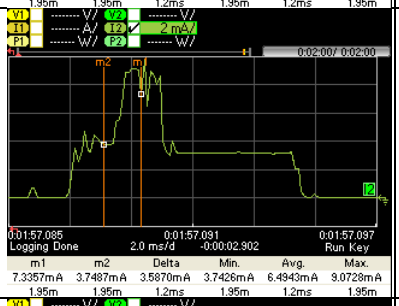
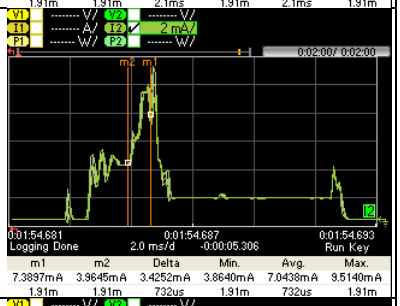
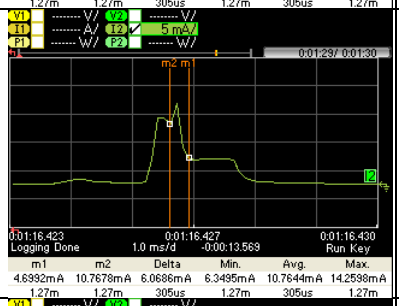
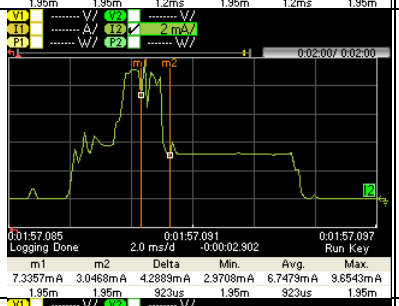
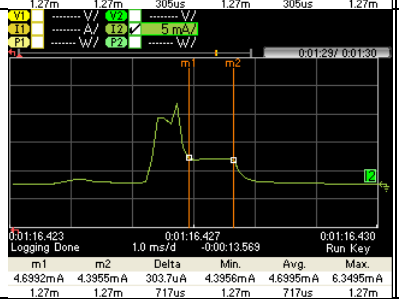
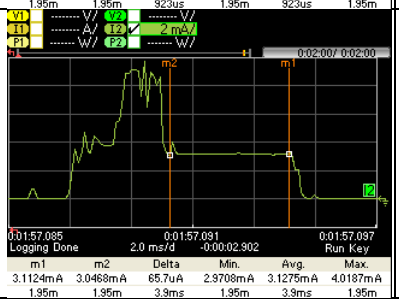
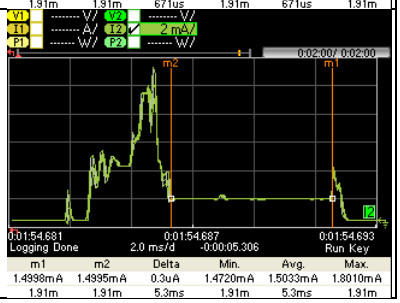
This appendix section provides the current consumption patterns for the active part of a connection event (where a data packet is not communicated), for the Nordic, LaunchPad and SensorTag devices used in our 6LN evaluation.

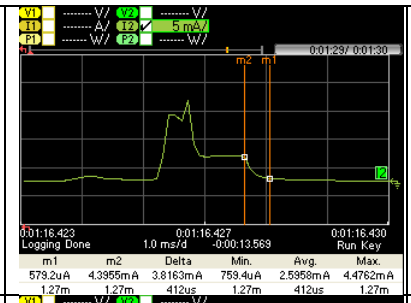
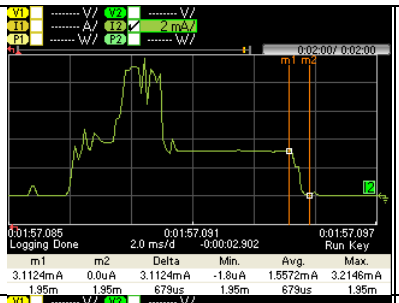
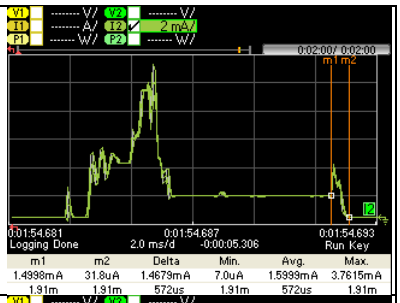
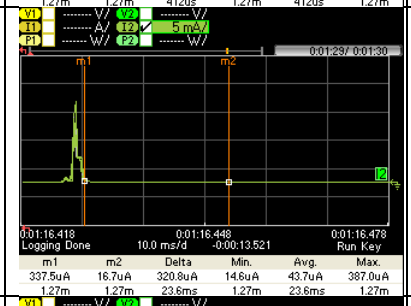
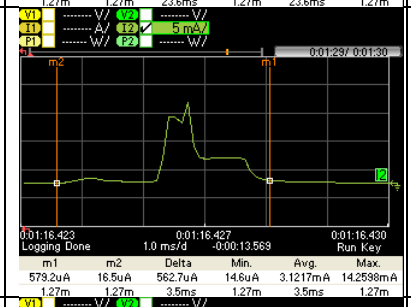
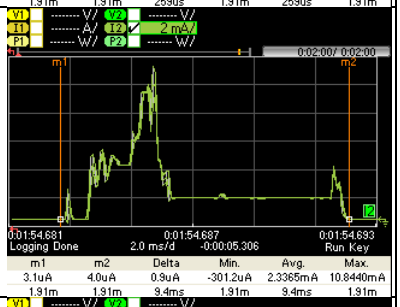
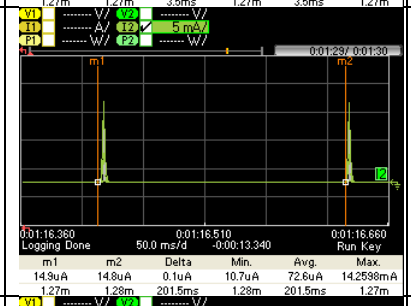
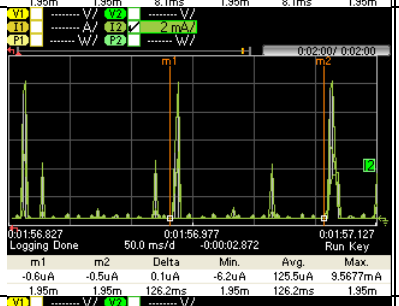
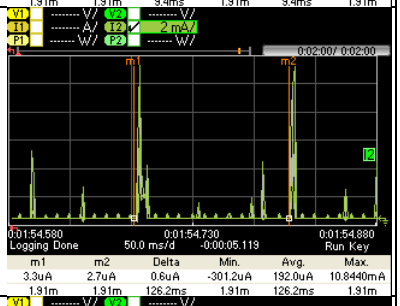
State number	Nordic	LaunchPad	SensorTag
15			
16			
17			
18			

19	 5 mA/ 0:01:29/ 0:01:30 m1 m2 0:01:15.823 0:01:15.827 0:01:15.830 Logging Done 1.0 ms/d -0:00:14.169 Run Key m1 m2 Delta Min. Avg. Max. 4.4252mA 672.2uA 3.7530mA 851.0uA 2.8352mA 4.4260mA 1.26m 1.26m 397us 1.26m 397us 1.26m	 2 mA/ 0:02:00/ 0:02:00 m1 m2 0:01:57.835 0:01:57.839 0:01:57.842 Logging Done 1.0 ms/d -0:00:02.157 Run Key m1 m2 Delta Min. Avg. Max. 2.2177mA 40.4uA 2.1773mA 350.1uA 1.5833mA 2.3590mA 1.96m 1.96m 305us 1.96m 305us 1.96m	 2 mA/ 0:02:00/ 0:02:00 m1 m2 0:01:54.806 0:01:54.810 0:01:54.813 Logging Done 1.0 ms/d -0:00:05.186 Run Key m1 m2 Delta Min. Avg. Max. 2.4823mA 25.8uA 2.4571mA 60.8uA 1.0330mA 2.4841mA 1.91m 1.91m 267us 1.91m 267us 1.91m
20	 5 mA/ 0:01:29/ 0:01:30 m1 m2 0:01:15.813 0:01:15.843 0:01:15.873 Logging Done 10.0 ms/d -0:00:14.126 Run Key m1 m2 Delta Min. Avg. Max. 658.0uA 15.9uA 642.1uA 15.1uA 46.4uA 851.0uA 1.26m 1.26m 28.5ms 1.26m 28.5ms 1.26m	 2 mA/ 0:02:00/ 0:02:00 m2 m1 0:01:57.835 0:01:57.839 0:01:57.842 Logging Done 1.0 ms/d -0:00:02.157 Run Key m1 m2 Delta Min. Avg. Max. 0.2uA 17.0uA 16.8uA -9.5uA 46.6uA 274.5uA 1.96m 1.96m 710us 1.96m 710us 1.96m	 2 mA/ 0:02:00/ 0:02:00 m1 m2 0:01:54.806 0:01:54.810 0:01:54.813 Logging Done 1.0 ms/d -0:00:05.186 Run Key m1 m2 Delta Min. Avg. Max. 63.0uA 4.9uA 64.1uA -4.2uA 27.8uA 75.0uA 1.91m 1.91m 183us 1.91m 183us 1.91m
	 5 mA/ 0:01:29/ 0:01:30 m1 m2 0:01:15.823 0:01:15.827 0:01:15.830 Logging Done 1.0 ms/d -0:00:14.169 Run Key m1 m2 Delta Min. Avg. Max. 34.1uA 638.8uA 604.7uA 14.1uA 2.5522mA 11.9975mA 1.26m 1.26m 2.8ms 1.26m 2.8ms 1.26m	 2 mA/ 0:02:00/ 0:02:00 m1 m2 0:01:57.835 0:01:57.839 0:01:57.842 Logging Done 1.0 ms/d -0:00:02.157 Run Key m1 m2 Delta Min. Avg. Max. 0.0uA 17.0uA 17.0uA -1.2uA 4.7092mA 9.7537mA 1.96m 1.96m 4.0ms 1.96m 4.0ms 1.96m	 2 mA/ 0:02:00/ 0:02:00 m1 m2 0:01:54.806 0:01:54.810 0:01:54.813 Logging Done 1.0 ms/d -0:00:05.186 Run Key m1 m2 Delta Min. Avg. Max. 0.1uA 1.8uA 1.5uA -3.3uA 4.2925mA 10.6949mA 1.91m 1.91m 3.2ms 1.91m 3.2ms 1.91m
	 5 mA/ 0:01:29/ 0:01:30 m1 m2 0:01:15.763 0:01:15.915 0:01:16.065 Logging Done 50.0 ms/d -0:00:13.934 Run Key m1 m2 Delta Min. Avg. Max. 404.7uA 14.0uA 390.7uA 11.0uA 55.4uA 11.9975mA 1.26m 1.27m 199.9ms 1.27m 199.9ms 1.26m	 2 mA/ 0:02:00/ 0:02:00 m1 m2 0:01:57.740 0:01:57.890 0:01:58.040 Logging Done 50.0 ms/d -0:00:01.959 Run Key m1 m2 Delta Min. Avg. Max. -0.5uA 2.2466mA 2.2470mA -9.5uA 200.6uA 9.7537mA 1.96m 1.97m 128.4ms 1.96m 128.4ms 1.96m	 2 mA/ 0:02:00/ 0:02:00 m1 m2 0:01:54.709 0:01:54.859 0:01:55.010 Logging Done 50.0 ms/d -0:00:04.989 Run Key m1 m2 Delta Min. Avg. Max. 2.7uA 3.1uA 0.3uA -274.7uA 127.0uA 10.6949mA 1.91m 1.92m 126.2ms 1.92m 126.2ms 1.91m
14	 5 mA/ 0:01:29/ 0:01:30 m1 m2 0:01:15.763 0:01:15.915 0:01:16.065 Logging Done 50.0 ms/d -0:00:13.934 Run Key m1 m2 Delta Min. Avg. Max. 30.2uA 13.9uA 16.3uA 11.5uA 15.6uA 34.0uA 1.26m 1.27m 186.8ms 1.27m 186.8ms 1.26m	 2 mA/ 0:02:00/ 0:02:00 m1 m2 0:01:57.740 0:01:57.890 0:01:58.040 Logging Done 50.0 ms/d -0:00:01.959 Run Key m1 m2 Delta Min. Avg. Max. 0.0uA 0.0uA 0.0uA -7.8uA 14.2uA 4.2869mA 1.96m 1.97m 117.1ms 1.97m 117.1ms 1.96m	 2 mA/ 0:02:00/ 0:02:00 m1 m2 0:01:54.709 0:01:54.859 0:01:55.010 Logging Done 50.0 ms/d -0:00:04.989 Run Key m1 m2 Delta Min. Avg. Max. 2.8uA 3.1uA 0.3uA -274.7uA 19.0uA 3.9044mA 1.91m 1.92m 118.2ms 1.92m 118.2ms 1.91m

C. Current consumption pattern of the active part of a connection event (with data)

This appendix section provides the current consumption patterns for the active part of a connection event (where a data packet is communicated), for the Nordic, LaunchPad and SensorTag devices used in our 6LN evaluation.

State #	Nordic	LuanchPad	Sensor Tag
22			
23			
24			
25			

26			
27			
			
			
21