Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

An Evaluation of Network Protocols for Bluetooth Low Energy Mesh Networks

Examensarbete utfört i Datorsystem
vid Tekniska högskolan vid Linköpings universitet
av

Oscar Hinrichsen

LiTH-ISY-EX--15/4898--SE

Linköping 2015

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

An Evaluation of Network Protocols for Bluetooth Low Energy Mesh Networks

Examensarbete utfört i Datorsystem
vid Tekniska högskolan vid Linköpings universitet
av

Oscar Hinrichsen

LiTH-ISY-EX--15/4898--SE

Handledare: **Marcus Karlsson, Jingcheng Zhang**
Examinator: **Danyo Danev**

Linköping, 27 oktober 2015

Avdelning, Institution
Division, Department

Research
Department of Electrical Engineering
SE-581 83 Linköping

Datum
Date

2015-10-27

Språk
Language

□ Svenska/Swedish
☒ Engelska/English
□ _____

Rapporttyp
Report category

□ Licentiatavhandling
☒ Examensarbete
□ C-uppsats
□ D-uppsats
□ Övrig rapport
□ _____

ISBN
—

ISRN
LiTH-ISY-EX--15/4898--SE

Serietitel och serienummer
Title of series, numbering

ISSN
—

URL för elektronisk version
http://www.ep.liu.se

Titel
Title

En utvärdering av nätverksprotokoll för Bluetooth Low Energy meshnätverk

An Evaluation of Network Protocols for Bluetooth Low Energy Mesh Networks

Författare
Author

Oscar Hinrichsen

Sammanfattning
Abstract

Internet of Things (IoT) is a scenario that theorizes objects and people as potential nodes in an ever-growing wireless network. This idea pushes the development of low-cost wireless technologies that can run on portable power sources for months, or even years. One candidate technique that has shown promising results in this area thru the last years is Bluetooth Low Energy (BLE). This thesis studies various techniques to enable and maintain large scale mesh networks over BLE communication. The initial study puts focus on an existing flooding based BLE mesh protocol. The thesis later presents an improved protocol that reduces power consumption with respect to the packet delivery ratio. Other enhancements which are added to the improved protocol are a self-adapting procedure and a packet routing algorithm. Simulations show that the improved protocol can save up to 50 % of the power consumption for a device, compared to the original protocol.

Nyckelord
Keywords

Bluetooth Low Energy, Mesh Networking

Abstract

Internet of Things (IoT) is a scenario that theorizes objects and people as potential nodes in an ever-growing wireless network. This idea pushes the development of low-cost wireless technologies that can run on portable power sources for months, or even years. One candidate technique that has shown promising results in this area thru the last years is Bluetooth Low Energy (BLE). This thesis studies various techniques to enable and maintain large scale mesh networks over BLE communication. The initial study puts focus on an existing flooding based BLE mesh protocol. The thesis later presents an improved protocol that reduces power consumption with respect to the packet delivery ratio. Other enhancements which are added to the improved protocol are a self-adapting procedure and a packet routing algorithm. Simulations show that the improved protocol can save up to 50 % of the power consumption for a device, compared to the original protocol.

Sammanfattning

Sakernas Internet (IoT) är ett scenario som skisserar objekt och människor som potentiella noder i ett ständigt växande trådlöst nätverk. Denna vision driver utvecklingen av trådlösa lågkostnadsteknologier som kan köras på portabla strömkällor i flera månader. En kandiderande teknik som har visat goda resultat inom detta område är Bluetooth Low Energy (BLE). Detta uppsatsarbete studerar flera tekniker för att möjliggöra och upprätthålla storskaliga meshnätverk över BLE-kommunikation. Den inledande studien granskar ett existerande översvämningsbaserat meshprotokoll för BLE. Uppsatsarbetet presenterar därefter ett förbättrat protokoll som reducerar strömförbrukningen med avseende på kvoten mellan antalet mottagna paket genom antalet skickade paket. Ytterliggare upprustningar som tillkommer i det förbättrade protokollet är en procedur för självanpassning, samt en algorithm för dirigering av paket. Simuleringar visar att det förbättrade protokollet kan spara upp till 50 % av strömkonsumptionen för en enhet, jämfört med originalprotokollet.

Acknowledgments

I would like to express my gratitude to the Ericsson LINLAB team and the Ericsson BLE research team in Kista for supporting me with my thesis. I would also like to dedicate special thanks to Jingcheng Zhang, Wei Shen, Thomas Rimhagen, Per Elmdahl and Mehdi Amirijoo for all special support during this year.

Besides my advisors at Ericsson, I would further like to thank Marcus Karlsson and Danyo Danev for helping me with this report.

Lastly, I thank my fiancée, my brother, and my parents for all the support, in good and bad times, during my education. You have all brought amiability and brilliance to me in so many ways.

Linköping, October 2015
Oscar Hinrichsen

Contents

Notation

Abbreviations

Abbreviation	Meaning
AD	Advertising Data
AODV	Ad Hoc On-Demand Distance Vector
APP	Application
ATT	Attribute Protocol
BLE	Bluetooth Low Energy
CDF	Cumulative Distribution Function
CRC	Cyclic Redundancy Check
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GFSK	Gaussian Frequency Shift Keying
GRXWIN	Global Reception Window
HCI	Host Controller Interface
ID	Identifier
ISM	Industrial, Scientific and Medial
L2CAP	Logical Link Control and Adaptation Protocol
LFSR	Linear Feedback Shift Register
LL	Link Layer
MIC	Message Integrity Check
MLID	Mesh Layer Identifier
NO	Number
NSMP	Non-Synchronized Mesh Protocol
OPCODE	Operation Code
PDR	Packet Delivery Ratio
PDU	Protocol Data Unit
PHY	Physical Layer
RFU	Reserved for Future Use
RREP	Route Discovery Reply
RREQ	Route Discovery Request
RSSI	Received Signal Strength Indication
RTC	Real-Time Clock
RXWIN	Reception Window
SCA	Sleep Clock Accuracy
SEQ	Sequence
SINR	Signal-to-Interference-Plus-Noise Ratio

Abbreviations

Abbreviation	Meaning
SMP	Synchronized Mesh Protocol (also Security Manager Protocol)
SNR	Signal-To-Noise Ratio
TB	Time Beacon
TBA	Time Beacon Acknowledgment
TRSP	Transport
TTL	Time-To-Live
TXWIN	Transmission Window

1 Introduction

This chapter gives a brief background, and presents the overall structure of this report. This chapter is further intended to give clarity of the motivation, purpose, and approach for the work that has been performed during this thesis.

1.1 Background

Bluetooth Low Energy (BLE) is a short range wireless technology targeted for ultra-low power consumption and manufacturing cost. A BLE device can typically run on a single coin cell battery for months, or even years, which allows the technology to be included in nearly any portable device. The technology used in BLE is also very simple which makes it appealing to many developers. Bluetooth Special Interest Group (SIG), the organization behind Bluetooth, predicts that at least 96 % of all mobile phones will support BLE by 2020 [Blu, 2014]. That is more than 7 billion devices according to forecasts of mobile phone consumptions [Ericsson, 2014]. On top of this, another 22 billion wireless devices are predicted in form of sport utilities, sensors, security systems, health monitors and much more [ABI, 2013]. Imagine controlling your TV, movie player and surround system from a single device; your smartphone. Then picture that you can use the same device to control lights, schedule your home equipment, turn on the car heater, or stream music to distributed speakers. This is the future that can be unlocked by BLE.

The BLE specification is however not perfect in every way. It has lately been discussed that BLE lacks proper ways to transmit data across a network when the two ends of the communication line are not directly connected [Branko, 2014] [Mikhaylov, 2013]. This severely hinders the expansion of the technology, princi-

pally because a BLE device cannot usually transmit data further away than about 30 m [Townsend, 2014, p. 9]. In order to solve this problem it has been proposed to investigate different techniques for establishing distributed BLE networks. The network should allow any two connected nodes to exchange data either via direct communication, or via diverse multi-hop propagation techniques. A network topology that has been especially requested to investigate is mesh (Figure 1.1) [Sch, 2010]. This exceptional topology presents many appealing features that will be discussed in detail in this report. One particular feature that is very well suited for mesh is so-called message flooding [Tanenbaum, 2011, pp. 368–370]. It will be displayed in this thesis that message flooding is an adequate keystone for the foundation of future BLE networks.

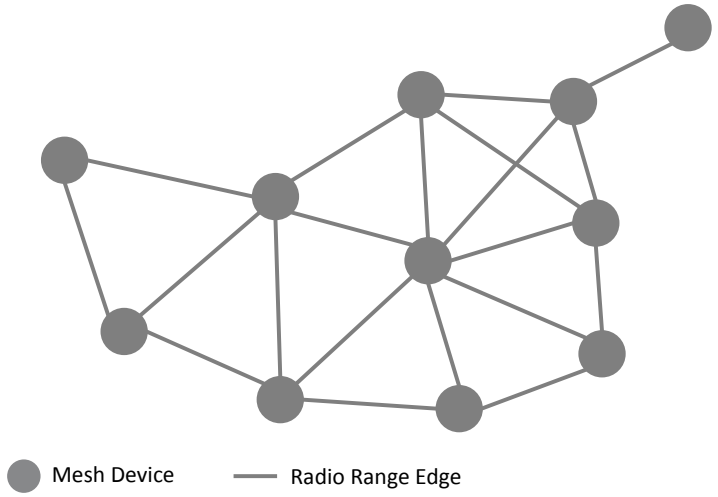


Figure 1.1: *A mesh network.*

1.2 Problem Formulation

The objective of this thesis is to construct and investigate different techniques that can be used with BLE to establish and maintain a mesh network. The initial work is to thoroughly study the BLE specification and investigate the state-of-the-art for BLE mesh protocols. The thesis will later focus on proposing a new protocol which will improve the network performance in aspect of power consumption and packet delivery ratio [Pankaj, 2013]. The thesis is aimed to answer the following questions:

1. *What protocols can be used to establish a mesh network using* BLE *communication?*

2. *How well do the individual mesh protocols work for various scenarios?*
3. *How much power is consumed by running the individual mesh protocols?*
4. *How long is the message latency for the individual mesh protocols?*
5. *Is there any support for multi-hop message routing in the individual mesh protocols?*

1.3 Methods

This report was written at Ericsson Research in Linköping. The contents of this report have been discussed with expert researchers at the site, along with employees of Ericsson Research in Kista. The technical background of this report is substantially based on the Bluetooth Low Energy core specification. All the presented results in this thesis were obtained from Ericsson's internal BLE simulator.

1.4 Thesis Overview

This thesis is divided into the following chapters:

Chapter 2 describes the architecture of BLE, and its limitations in the design of mesh network protocols.

Chapter 3 presents a simulation model for an existing BLE mesh protocol. This model will later be used to evaluate the performance of a device that implements the protocol.

Chapter 4 introduces a new BLE mesh protocol that will be compared to the simulation model in Chapter 3.

Chapter 5 describes the simulation model for the whole system.

Chapter 6 presents and discusses the results of the simulation.

Chapter 7 concludes the report and gives a summary of the protocol evaluation.

2

Bluetooth Low Energy

This chapter presents the fundamental concepts of the BLE architecture and brings up some important notions that will be used later in this report.

2.1 Protocol Architecture

The BLE architecture is usually divided into three parts: the *controller*, the *host* and *applications* (Figure 2.1). The controller is the lowest part of the architecture. It includes the *Physical Layer* (PHY), the *Link Layer* (LL), the *Direct Test Mode* and the *Host Controller Interface* (HCI). The middle part is the host. It contains the *Logical Link Control and Adaptation Protocol* (L2CAP), the *Attribute Protocol* (ATT), the *Security Manager Protocol* (SMP), the *Generic Attribute Profile* (GATT) and the *Generic Access Profile* (GAP). At the top there is the application part. This part usually includes notions like *Characteristics*, *Services* and *Profiles*.

2.1.1 Physical Layer

The Physical Layer (PHY) is the bottommost layer of the BLE protocol stack. This layer is responsible for transmitting and receiving bits using the radio device on the chip. BLE defines 40 RF channels on the 2.4 GHz ISM band, each with a 2 MHz separation. The signal is modulated using Gaussian frequency shift keying (GFSK) [Gerez, 2013], where 0 and 1 are represented by a negative and positive frequency deviation of 185 kHz, respectively. The bit rate of the transmitter must be 1 Mbit/s [Heydon, 2013, p. 28].

The 40 RF channels are further divided into advertising channels and data channels. The advertising channels are used to initialize connections, broadcast data and manage quick packet exchanges. The data channels are only used when two

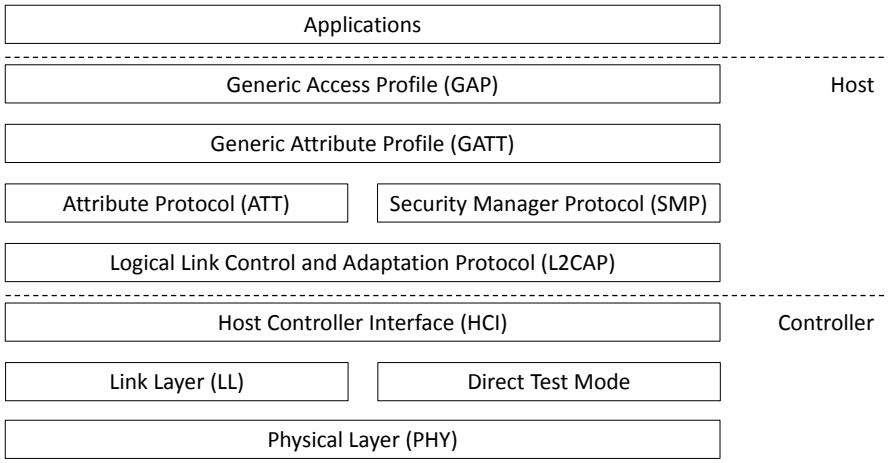


Figure 2.1: *The Bluetooth Low Energy stack architecture.*

devices have established a connection. The number of advertising channels in BLE has been set to three. These channels are however spread across the 40 RF channels in the frequency domain to avoid interference from other wireless technologies on the 2.4 GHz ISM band, such a Wi-Fi (Figure 2.2). The selected frequencies of the advertising channels are 2402 MHz, 2426 MHz and 2480 MHz. The data channels are put in between the advertising channels in the frequency domain [Heydon, 2013, pp. 84–87].

2.1.2 Direct Test Mode

There is a direct test mode defined in BLE which can be used to test the physical layer of a device. This allows different manufactures to test their physical layer without designing their own exclusive test methods. In the direct test mode the tester can analyze transmissions and receptions of packets using the higher layers of BLE. This feature is intended to reduce manufacture cost of BLE devices [Heydon, 2013, p. 29].

2.1.3 Link Layer

On top of the PHY, there is the Link Layer (LL). The LL is the heart of the BLE device; it controls the state of the radio device, handles data transportation, establishes and upholds connections, and maintains data privacy. Usually the LL is modeled as a state machine with five states: standby, advertising, scanning, initiating and connection (Figure 2.3). A LL can have multiple instances of the state machine.

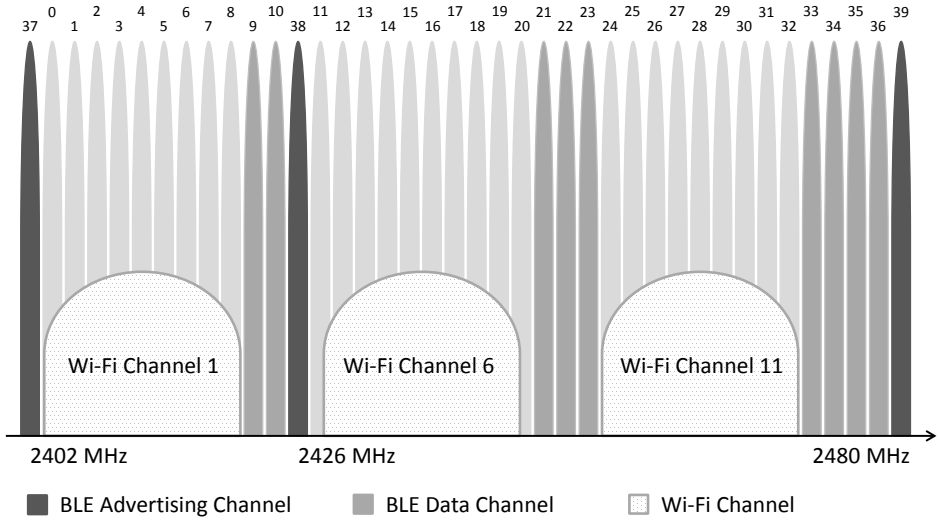


Figure 2.2: *Link Layer channel map and Wi-Fi channel coexistance.*

In the standby state the receiver and the transmitter are sleeping. The standby state can be entered from any other state.

The *advertising state* is used when the device transmits data in a disconnected state. This enables quick packet exchange and transmission without acknowledgment. It is also possible to broadcast data in this state. The advertising state is further used to notify peers that a device can be connected to. A packet that is transmitted in the advertising state is called an *advertising packet*. The only way to enter the advertising state is from the standby state. A device that is in the advertising state is called an *advertiser*.

The *scanning state* is when the device is receiving data without setting up any connection. If an advertising packet is received in the scanning state, then the device may respond if this is encouraged by the advertiser. The advertiser must then acknowledge the response so that a three packet exchange occurs. The only way to enter the scanning state is from the standby state. A device that is in the scanning state is called a *scanner*.

When setting up data connections, the *initiating state* is used. A device that is in the initiating state is called an *initiator*. A device enters the initiating state from the standby state and then starts to scan for advertising packets. If a *connectable advertising packet*[1] is received, then the initiator responds with an *initiating packet* and thereon enters *connection state*. When the advertiser receives the initiating packet it may choose to either go into connection state, or refuse

[1] Connectable advertising packets are a special class of advertising packets.

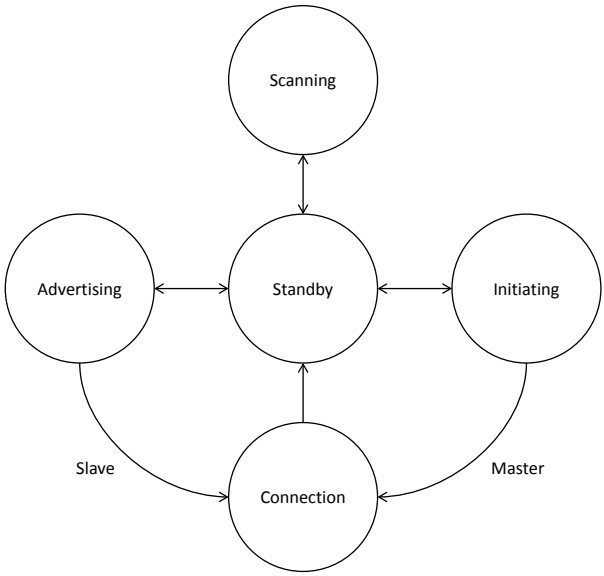


Figure 2.3: *The Link Layer (LL) state machine.*

the initiating packet and stay in the advertising state. If the advertiser rejects the connection, then the initiator will know this by the lack of feedback in the connection state. An initiator that goes into connection state will get the role as *master*. The advertiser that goes into connection state will get the role as *slave* [BLE, 2014a, pp. 30–32,39–45].

In the LL the RF channels are assigned unique indices. Due to several architectural issues it has been decided that the RF channels should be divided into two different types: advertising channels and data channels. The BLE data channels are assigned the indices 0 to 36. These channels are only used in the connection state. All other states are using the advertising channels for communication. The advertising channels are assigned the indices 37 to 39 [Heydon, 2013, pp. 84–87].

When entering the connection state, the slave and the master have already exchanged timing and channel information. This is included in the initiating packet. An event where two devices exchange data on a data channel is called a *connection event*. The first connection event will occur on a data channel selected by the initiator. At the end of a connection event, both devices will perform a frequency hop according to an agreed scheme [Heydon, 2013, pp. 120–121].

Data that is transmitted between two LL requires a special packet format that consists of 80 to 2120 bits. The lower limit is bounded by a mandatory overhead containing a preamble, access address, protocol data unit (PDU) and cyclic redundancy check (CRC). The preamble consists of 1 octet and is either 01010101b or 10101010b depending on the access address. If the least significant bit (LSB)

of the access address is 1 then 01010101b is used, otherwise 10101010b is used. The access address is 4 octets. The address must be 0x8E89BED6 when communicating via the advertising channels, otherwise a special random access address should be generated. The PDU is the actual payload of the message. It is mandatory that the payload is at least 2 octets regardless of the payload content. The first two octets are used for header information. The content of the header however differs between packets send on the advertising channel and the data channel. The final part of the packet structure is the CRC. The CRC is 3 octets and is used to identify bit errors in the packet [BLE, 2014a, pp. 38–39].

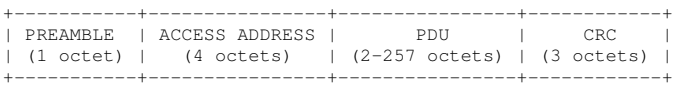


Figure 2.4: *The structure of a Link Layer packet.*

When sending PDUs over an *advertising channel*, the maximal size is 37 octets. However, the advertising channel PDU always starts with a 6 octet address field, consequently leaving only 31 octets for custom use. An advertising channel PDU is one of seven different types: indirect advertising, direct advertising, non-connectable advertising, scannable advertising, scan request, scan response or connection request. The connection request is commonly referred to as an initiating packet [BLE, 2014a, pp. 39–45].

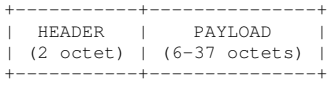


Figure 2.5: *The structure of an advertising channel PDU.*

Packets that are sent over a *data channel* are called data channel PDUs. A data channel PDU can contain up to 255 octets of information. The last 4 octets of the PDU are however reserved for an optional message integrity check (MIC), thus leaving 251 octets for custom use. Depending on the header of the packet, the payload can be either a LL control PDU or a LL data PDU. A control PDU is used to control the LL of the peer device, whereas a LL data PDU contains application data [BLE, 2014a, pp. 46–48].

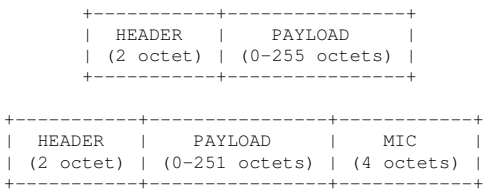


Figure 2.6: *A data channel PDU with and without message integrity check (MIC).*

When sending data from the LL to the PHY it is also required to perform whiten-

ing on the transmitted packet. This is to avoid long binary strings of one consistent symbol which is a problem for the GFSK modulation. The whitening is performed by using a linear feedback shift register (LFSR) representing the polynomial $x^7 + x^4 + 1$ (Figure 2.7). The first bit of the LFSR is set to one and the rest is set to the channel index [BLE, 2014a, pp. 59–60].

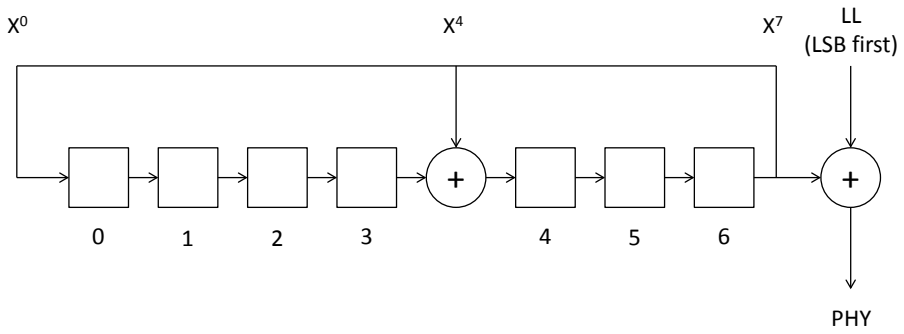


Figure 2.7: *The linear feedback shift register (LFSR) used for whitening.*

2.1.4 Host Controller Interface

The topmost layer of the controller is the Host Controller Interface (HCI). The HCI is used as a communication interface between the controller and the host within the BLE device. Typically the controller runs on a separate chip in the BLE device, whereas the host and applications can run on either a CPU or an application specific integrated circuit (ASIC). In either case it is often practical to separate the host and the controller, and use a transport interface such as SDIO, UART or USB to read and write data to the controller [Townsend, 2014, pp. 24–25]. Messages that are sent from the host to the HCI are called commands. Messages sent in the reversed direction are called events [BLE, 2014b, p. 962]. All HCI commands that are used in this report can be found in Chapter A.

2.1.5 Advertising Data

Data that is sent over the advertising channels must be arranged in a special format called *advertising data* (AD). The AD should be constructed of zero, one or many AD structures of at least 1 octet (Figure 2.8). Each AD structure usually contains a length, a type and some data [BLE, 2014c, p. 389].

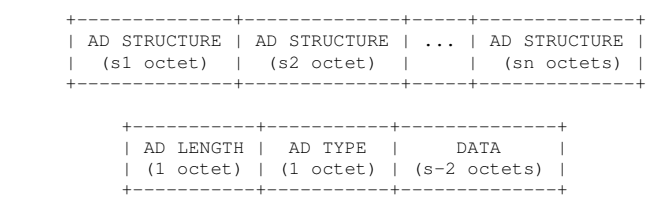


Figure 2.8: *The Advertising Data (AD) format and Advertising Data structure (AD structure).*

2.1.6 Logical Link Control and Adaptation Protocol

The Logical Link Control and Adaptation Protocol (L2CAP) is the lowest layer of the host part. The idea of this protocol layer is to enable wrapping of higher level protocols inside a BLE packet. Since the size of a BLE packet is limited to 2120 bits on the data channel, the L2CAP serves as a fragmentation/recombination protocol. Within the BLE architecture, the L2CAP is in charge of multiplexing two particular higher level protocols: the Attribute Protocol (ATT) and the Security Manager Protocol (SMP) [Townsend, 2014, p. 25].

2.1.7 Attribute Protocol

When two BLE devices are connected they can use the Attribute Protocol (ATT) to exchange data. ATT defines a client role and a server role, whereof at least one role should be selected in each device. In the server role the device listens to incoming requests from a client and simply acts as it is told. The client however will need to actively push/pull information to/from the server. Consequently, the complexity of the client is usually increased. [Heydon, 2013, p. 14]. Data exchange by ATT is done with six basic operations: command, confirmation, indication, notification, request and response (Figure 2.9).

A *command* is used to tell the server to perform a particular operation. This could for instance be that the client tells the server to turn on a light. Commands do usually not need any feedback via the data channel.

The *indication* is sent by the server when the state of the server is changed. When a client receives an indication it should respond to the server using a *confirmation*. ATT only allows one ongoing indication at the time. The server therefore has to wait for a confirmation before sending a new indication.

The server may also send state information that does not require any confirmation. In this case a *notification* should be sent. Notifications are therefore similar to commands regarding that they are also considered as an unreliable way to transmit data.

Usually a client and a server exchange data using *requests* and *responses*. This procedure is initiated by the client, using a request message. When the server obtains the request it should give the client a response. Just like indications there may be only one active request at the time [Heydon, 2013, pp. 217–219].

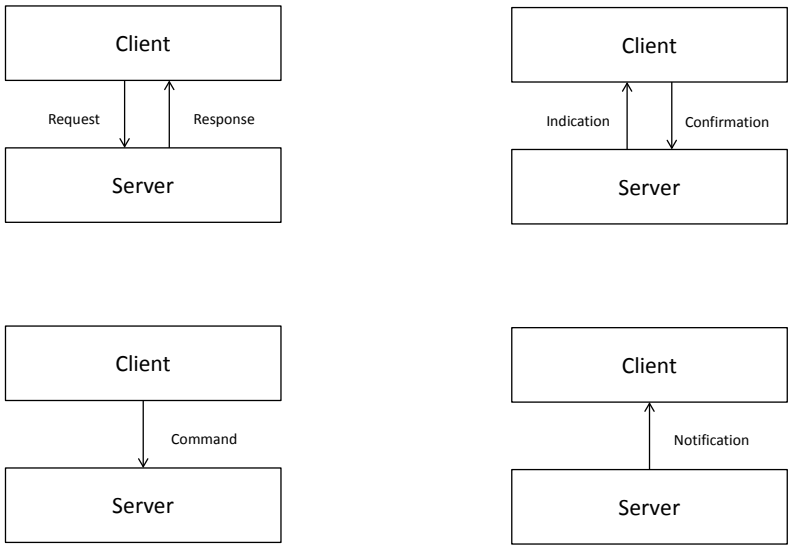


Figure 2.9: *The Attribute Protocols (ATT) operations.*

The ATT is used to read, write and discover attributes on the server. An attribute is a single field of information that tells the client about the current state of the server. The attribute could contain information about some value measured by the server (like temperature) or some property in the server device (like the current transmission power) [BLE, 2014c, p. 472].

2.1.8 Security Manager Protocol

A BLE device that requires secure data connections should implement a security manager. The security manager is using a key distribution approach to encrypt links between two devices [BLE, 2014c, p. 592]. When two security managers want to exchange key information, then they use the Security Manager Protocol (SMP). The SMP is completely separated from the ATT but they still share the same upper and lower protocol layers [BLE, 2014c, p. 633].

2.1.9 Generic Attribute Profile

Whereas the ATT layer describes how attributes are exchanged via a BLE connection, the Generic Attribute Profile (GATT) rather describes how the attributes are organized. In a server device, the attributes are represented as rows in a large table. Each attribute is associated with a unique 16-bit handle, a 16-bit to 128-bit UUID describing the type, a permission and a value.

The *attribute handle* is used as an address and is guaranteed not to change. Usually when referring to attributes in a higher-level context it is more representative to use a range of attributes addresses. Such a collection of addresses is called a

Handle	Type	Type	Permissions	Hex Value	Description/Value (Text)
0x1	0x2800	GATT Primary Service Declaration	R	18:00	Generic Access Service
0x2	0x2803	GATT Characteristic Declaration	R	02:03:00:00:2A	Device Name
0x3	0x2A00	Device Name	R	4D:79:20:42:4C:45:20:4B:65:79:62:6F:61:72:64	My BLE Keyboard
0x4	0x2803	GATT Characteristic Declaration	R	02:05:00:01:2A	Appearance
0x5	0x2A01	Appearance	R	C1:03	HID Keyboard
0x6	0x2803	GATT Characteristic Declaration	R	0A:07:00:02:2A	Peripheral Privacy Flag
0x7	0x2A02	Peripheral Privacy Flag	RW	00	Disabled
0x8	0x2803	GATT Characteristic Declaration	R	08:09:00:03:2A	Reconnection Address
0x9	0x2A03	Reconnection Address	W		
0xA	0x2803	GATT Characteristic Declaration	R		Peripheral Preferred Connection Parameters
0xB	0x2A04	Peripheral Preferred Connection Parameters	R	50:00:A0:00:00:00:00:03	Minimum Connection Interval = 80, Maximum Connection Interval = 160, Slave Latency = 0, Connection Supervision Timeout Multiplier = 768
0xC	0x2800	*GATT Primary Service Declaration*	*R*	*18:01*	*Generic Attribute Service*
0xD	0x2803	GATT Characteristic Declaration	R	20:0E:00:05:2A	Service Changed
0xE	0x2A05	Service Changed	I		
0xF	0x2902	Client Characteristic Configuration	RW	00:00	Disabled

Figure 2.10: *Table representation of data organized by the Generic Attribute Profile (GATT).*

handle range.

The *attribute type* is a unique identifier that is used to form a meaningful value of an attribute. The type could for instance indicate that an attribute value is an IEEE-754 64-bit floating point number.

The *permission* that is associated with an attribute tells the server how an attributes may be accessed. An attribute could have one of the following four permissions: none, readable, writable, readable and writable.

The *attribute value* is the actual data associated with the attribute. The value could contain any type of data; however maximal data size is restricted to 512 bytes [Townsend, 2014, pp. 51–56].

The GATT is also used to declare services for a particular device. A service is a handle range that includes useful attributes for a certain purpose. The service starts with a service declaration attribute, i.e. an attribute with the type *Primary Service* or *Secondary Service*. Services are fundamental building blocks of BLE profiles [BLE, 2014c, pp. 527–530].

In addition to keeping attributes organized, the GATT also defines a set of procedures. A procedure is a way of communicating with a certain profile using a high level interface. The GATT defines the following types of procedures: client-initiated, discovery and server-initiated. The *client-initiated procedures* are used to read and write attributes from/to the server using the ATT. *Discovery procedures*

can be called by the client to find services on the server device. *Server-initiated procedures* are used to send indications and notifications to a client. The GATT procedures can be combined to form more complex actions for profiles built on top of GATT [Heydon, 2013, pp. 231–240].

2.1.10 Generic Access Profile

Before any application data can be exchanged between two BLE devices, they first have to establish a connection between the higher layers of the protocol stack. The way of creating such a connection is by going through a series of basic steps. The Generic Access Profile (GAP) defines how this procedure is done, including whether or not bonding and encryption key exchange should be considered.

To start with, a BLE device sets its mode among the ones that are available in its current GAP role(s). A mode is a device behavior that spans for an unspecified amount of time. There are nine different modes that can be used: bondable, broadcast, direct-connectable, general-discoverable, limited-discoverable, non-bondable, nonconnectable, nondiscoverable and undirected-connectable. The mode defined in the GAP is strongly connected to the state of the LL state machine. Whereas the mode describes the current behavior of the device, the GAP role is more of a description of what the device can do. There are four roles defined in BLE: broadcaster, observer, peripheral and central. The peripheral and central role implicate that the device has a transmitter and receiver. A broadcaster must have a transmitter but may not have a receiver. An observer must have a receiver but may not have a transmitter.

The first step of establishing a higher layer connection is for a device to announce its presence. This is done by turning on *initial discoverability*, which in most cases uses the *limited-discoverable mode*. A device that activates limited-discoverability should be possible to distinct from other discoverable devices. To guarantee this, it has been proposed that a device may not be in limited-discoverable mode for more than about 30 seconds.

When a device is discovered by a peer, then it is up to the peer device to establish a connection. If a connection has not been established before, then it may be helpful for the peer device to obtain some primal information, such as the device name and the available services. Usually the peer device performs exhaustive search of all services, or looks for a particular service of interest.

In most scenarios for BLE, a low energy device will act in the peripheral role. It is then very common that the peripheral is associated with a single central, like a smartphone. In these cases where two devices are associated with each other, they are said to be bonded. Bonding may include that the devices exchange keys for encryption, identity resolving and connection signature resolving. When two bonded devices disconnect, they can use the stored bonding information to establish a new connection, much quicker than the first time [Heydon, 2013, pp. 255–263].

To perform all necessary steps from initialization to bonding, there is a funda-

mental set of procedures defined for the GAP. One suitable way of grouping these procedures is to divide them into four subsets: connection establishment, discovery, security and additional GAP procedures.

The *connection establishment procedures* are used to create a connection between a central and a peripheral. A connection can be established to an explicit set of whitelisted peripherals, or to any peripheral that is within range.

The *discovery procedures* enable detection of proximate peripherals. There are two ways of using these procedures: either to detect peripherals that are in limited-discoverable mode, or to detect peripherals that are in any discoverable mode.

The *security procedures* are used for authentication, authorization and encryption. These procedures usually communicate with the security manager of the device. Bonding is also a part of the security procedures.

The majority of the GAP procedures require that the invoking device is in a specific mode [Townsend, 2014, pp. 35–47].

2.1.11 Characteristics

From an application viewpoint, the characteristics are the bottommost entities of composed information that can be read from and written to a connected device. Each characteristic consists of a declaration, a value and a number of descriptors. The *declaration* is the header of the characteristic. It contains access information, a reference to the value and the data type that the value represents. A *descriptor* is used to add extra information to a characteristic, like a user text or a presentation format. Descriptors may also give additional access information that does not fit in the declaration. The characteristic value is basically a data type associated with some data content; very much like an attribute in the GATT. In fact, when it comes to the actual data representation of a characteristic it is nothing else but a set of attributes [BLE, 2014c, pp. 531–542].

When writing BLE applications, a common way to access peer data is by providing a GATT reference that can obtain characteristics and services. This way of accessing and writing data gives an abstraction that is easy to relate to object-oriented languages. Two application libraries that have chosen this exact approach are BluetoothAPIs (Windows) [Mic, 2015] and android.bluetooth (Android) [Goo, 2015].

2.1.12 Services

A service is a set of multiple characteristics that are composed to manage some specific task. There are two types of services defined in BLE: primary services and secondary services. A primary service exposes a functionality that can be useful for the peer device, e.g. a temperature service or an accelerometer service. A secondary service is only intended to be referred to by another service. The content of a secondary service may therefore be irrelevant outside the context of the service that references it.

In the GATT, a service starts with its service definition attribute. The service definition is then followed by a number of include definitions that refers to other services. The characteristics of the service must then immediately follow the last include definition [BLE, 2014c, pp. 527–530].

2.1.13 Profiles

A Profile describes how services and devices interact in the context of a use case. Most profiles require two devices, for instance a reporter and a monitor. The reporter provides data for the monitor, usually by defining one or many services. The reporter therefore usually acts as peripheral. The monitor may not require any services at all; in fact it might just fetch data from the reporter and then present it to the user. Typically the monitor is acting as central [Heydon, 2013, pp. 294–295].

The GATT defines profiles as the availability of some requested services [BLE, 2014d, pp. 100–101]. Imagine that a heart rate monitor wants to check if a sensor includes the *heart rate profile* (HRP). The monitor then requests the heart rate service and the device information service from the sensor, as they are required by the HRP. If these services are available (and additional requirements are met) then the monitor accepts the sensor as a heart rate sensor [Gupta, 2011].

2.2 Limitations

This subsection exposes some limitations of the BLE architecture, and how the limitations impact the design of BLE mesh protocols.

2.2.1 Network Topologies

When the first BLE specification (version 4.0) was published in 2010, there was an explicit restriction that a device cannot act as a master and slave simultaneously, even if multiple instances of the LL state machine where present. Furthermore, a device was not allowed to act as slave to more than one master at the same time. Consequently, the only supported network topology for BLE 4.0 is *star topology*. A peer-to-peer topology is considered to be a special case of a star topology [BLE, 2010, pp. 31–32]. The two stated restrictions were however dropped in 2013 when the second BLE specification was published (version 4.1). Using the newer specifications thereby allows BLE controller manufactures to include features for other network topologies; such as *scatternet* (Figure 2.11) [BLE, 2013, pp. 32–33].

When constructing a BLE mesh protocol, the scatternet topology might seem like a good selection. The fact is, however, that using a scatternet will complicate and deteriorate some essential procedures for a mesh network.

To begin with, two connected devices in a scatternet will communicate via the BLE data channels using the LL connection state. Each link in the network will then use a distinct pattern of physical channels (due to frequency hopping), and is unable to receive data from any other physical channel. This is good from an interfer-

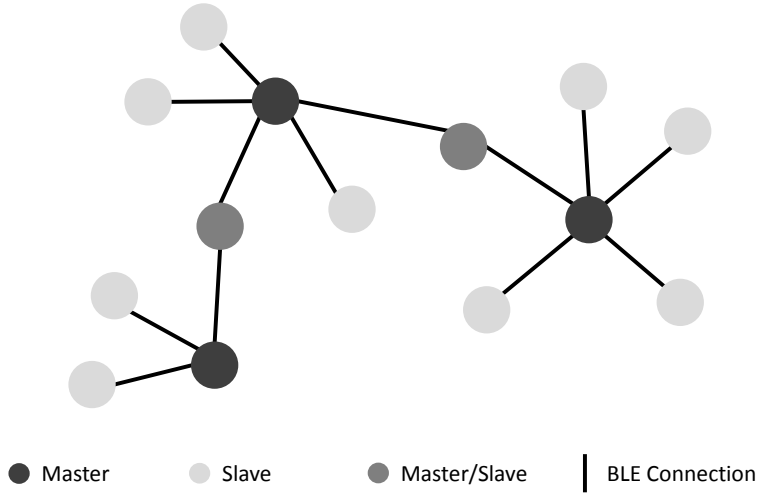


Figure 2.11: *An illustration of a Bluetooth scatternet topology.*

ence point of view, however, this also hinders message broadcasting/groupcasting which is required for algorithms like message flooding. A device can, without breaking any restriction, participate in a scatternet while scanning the advertising channels for broadcasts; the question is then whether or not the scatternet is even required.

Secondly, transmitting and receiving are expensive operations for a low energy device. To uphold a connection in BLE requires some data exchange, and when it comes to scatternets this maintenance cost is multiplied by the number of neighbors.

Finally, a scatternet require that some devices act as LL master and slave simultaneously. These devices must include multiple LL state machines, which increases their complexity and, moreover, their power consumption.

2.2.2 Multi-hop Communication

The BLE architecture is specifically designed for direct communication between a low energy peripheral and a central. Consequently there is no support for multi-hop communication (Figure 2.12) which is essential for mesh networks. Any packet that is sent across a mesh network is required to use some routing algorithm in order reach its destination; such algorithms are not defined in the BLE specification.

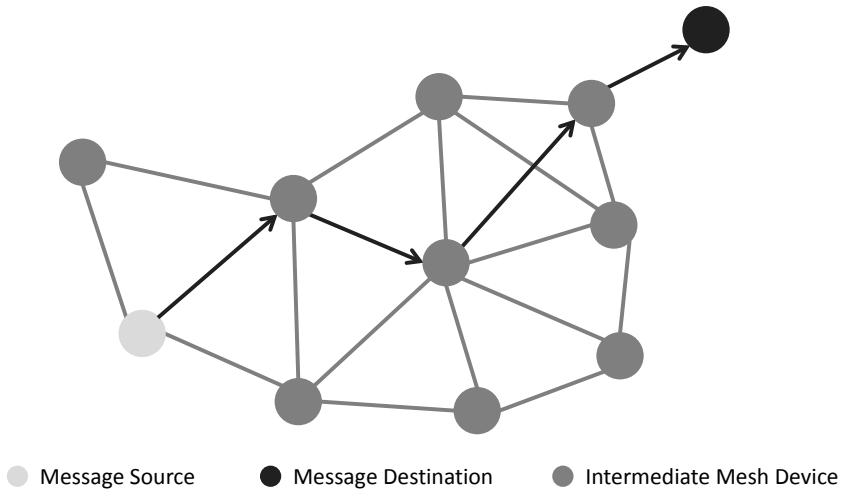


Figure 2.12: *An illustration of multi-hop communication in a mesh network.*

2.2.3 Timing Uncertainty

When communicating between the host and the HCI there is no specification regarding latency between sending a command and receiving an event. Furthermore, there is nowhere declared what the expected latency is before a command is executed. The consequence of this uncertainty leads to difficulties when trying to synchronize multiple BLE devices from data obtained via the HCI.

2.2.4 Advertising Control

The BLE specification clearly restricts the host from using an advertising interval that is shorter than 100 ms for nonconnectable advertising and scannable advertising. However, nothing is stated about how often the advertising may be disabled, modified and then enabled again [BLE, 2014b, p. 968]. This leaves room for manufacture dependent interpretations regarding how fast advertising data can be relayed via a BLE device within a mesh network.

3

A Non-Synchronized Mesh Protocol

This chapter introduces a model for one of the existing mesh protocols for BLE. This model will later be used as a baseline for the evaluation of the new mesh protocol that is proposed in this report. All references to the authentic protocol has been excluded in this report to protect proprietor of the product and to guard from any misinterpretations of technical details. The protocol will be referred to as the *non-synchronized mesh protocol* (NSMP). The network in which the protocol operates is called a *non-synchronized mesh network*, whereas a device that supports the protocol will be called a *non-synchronized mesh node*.

3.1 Overview

A non-synchronized mesh node can act in various modes depending on the scenario in which it is used. There are four particular modes in NSMP that cover the majority of use cases: controller mode, bridge mode, relay mode and listener mode.

A node that operates in the controller mode provides an interface for controlling other devices within range of the non-synchronized mesh network. Switches, sensors and id tags are typical examples of nodes that act as controllers. A controller can typically run on a coin cell battery for a long time.

The bridge mode is used to act as an interface between the non-synchronized mesh network and any other device that supports BLE. The bridge can be a BLE device that contains a NSMP bridge application, or, more commonly, a physical interface (e.g. a switch) that operates both as a controller and a bridge. Nodes that operate in the controller mode or bridge mode do not necessarily participate in the propagation of messages over the mesh. They rather act as input for any

action that can affect the state of other devices that are connected to the non-synchronized mesh network.

When a node operates in the relay mode it will retransmit any received data that is not exclusively addressed to the node itself. A relay may therefore have to receive, process and retransmit a lot of data all the time. These procedures are typically very power consuming so the relay is therefore usually connected to the power grid, or is attached to a high capacity battery, like a cell phone battery.

A node that operates as listener only receives messages and acts by its content. It is thereby very common to illustrate lights, alarms, industrial machines and home equipment as possible implementers of the listener mode.

NSMP allows a node to act in several modes simultaneously. The two most representative combinations are however to let a message source operate in controller and/or bridge mode, whereas the destination node often combines the relay mode and the listener mode.

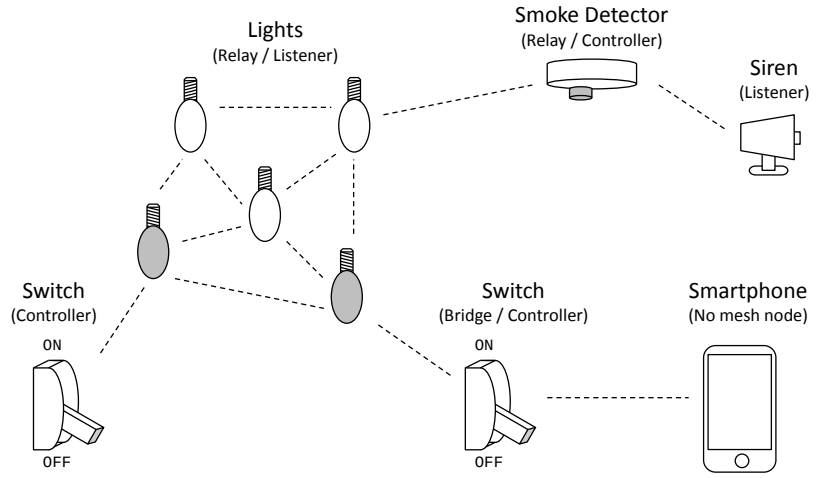


Figure 3.1: *Exemplary scenario for the non-synchronized mesh protocol (NSMP).*

3.2 Association

In order to maintain the non-synchronized mesh network the NSMP requires that transmitting devices are associated to each other. When relay mode is enabled in a non-synchronized mesh node, then the device starts to periodically transmit association messages every few seconds. Controllers and bridges can use these

transmissions to find nearby relays and, consequently, find proximate mesh networks.

Similarly, if a node is set to bridge mode, then the device starts to transmit bridge association messages once per second. This allows other BLE devices to find a bridge to a non-synchronized mesh network so that listeners can be controlled from for instance a smartphone or a computer.

3.3 Communication

When a non-synchronized mesh node enters the relay mode or listener mode, then it will set its scanning window length to

$$T_d = D_{scan} T_i,$$

where D_{scan} is the duty cycle and T_i is the scanning interval. D_{scan} can be configured within each device to tune the performance for either power saving or high packet delivery ratio. T_i is unspecified by the proprietor of the protocol so, for now, it is assumed to be 45 ms. It will later be shown that the selection of T_i does not have a major impact on the performance as long as it is less than $T_{i,max} = 100$ ms. After T_i is set then the relay will eventually start to scan the advertising channels.

All communication from a controller (or bridge) to a relay is performed by using non-connectable advertising. The controller sends its message three times on each advertising channel, with some random delays T_1, T_2 and T_3 in-between. The simulation model for this thesis will assume that the random delays between the retransmissions are calculated as

$$T_n = T_{n,min} + (T_{n,max} - T_{n,min})X,$$

where T_n is the random delay between the (n-1):th and n:th retransmission, $T_{n,min}$ and $T_{n,max}$ are the lower and upper bound of the delay respectively, and X is a uniformly distributed random variable between 0 and 1. Observations from BLE channel sniffing have shown that the boundaries in Table 3.1 give a fair resemblance of the actual protocol. The real model for transmitting messages is unfortunately protected by the protocol proprietor.

Communication among relay nodes is performed by using message flooding. A relay that receives a message from another node will eventually retransmit the message to all other neighbor nodes. Messages will be retransmitted three times in a similar way as the controller. Relay devices will, moreover, dampen the flooding effect by using a sequence number (SEQ NO) and a time-to-live (TTL) counter from the message. If the TTL is zero or the SEQ NO has already been received from a particular source, then the message is not retransmitted. The

Table 3.1: *The timing parameters for a NSMP controller and NSMP bridge.*

Parameter	Value
Scanning interval, T_i	45 ms
Minimal delay, first transmission, $T_{1,min}$	0 ms
Maximal delay, first transmission, $T_{1,max}$	0 ms
Minimal delay, second transmission, $T_{2,min}$	95 ms
Maximal delay, second transmission, $T_{2,max}$	110 ms
Minimal delay, third transmission, $T_{3,min}$	0 ms
Maximal delay, third transmission, $T_{3,max}$	20 ms

communication is performed by sending non-connectable advertising over all advertising channels. The boundaries of T_1 have been set to $T_{1,min} = 15$ ms and $T_{1,max} = 35$ ms for the relays in the simulation model.

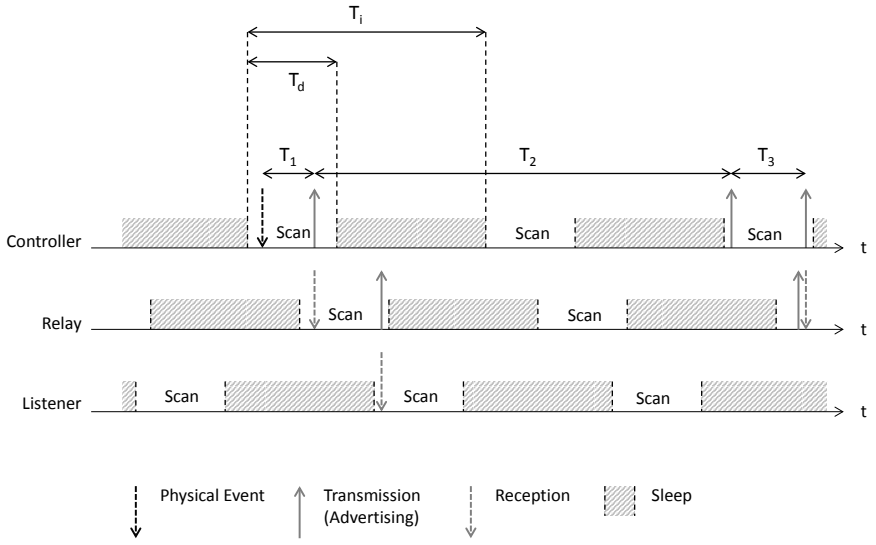


Figure 3.2: *A timing diagram of a message relaying in NSMP.*

3.4 Design Concerns

As shown in Table 3.1 the delay between the second and third retransmission will always be less than 20 ms. This means that the stated model actually violates one of the constraints in the BLE specification; namely that the interval for non-connectable advertising may not be less than 100 ms [BLE, 2014b, pp. 968]. Observations of the authentic protocol however confirm that this behavior is oc-

casionally triggered in both the controllers and the relays. The consequence of this violation is further commented in the results.

4

A Synchronized Mesh Protocol

This chapter thoroughly describes the proposed BLE mesh protocol that has been designed during this thesis. The proposed protocol uses various synchronization techniques to reduce the power consumption, and will therefore be referred to as the *synchronized mesh protocol* (SMP) (not to be confused with Security Manager Protocol, Section 2.1.8).

4.1 Overview

The SMP is targeted to support hundreds of connected devices while still retaining a low power consumption and low packet latency. To accomplish this, it has been proposed to use a special advertising message, called time beacon, in order to align the scanning windows of multiple devices. This implies that any device that has joined the mesh network only needs to transmit data during the time when other devices have their receivers turned on. This technique can probably save a lot of power compared to asynchronous communication. It has further been suggested that all communication should be exchanged via the advertising channels, since this enables quicker broadcasting and less device complexity. Moreover, the SMP is intended to support multi-hop communication. The proposed way to enable this is to let the application choose between two multi-hop algorithms: message flooding and message routing. Multi-hop messages should further be able to access attributes using the GATT in a similar way as a traditional BLE connection.

The suggested way to support all the requested features is to add a new layer to the BLE architecture that can coexist with the current BLE architecture. This layer is, for this thesis, assumed to be put between the controller and the host, so that it

can communicate directly with the HCI and exchange L2CAP packets with a BLE host. The added layer is referred to as the Mesh Layer (Figure 4.1).

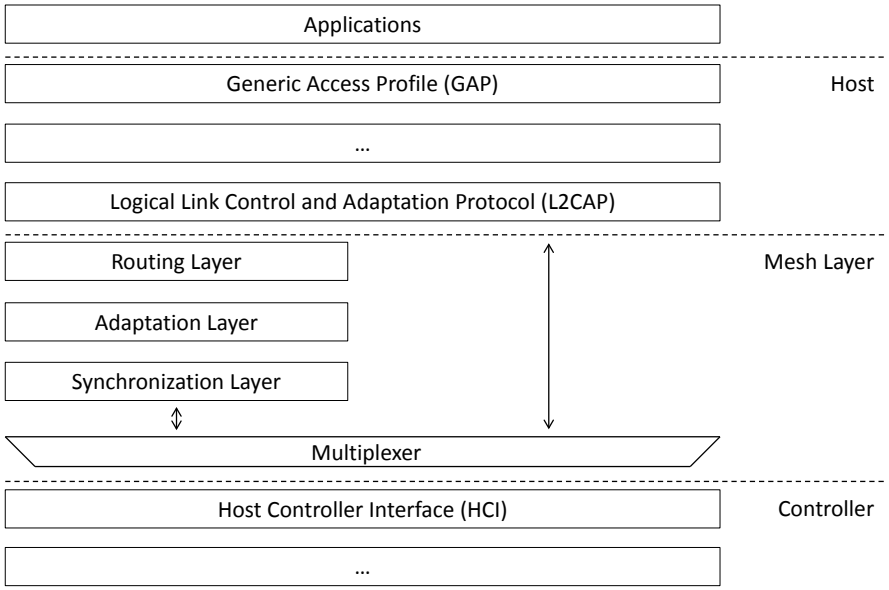


Figure 4.1: *The add-on module for the synchronized mesh protocol (SMP) within the BLE stack architecture.*

The mesh layer itself is composed of three internal layers: a synchronization layer, an adaptation layer and a routing layer. The synchronization layer and parts of the routing layer are mandatory, whereas the adaptation layer can be ignored if the manufacturer intends to create a less complicated system. These internal layers are carefully described in the following sections.

4.2 Definitions

This subsection defines some notions that are closely related to the SMP.

4.2.1 Transmission Window

A transmission window (TXWIN) is a finite time period for which the transmitter of a device may be active. The duration of a transmission window is denoted by T_{TX}, and is composed of multiple time slots of 0.625 ms. A device that runs the SMP may at most transmit data once per transmission window.

4.2.2 Reception Window

A reception window (RXWIN) is a finite time period for which the receiver of a device may be active. The device can receive data within a reception window only

if both the following conditions are met:

The transmitter of the device is not active.

The device is not switching between reception and transmission.

The duration of a reception window is denoted by $T_{RX,Di}$, where Di is the neighbor identifier for which the reception window is intended. $T_{RX,D0}$ is thereby the reception window for device D0. Each device that runs the SMP and wants to receive data must further have its own local reception window, whose duration is denoted as T_{RX}.

4.2.3 Global Reception Window

The global reception window (GRXWIN) of a mesh node is defined as the union of all its reception windows. The global reception window is usually identical to the scanning window of the mesh node[1]. The duration of the global reception window is denoted by T_{GRX}.

4.2.4 Mesh Node

A mesh node is a device that is a part of a mesh network. A device is considered to be a part of a mesh network if it has received a so-called time beacon (TB) and used the contained information to initialize a transmission window (Section 4.2.1) and/or a reception window (Section 4.2.2). For the synchronized mesh protocol, each node is assumed to have a distinctive address of two octets. The addresses 0x1000 to 0xffff are, for now, reserved for special purposes and should not be used. Address distribution will however not be discussed in this document.

4.2.5 Mesh Packet

A device that wants to send data to a mesh node via the mesh network is forced use an AD structure (Section 2.1.5) that is specially designed for the SMP. The AD structure consists of a single octet header, and some payload. The header begins with a Mesh Layer Identifier (MLID) of 2 bits. A `MLID` equal to 11b represents a mesh control packet (Figure 4.2). This implies that the last 3 bits in the header contains an operation code (OPCODE). Remaining bits in the header are reserved for future use. If, on the other hand, the `MLID` is equal to 10b or 01b then the packet is a L2CAP start or continuation packet, respectively, which implies that the last 2 bits of the header describes a packet transport (TRSP). A mesh packet with `MLID` set to 10b or 01b is called a mesh data packet (Figure 4.3). A device that wants to send data to a mesh node via the mesh network is forced use an AD structure (Section 2.1.5) that is specially designed for the SMP. The AD structure consists of a single octet header, and some payload. The header begins with a

[1] This may not be the case if the device acts in multiple modes/roles at the same time, or if the global reception window is disjointed.

Mesh Layer Identifier (MLID) of 2 bits. An `MLID` equal to 11b represents a mesh control packet (Figure 4.2). This implies that the last 3 bits in the header contains an operation code (OPCODE). If, on the other hand, `MLID` is equal to 10b or 01b then the packet is a L2CAP start or continuation packet, respectively, which implies that the last 2 bits of the header describes a packet transport (TRSP). A mesh packet with `MLID` set to 10b or 01b is called a mesh data packet (Figure 4.3).

The payload of a mesh data packet is further divided into two parts: routing information and L2CAP payload. The routing information contains data of how a packet should be transported within the mesh network. The L2CAP payload follows the normal format of a L2CAP packet (Section 2.1.6).

If the `TRSP` is 11b (flooding) then the routing information is 6 octets. The first octet is a packet identifier that should be monotonically incremented for each packet sent from a source to the same destination. The next two octets indicate the source address of the packet, i.e. the address of the originator mesh node. Subsequently there is a two octet destination address field, and a single octet time to live (TTL) counter.

If the `TRSP` is 00b (direct) then the routing information will contain the same fields as for flooding, but without the TTL counter.

If the `TRSP` is 01b (indirect) then the TTL field is also excluded, and instead the routing information ends with a two octet address field that indicates the next hop of a route.

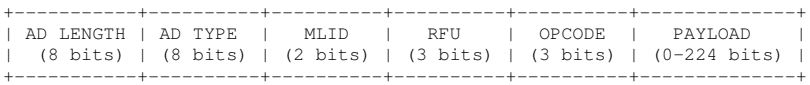


Figure 4.2: *The structure of a SMP control packet.*

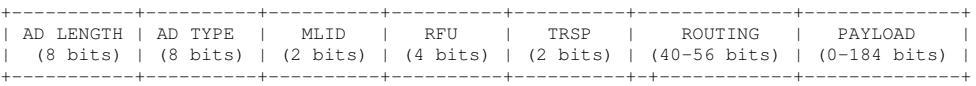


Figure 4.3: *The structure of a SMP data packet.*

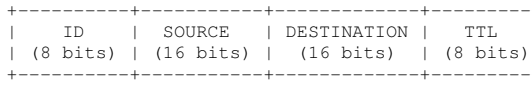


Figure 4.4: *The SMP flooding packet transport format (`TRSP` = 11b).*

4.2.6 Cycle Time

The cycle time is defined as the time between the start of two transmission windows in a device. The cycle time is denoted as T_{cycle}.

Figure 4.5: *The SMP direct packet transport format (`TRSP` = 00b).*

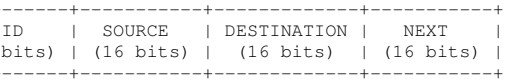


Figure 4.6: *The SMP indirect packet transport format (`TRSP` = 01b).*

4.2.7 Cycle

The period between the start of two transmission windows is called a cycle. Each cycle shall be associated with a unique number which is obtained by reading a counter at the start of the cycle. The counter shall be incremented by one for each reading. The number that is associated with the cycle is called cycle number and is denoted by $Z(C)$, where C is the cycle.

4.2.8 Time Beacon

A time beacon (TB) is a mesh control packet used to synchronize neighbor mesh nodes. The advertising data of a TB is arranged according to the format in Figure 4.7.

The `TB_HEADER` (Figure 4.8) contains three fields: `ACK_REQ` (1 bit), `SCA` (3 bits) and `TB_EXP` (4 bits). If `ACK_REQ` is set then the receivers should respond with a so-called *time beacon acknowledgment* within (`TB_EXP` + 1) cycles, if possible. `TB_EXP` indicates the expected time beacon reciprocal rate, λ_{TB}, minus one. That is the expected number of cycles between two TBs from the same mesh node. `TB_EXP` should be in range 0 to 15, where `TB_EXP` = 0 indicates that the next TB is expected in the next cycle.

The `SCA` field is used to indicate the sleep clock accuracy of the transmitting device. The value of this field must be set as defined in Table 4.1.

The source field of the TB indicates the address of the transmitting device. This field can be used to check whether the transmitting mesh node is still proximate to the receiving mesh node.

The `ID` field is a pseudo-unique identifier for the TB. The `ID` shall be a random 16-bit value. `CYCLE_TIME`, `TX_WIN` and `DT` are all two octet fields that specifies time in units of 0.625 ms. `CYCLE_TIME` indicates the mesh cycle time (T_{cycle}), `TX_WIN` indicates the duration of the transmission window (T_{TX}), and `DT` indicates the time between the start of the transmission window and the transmission of this TB (T_{δ}).

The `SYSTEM_TIME` field is used to synchronize the universal time in the mesh network, by using the POSIX time format [IBM, 2012]. This value can be used for encryption and validation purposes but is not further discusses in this document.

The transmission time, τ_{TB}, for a TB is 272 µs.

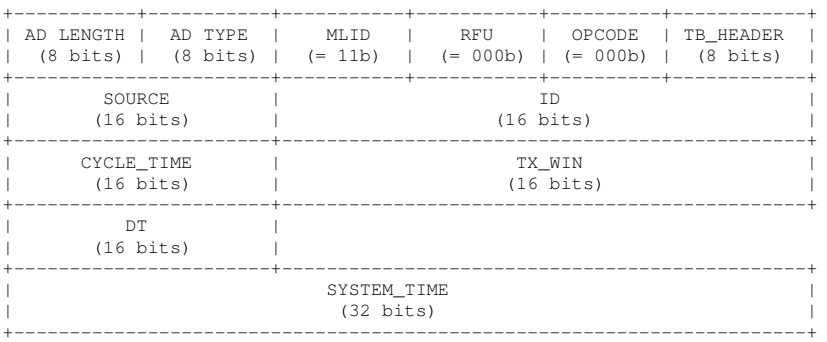


Figure 4.7: *The packet format of a SMP time beacon (TB).*

Figure 4.8: *The SMP time beacon header (TB Header) format.*

Table 4.1: *A table of SCA field encodings.*

SCA	Sleep clock drifting, ρ
0	251-500 ppm
1	151-250 ppm
2	101-150 ppm
3	76-100 ppm
4	51-75 ppm
5	31-50 ppm
6	21-30 ppm
7	0-20 ppm

4.2.9 Time Beacon Acknowledgment

A time beacon acknowledgment (TBA) is a mesh control packet used to notify the neighborhood of a mesh node that a certain set of time beacons has been received. The advertising data of a TB is arranged according to the format in Figure 4.9.

The TBA is essentially a list of time beacon identifiers. Each identifier in the list is a two octet field. The source field should indicate the address of the transmitting mesh node.

The transmission time, τ_{TBA}, for a TBA is 168 to 376 µs, depending on the number of items in the list.

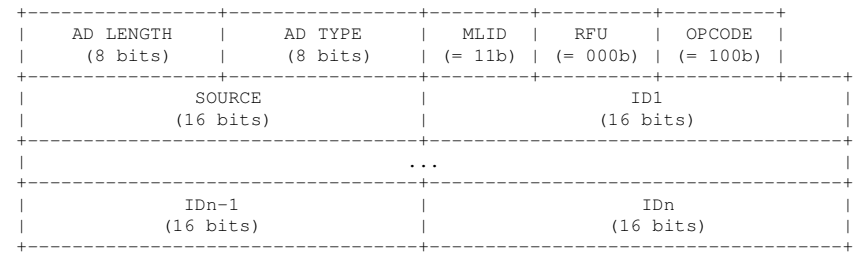


Figure 4.9: *The packet format of a SMP time beacon acknowledgment (TBA).*

4.2.10 Acknowledgment Table

A device which utilizes the later proposed transmission power adaptation technique must include an acknowledgment table. The acknowledgment table consists of two columns: TB identifier and acknowledgment counter. Furthermore, the adapting device must also include a row pointer which indicates the next empty row in the table.

4.2.11 Acknowledgment Vector

An acknowledgment vector contains all values in the acknowledgment counter column for each row less than the acknowledgment table row pointer. This vector can be used for statistical purposes.

4.2.12 Link

A link is a unidirectional channel between two mesh nodes. A link is created when a mesh node or a joining device receives a time beacon (Section 4.2.8) and uses the contained information to start a reception window (Section 4.2.2). The link is said to go from the transmission window of one mesh node to the reception window of another mesh node. For bidirectional communication, two links are required.

4.2.13 Skewness

Skewness is defined as the offset in time between the centers of two time windows. Link skewness is the skewness between a transmission window and a reception window which are linked. In this report, skewness is denoted by S.

4.2.14 Neighbor

A mesh node is considered to be a neighbor of another mesh node if there is a link from the first to the second node. To be a neighbor is by this definition a unidirectional property. Two nodes are only considered as "neighbors" if there is a link in both directions.

4.2.15 Route

A route is a transmission path between an originator mesh node and a destination mesh node. A route is always requested on-demand by the originator and acknowledged by the destination.

4.2.16 Routing Table

Each mesh node that supports packet routing must contain a routing table. The routing table is used to keep track of the next hop in a route towards some specific destination. Each entry in the table must at least contain the fields described in Figure 4.10.

Figure 4.10: *A SMP routing table entry.*

The DESTINATION field is the two octet address of the route destination. The value of this field should be unique within the routing table of each mesh node.

The destination sequence number (Destination_SEQ_NO) indicates the freshness of the table entry. A higher sequence number represents a fresher entry.

The FLAGS field contains additional route information. For now this field only contains two flags: invalid (I) and unknown sequence number (U).

HOPS indicate the number of hops required to reach the destination node. Zero indicates that the next hop is the destination itself.

The NEXT field contains the two octet address of the subsequent hop towards the destination.

Finally, the LIFETIME field indicates for how many milliseconds the route should remain active. The value of this field is derived from the RREP_DELAY field, in a so-called route discovery reply (RREP).

4.2.17 Routing Discovery Request

A route discovery request (RREQ) is a mesh control packet used to create a route between a source and a destination. The AD structure of a RREQ has the following format:

The packet starts with a single octet identifier that must be monotonically incremented by each mesh node for every transmitted RREQ. The identifier should be used by intermediate mesh nodes during a route discovery procedure to keep track of already processed RREQs.

The next two fields are the SOURCE and DESTINATION addresses of the RREQ. The values of these fields must be equal to the two octet addresses associated with the corresponding mesh nodes.

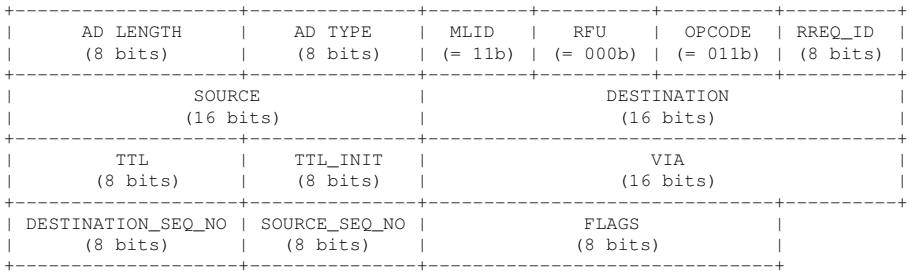


Figure 4.11: *The packet format of a SMP routing discovery request (RREQ).*

Next there are two single octet fields: time-to-live (TTL) and initial time-to-live (TTL_INIT). The TTL counter should be decremented by one for each hop during route discovery, whereas the TTL_INIT should stay unmodified. The number of hops between a source and a destination can be calculated by subtracting TTL from TTL_INIT.

The subsequent field, VIA, contains the address of the transmitting node. This field is updated in every intermediate mesh node.

The following fields are the source sequence number (SOURCE_SEQ_NO) and a destination sequence number (DESTINATION_SEQ_NO). These fields indicate the freshness of a routing request. The SOURCE_SEQ_NO should be incremented by one for each routing discovery request or reply. The DESTINATION_SEQ_NO should be set to the last known sequence number received from the destination mesh, or zero if none is known.

The final field, FLAGS, contains supplementary information that indicate how the RREQ should be handled. For now, it only contains a single flag U that should be set if the sequence number of the destination is unknown.

The transmission time, τ_{RREQ}, for a RREQ is 248 µs.

4.2.18 Route Discovery Reply

A route discovery reply (RREP) is a mesh control packet used to acknowledge a route discovery request. The AD structure of a RREP has the format described in Figure 4.12.

The RREP begins with a single octet identifier that shall be monotonically incremented by each mesh node for every transmitted RREP. The identifier should be used by intermediate mesh nodes during a route discovery procedure to keep track of already processed RREPs.

The following two fields, SOURCE and DESTINATION, are the two octet addresses of the RREP originator and the RREQ originator, respectively.

The NEXT field is the two octet address of the next mesh node in the route from the RREP destination back to the RREQ originator. The value of this field should

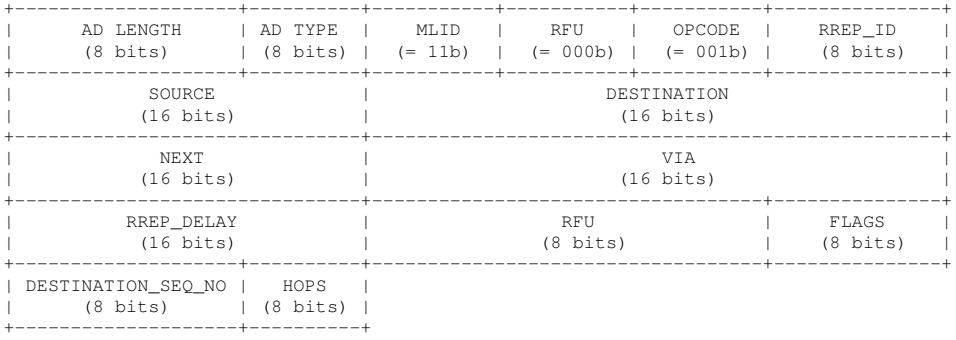


Figure 4.12: *The packet format of a SMP routing discovery reply (RREP).*

be retrieved from the local routing table of the transmitting mesh node.

VIA indicates the address of the transmitting mesh node. This field should be updated by all intermediate mesh nodes before retransmitting a RREP.

The RREP_Delay field is used to indicate the time in milliseconds from when a RREQ was received by the destination node, until the RREP was transmitted. This field shall not be modified by intermediate nodes.

The FLAGS field contains additional route reply information. This whole field is, however, reserved for future use and should for now be set to zero.

The destination sequence number (Destination_SEQ_NO) indicates the freshness of the acknowledged route. The Destination_SEQ_NO should be incremented by one for each for each transmitted RREQ or RREP.

Finally, the HOPS field indicate the number of intermediate nodes between the source and destination mesh node. This number should be generated by the RREP originator and stay unmodified by the intermediate mesh nodes.

The transmission time, τ_{RREP}, for a RREQ is 272 µs.

4.2.19 Route Ping

A route ping (PING) is a mesh control packet used to check the status of an active route. The AD structure of a PING is arranged according to the format in Figure 4.13.

The PING begins with a single octet identifier (PING_ID) that should initially be set to zero. The PING_ID should then be incremented by one for each time the PING is returned to its originator. The PING_ID makes it is possible for intermediate mesh nodes to keep track of already forwarded PING packets.

The Source and Destination fields are the two octet addresses of the source and destination node, respectively.

The following field, NEXT, is the two octet address of the next mesh node on the

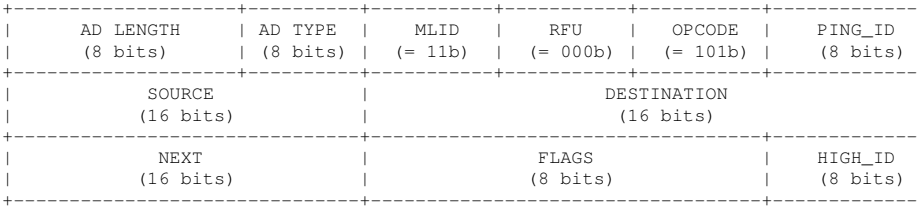


Figure 4.13: *The packet format of a SMP ping (PING).*

route for this PING message. The value of this field should be retrieved from the local routing table of the transmitting mesh node.

For now, the `FLAGS` field only contains a single flag, *From Originator* (O), that should be set if the ping message was sent from the route originator towards the route destination. When the PING reaches its destination, then the *From Originator* flag should be cleared.

At the end of the PING packet, there is a single octet field, `HIGH_ID`, that indicates the highest cohesive data packet ID that has been received from the other end node. The other end node can use this field to detect packet losses along the route.

The transmission time, τ_{PING}, for a PING is 224 µs.

4.3 Synchronization

BLE devices consume a lot of power by just having their receivers turned on. One feature that can reduce this cost is to introduce time synchronization so that devices only collect data when there is a fair chance that some other device is transmitting. One way of achieving this is to introduce time beacon synchronization, i.e. to use advertising packets to synchronize the scanning window of multiple devices. All synchronization procedures are handled by the synchronization layer of the mesh node.

4.3.1 Assumptions

To perform the proposed mesh synchronization, some essential features are required by the BLE controller. The necessary features stated below are neither required nor forbidden by the BLE specification; hence the support for these is up to the manufacturer.

1. The time from fully receiving a packet in the controller until an event is reported by the HCI must be deterministic. This time is denoted as T_e.
2. The time from when the host sends a scan enable command to the HCI until the first scanning window is started must be deterministic. This time is denoted as T_s.

3. The time from when the host sends an advertising enable command to the HCI until the first advertising packet is transmitted must be deterministic. This time is denoted as T_a.

4.3.2 Initialization

The first device in the mesh network begins the mesh initialization procedure by preparing its transmission window. This is performed by invoking some specific HCI commands (see Chapter A). First the device must invoke `LE Set Advertising Parameters`, so that assumption 3 is covered (Section 4.3.1). The proposed way of doing this is to let the controller accept the HCI command `LE Set Advertising Parameters` with `ADVERTISING_INTERVAL_MIN` = 100 ms, `ADVERTISING_INTERVAL_MAX` = 100 ms and `ADVERTISING_TYPE` = `ADV_NONCONN_IND`. By doing this, the host can send advertising packets with precise timing by enabling advertising, wait for some time, and then disabling advertising again. The advertising state can be changed by invoking the HCI command `LE Set Advertising Enable`. The host, however, has to bear in mind that there is a delay T_a before the HCI command takes effect. The mesh initiator must ensure that $T_{cycle} - T_{TX} > 100$ ms to avoid any violation of the constrain mentioned in Section 2.2.4.

The next step is for the mesh initiator to decide the cycle time T_{cycle} and transmission window duration T_{TX} for the mesh network. The reception window should then be initialized by invoking the HCI command `LE Set Scan Parameters` with `LE_SCAN_INTERVAL` = T_{cycle} and `LE_SCAN_WINDOW` = T_{TX}. With the scanning configured, the host can then control the state of reception window by invoking `LE Set Scan Enable`. Correspondingly, the host must recall that there is a delay T_s before the HCI command takes effect. To begin the mesh synchronization, the host starts by opening its reception window. For the time being, the reception window, the transmission window and the global reception window are indistinguishable. In other words, within this time window the node is either transmitting or receiving data.

The next step for the host is to select one of its 0.625 ms time slots within its transmission window (Section 4.2.1); the offset of this time slot is denoted as T_δ. In addition, the host must create a TB with `CYCLE_TIME` = T_{cycle}, `TX_WIN` = T_{TX} and `DT` set to the offset of the time slot, T_δ. The host may, for now, set the fields `ACK_REQ` = 0 and `TB_EXP` = 0. This TB should then be transmitted at the start of the selected time slot. When the TB has been transmitted, the host waits until the reception window ends whereon it sends the controller to sleep. After $T_{cycle} - T_{TX} - T_s$ seconds, the host should awaken the controller, and then select a new time slot for transmission. This procedure is repeated for every cycle, i.e. once every T_{cycle} seconds (Figure 4.14).

4.3.3 Joining

A device that wants to join a mesh network starts by preparing its transmission window (Section 4.3.2) followed by activating its scanning state. If there is a

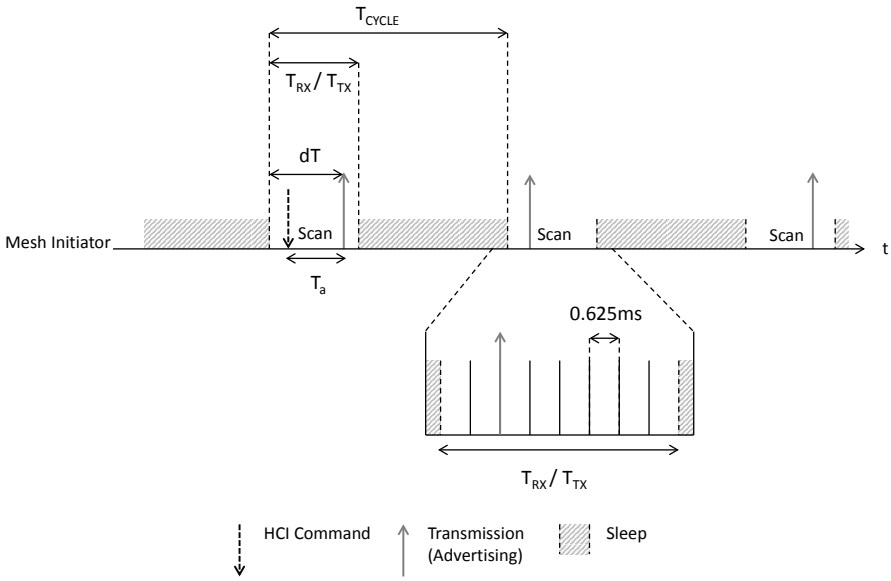


Figure 4.14: *Timing diagram for mesh initialization in SMP.*

mesh node nearby, then the joining device will eventually receive a TB. When a TB is received, the joining device can use the information contained in the TB to initiate a reception window and transmission window that are aligned with the transmission window of the mesh node.

If clock drifting is ignored, then the joining device should configure its reception window with the `LE Set Scan Parameters` command using `LE_SCAN_INTERVAL` = T_{cycle} and `LE_SCAN_WINDOW` = T_{TX}. To align the reception window correctly, the `LE Set Scan Enable` command should be invoked $T_{cycle} - (T_e + \tau_{TB} + T_\delta + T_s)$ seconds after the TB was reported from the HCI.

In addition, the joining device shall also open a transmission window. For now, the transmission window will coincide with the reception window, as clock drifting is ignored. The joining device must use its transmission window to send out its own TB. Similar to the mesh initiator, the joining device randomly selects one of the 0.625 ms time slots within its transmission window for the transmission of its TB. Just like for the mesh initiator, the joining device has to recall that there is a delay T_a between invoking the HCI command and the TB is sent (Figure 4.15).

4.3.4 Linking

A mesh node or a joining device can receive multiple TB from other mesh nodes. The receiver should then start multiple reception windows so that each window is aligned with the transmission window of the corresponding transmitter. In the ideal case, all windows in the network are perfectly aligned; however, this is not the case in the real world. Due to differences in each device there will be

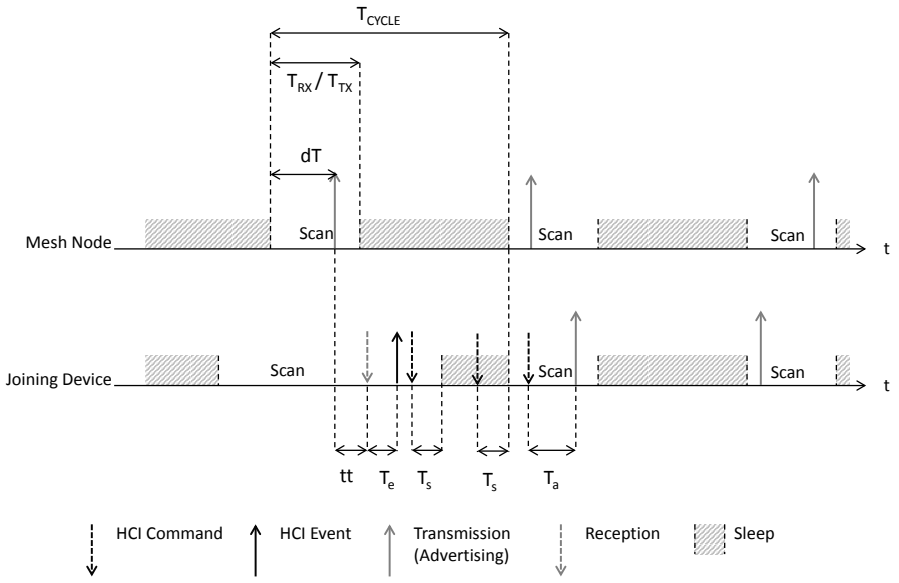


Figure 4.15: *Timing diagram for mesh joining in SMP.*

some skewness between the time windows, which leads to the conclusion that each reception window must be maintained separately.

For the proposed mesh protocol, the term link is consequently introduced. When a TB is received by a joining device, or a mesh node, and a reception window is started, then it is said that a unidirectional link has been established. Each link thereby connects a transmission window with a reception window. A reception window can at most be associated with one link, whereas a transmission window can be associated with many. If a TB is not received via a particular link within some time the link is considered to be broken, and its reception window can hence be discarded. It is recommended to wait three times (`TB_EXP` + 1) cycles before breaking a link. A mesh node that does not have any remaining links is considered to be disconnected.

Each mesh node should have one unlinked reception window that coincides with its transmission window. This is required to let new devices join the mesh network.

4.3.5 Clock Drifting

Within distributed systems, the concept of time is usually imprecise. Any device with its own real-time clock (RTC) will experience that the local time runs in different speed compared to other devices. This phenomenon is called clock drifting, and is often a vexatious problem in synchronized systems. The clock drifting of a device usually varies over time; partly because of temperature changes, but also because of irregularities in the supply voltage [Windl, 2006]. Consequently,

RTC manufacturer usually gives the upper bound of the clock drifting for some specific conditions. The clock drifting is typically specified in parts per million (ppm) [STM, 2013] [Max, 2015].

4.3.6 Window Widening

For the proposed mesh protocol, clock drifting is partly seen as a gradual increase of skewness between the transmission window and the reception window for a particular link. This effect must somehow be rectified. The proposed way of doing this is to widen the reception window for each cycle. This is very similar to the window widening which is used for normal BLE connections [Heydon, 2013, pp. 307-308].

The increase in skewness between a reception window and a transmission window for a particular link will for every cycle be bounded by

$$S_{max} = (\rho_{local} + \rho_{peer})T_{cycle},$$

where S is the skewness, ρ_{local} is the sleep clock accuracy (SCA) of the local mesh node, and ρ_{peer} is the sleep clock accuracy of the mesh node on the other side of the link. The gradual widening of a reception window must therefore be

$$W = 2 \cdot S_{max},$$

and consequently the total duration of a reception window is calculated as

$$T = W(\mathrm{Z}(C_{current}) - \mathrm{Z}(C_{lastTB})),$$

where $C_{current}$ is the ongoing cycle and C_{lastTB} is the cycle of which a time beacon was last received for the particular link. $\mathrm{Z}(C)$ denotes the cycle number of C, as mentioned in Section 4.2.7.

This way of widening the reception window will prevent the receiver from missing a transmission due to clock drifting. To widen the reception window, the host has to increase the `LE_SCAN_WINDOW` parameter by W when calling `LE Set Scan Parameters`. The host also has to invoke the HCI command `Le Set Scan Enable` $W/2$ seconds earlier than planned.

A time window of a particular link can be shrunk if newer information regarding that link is received via a TB.

4.3.7 Window Translation

The previous subsection showed that window widening can be used to solve the problem of skewness within a link. However, clock drifting does not only cause

skewness to appear within links, but also between them. Two transmission windows, engendered by two different devices, will eventually drift apart enough to affect the duration of the global reception window for a third device. This severely impacts the power consumption for the third device, and must somehow be handled by the protocol.

One way of doing this is to introduce window translation. If a device detects that its own transmission window impairs the power consumption of another device, then it is allowed to translate its transmission window to a better position in time. The proposed metric for finding a better position in time is to minimize the skewness, S, between the global reception window and the transmission window.

Window translation is performed by adding or subtracting a small value from T_δ within a TB, and compensate this action by translating the transmission window with the same value (Figure 4.16).

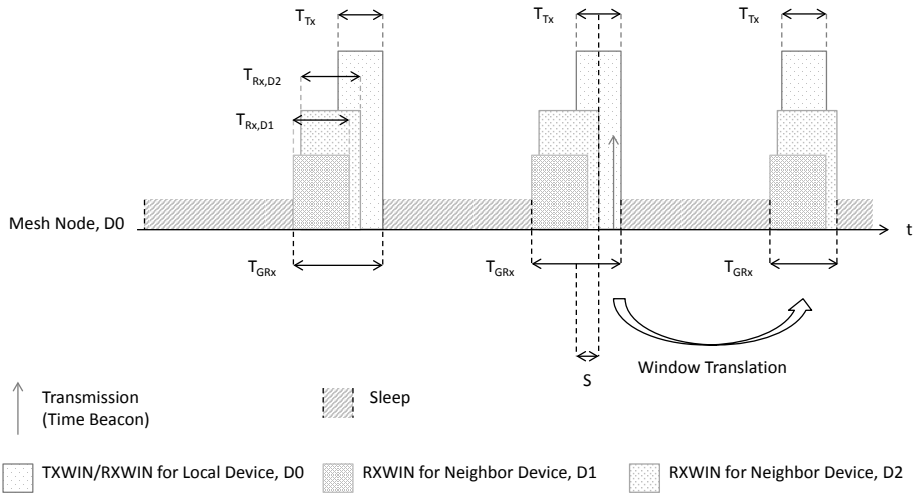


Figure 4.16: *Timing diagram for window translation in SMP.*

4.3.8 Custom Advertising Data

The `TB_EXP` field in a TB indicates the expected number of cycles between TBs from the transmitting mesh node. With `TB_EXP` = 0, a mesh node is expected to send out a time beacon once per cycle, which prevents any other data from being transmitted. However, `TB_EXP` > 0 allows mesh nodes to transmit custom advertising data at some random time within its transmission window. At most one advertising packet can be sent in each transmission window.

A common situation is that some mesh nodes receive bursts of packets within their reception window. It is, moreover, typical that some packets contain routing

information, and should likely be forwarded to other nearby mesh nodes. This calls for a packet queue, and some priority system, so that both application data and synchronization data is conveyed. In this document it is assumed that a mesh node does not have any limit regarding the number of packets that can be queued. The priority for transmitting packets is:

1. A time beacon should be transmitted at least once per $2 \cdot (\texttt{TB\_EXP} + 1)$ cycles.
2. If the transmission queue is empty, then a time beacon should transmitted every $(\texttt{TB\_EXP} + 1)$ cycle.
3. If the transmission queue is not empty, then one packet should be dequeued and transmitted every cycle.

If a mesh node wishes to retransmit a packet multiple times, then the packet should repeatedly be put into the transmission queue.

4.4 Adaptation

The previous section explained how devices can be synchronized by using time beacons, and how custom data can be transmitted within a synchronized mesh network. At this point it is possible for devices to exchange data and thereby perform various adaptation techniques, in order to increase their efficiency in some aspect. This chapter will present potential adaptations techniques. All adaptation techniques are optional. The adaptation techniques are handled by the internal adaptation layer of the mesh node.

4.4.1 Transmission Power

Usually a BLE device has a fixed transmission power that is specified by the chip manufacturer. A high transmission power implies a long propagation range, but also higher interference on the shared channel. When creating the device deployment for a particular scenario, it is often a problem that the operator does not know how the signal propagates in the environment, and thereby places the devices unjustifiably close to each other. Moreover, when some devices are portable, the distance between devices will change over time. It can therefore be suspected that using a fixed transmission power is likely to lead to reduced performance. This subsection will introduce a simple transmission power adaptation technique that is targeted to reduce the effect of this problem, and without causing high traffic load in the mesh network.

The general idea behind this adaptation technique is to estimate the probability of reaching a number of mesh nodes with a certain transmission power. The estimation is then compared to a reference value in order to determine if the transmission power should be increased or decreased.

When a mesh node receives a TB which requires acknowledgment ($ACK\_REQ = 1$), then the node is encouraged (not required) to store the TB identifier in its

memory. At some later time (preferably before the next expected TB via the same link), the mesh node should respond with a time beacon acknowledgment (TBA). The TBA must contain a list of unacknowledged TB identifiers that have been stored. A TBA can contain up to 13 TB identifiers. If the node has received more than 13 TB identifiers, then it must randomly select 13 from its memory.

In the event of receiving a TBA, the receiving mesh node should search the contained identifier list for a TB identifier that matches one in its acknowledgment table. If a match is detected, then the acknowledgment counter for this particular row shall be incremented by one. Eventually, as more TBAs are received, an acknowledgment vector is obtained.

There are numerous ways of extracting information from the acknowledgment vector in order to find a suitable transmission power. One way that has been verified to work in a dense networks with up to 150 devices, is to use P-regulation based on statistics.

Assume that an acknowledgment vector contains the elements $x_0, ..., x_{n-1}$. The mean value, $\overline{u}$, and the standard deviation, $\overline{d}$, can then be calculated as

$$\overline{u} = \frac{\sum_{i=0}^{n-1} x_i}{n},$$

$$\overline{d} = \sqrt{\frac{\sum_{i=0}^{n-1} (x_i - \overline{u})^2}{n-1}}.$$

The probability of reaching u neighbor mesh nodes with the current transmission power is then estimated as

$$\overline{p} = \mathrm{Q}\left(\frac{u - \overline{u}}{\overline{d}}\right),$$

where Q(x) is the Q-function. The error, e_p, for the P-regulator can then be calculated as

$$e_p = p - \overline{p},$$

where p is the desired probability of reaching u neighbor mesh nodes.The new transmission power should then be set to

$$P_{new} = P_{current} + ke_p,$$

where $P_{current}$ is the current transmission power and k is some positive constant.

This means that if the probability of reaching u neighbors is too small ($e_p > 0$), then the transmission power will be increased, whereas too high probability will decrease the transmission power.

One known problem with this estimation is when the desired probability is set to a high value, e.g. $p = 0.99$. The estimated probability, $\overline{p}$, will obviously be in the range [0,1] which means that if $\overline{p} > p$ then $e_p = p - \overline{p} > p - 1 = -0.01$. This leads to bad recovery from overshooting the desired probability. When it is necessary to use p close to 0 or 1, it is instead recommended to use some function $g(x)$ and calculate the error as

$$e_p = g(p) - g(\overline{p}).$$

Some properties of $g(x)$ that has been observed to give good recovery from overshooting, and that gives an error that stabilizes around zero are:

1. $\lim_{x \to 1} g(x) = \infty$
2. $\lim_{x \to 0} g(x) = -\infty$
3. $g(\frac{1}{2} + x) = -g(\frac{1}{2} - x)$
4. $g(x)$ is strictly monotonic

The function that has been used in the SMP simulations is

$$g(x) = \begin{cases} 1 + \log_2(x), & if\, x \leq \frac{1}{2} \\ -1 - \log_2(1 - x), & if\, x > \frac{1}{2} \end{cases}.$$

If the value of k is small enough then each acknowledgment vector will bring $\overline{p}$ closer to p as the transmission power is updated. The described adaptation procedure may run sporadically or periodically. If the procedure is run periodically, then the adaptation interval is denoted by T_{adapt}.

4.4.2 Retransmission Control

Typically the desired probability, p, cannot be reached by only increasing the transmission power. In most cases it is more applicable to instead retransmit a packet multiple times. A mesh node that uses transmission power adaptation is further encouraged to set its retransmission counter, N, to

$$N = \left\| \frac{\log(1 - p)}{\log(1 - \overline{p})} \right\|,$$

when sending any higher layer packets. TBs and TBAs should, however, not be retransmitted multiple times as this would only increase the traffic load without

gaining any performance boost. When the retransmission counter is greater than one then the probability error, e_p, should instead be calculated as

$$e_p = g(p) - g(1 - (1 - \overline{p})^n).$$

4.4.3 Acknowledgment Vector Filtering

If a time beacon collides with another transmission, and its ACK_REQ field is set, then it is likely that there are no acknowledgments at all for this particular beacon. This will lead to a reduced value for some samples in the acknowledgment vector, and thereby the mean value, $\overline{u}$, will decrease, and the standard deviation, $\overline{d}$, will increase. It is in this case desirable to separate the samples which are likely to be an effect of time beacon collisions from the values that were obtained if the number of neighbors was too low. One proposed way of doing this is to exclude all the zeros in the acknowledgment vector if

$$\overline{d} > \epsilon * \overline{u},$$

where ϵ is a constant, was obtained if the zeroes were included. The constant ϵ is here called the *collision detection coefficient*, and it has been set to $\epsilon = 1.5$ for the SMP.

4.4.4 Time Beacon Rate

When performing initial synchronization of a mesh network, it is usually desired to have a high TB transmission rate in order to quickly establish all links. However, when application data should be transmitted then a high TB transmission rate can instead deteriorate the performance of the network because of a higher packet collision probability. A mesh node is therefore encouraged to initially select a low TB reciprocal rate (TB_EXP + 1) and then gradually increase its value. It is recommended that TB_EXP is incremented at most once per three transmitted TBs; this is to reduce the risk of breaking a links.

4.5 Communication

It has been explained how BLE devices can initiate and join a synchronized mesh network (Section 4.3) and how two neighbor mesh nodes can use their transmission windows to exchange data (Section 4.3.8). The proposed mesh protocol, however, requires that packets can be sent between any two mesh nodes within the network, not only between neighbor nodes. This calls for some packet forwarding algorithm. The following subsections will propose two different ways to forward a packet within a mesh network: flooding and routing. All routing procedures are handled by the routing layer of the mesh node.

4.5.1 Message Flooding

The proposed mesh protocol is required to endure hundreds of connected mesh nodes, and let any mesh node join or leave whenever they need. This leads to the fair assumption that no mesh node can have a full picture of the network; at least if the mesh topology is frequently changed. In this case, when two distant mesh nodes need to communicate, a likely scenario is that some messages must be routed through an uncharted area. One common way to forward messages through such areas is to use message flooding. Flooding means that any incoming message is retransmitted to all neighbors except the neighbor from which the message arrived. This way of forwarding a message clearly generates a vast number of transmissions, and can therefore be unpractical in a dense network with many communicating nodes. However, flooding is also a very robust forwarding technique. Even if some nodes are unable to receive a message, or that some in-between nodes are disconnected from the network, the message will still have a chance to reach its destination via alternative paths. Flooding also guarantees that messages will be forwarded via the shortest possible path to its destination (since all paths are covered). This means that there is no shorter message latency than the latency got from using flooding [Tanenbaum, 2011, pp. 368–370].

A flooding packet (Figure 4.17) in SMP includes two dampening features to reduce the number of transmissions: a packet identifier (ID) and a time-to-live (TTL) counter. The packet `ID` is used to keep track of already transmitted packets, and thereby preventing packets from going around in a loop. A mesh node should use a monotonically increasing `ID` for each destination when sending flooding messages. The `TTL` counter is used to limit the range of the flooding. A mesh node that receives a flooding message should decrement the `TTL` counter before forwarding the message. Messages with zero `TTL` should not be forwarded.

Moreover, a flooding message shall include a `SOURCE` and `DESTINATION` address. The address fields shall be the two octet addresses associated with the source and destination mesh node (Section 4.2.4). To broadcast a message, the destination address 0xffff shall be used.

Any data attached to the message shall use the L2CAP packet format. If the L2CAP packet does not fit in one flooding message then MLID in the packet header can be set to 01b in the next message to continue a L2CAP transaction. The packet `ID` can be used to reconstruct the original L2CAP packet if the packet parts arrive in inconsistent order.

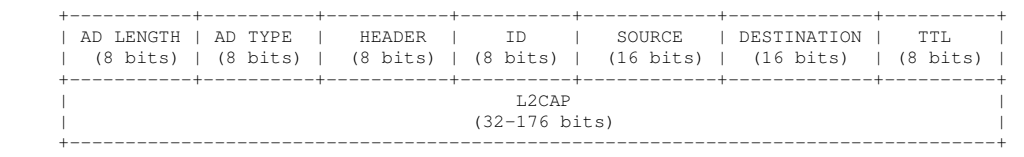


Figure 4.17: *The format of a SMP L2CAP flooding packet.*

When a flooding message is received, then the mesh node should check if the

packet ID from the provided source to the given destination has already been handled within some time; in this case the packet shall be ignored. Next, the mesh node should check if the message DESTINATION is equal to its own address, or if it is 0xffff. If so, then the packet should be sent to an upper layer in the device. If TTL is greater than zero and the destination address is not equal to the address of the mesh node, then a copy of the packet should be put into the transmission queue, with TTL decremented by one.

4.5.2 Routing Algorithms

When two non-neighbor mesh nodes frequently exchange data, then flooding may be very unhandy. If there exists a robust path between the two mesh nodes, then single path message routing typically outperforms flooding in the aspects of resource usage and data throughput. The reason for this is that the resource and bandwidth usage per packet is reduced, which usually affects the overall performance of the network [Tanenbaum, 2011, pp. 362–364]. Still, it is very unlikely that any node has a full picture of the network, whereupon message routing may appear unrealistic. There are, however, some routing algorithms that are well adapted for similar situations. One such algorithm that has been investigated is Ad hoc On-Demand Distance Vector (AODV) routing.

AODV is a routing algorithm that is designed for mobile ad hoc networks. The algorithm provides route discovery on-demand which is well suited for networks with frequently changed topologies. Any node may initiate a route discovery request by asking its neighbors for a route to a specific destination. The neighbors then recursively ask their neighbors for the same route, which eventually reaches the destination node. During the discovery process, each intermediate node stores the address of the originator and the requester in its routing table. The table can in this way be used to route a packet back from the destination to the originator. When a request reaches its destination, then the destination node responds by routing a packet back to the originator via the same path. The intermediate nodes store the address of the destination and the previous node in the same table, and thereby create a bidirectional route between the originator and the destination.

To maintain an active route, the algorithm requires that each intermediate node periodically reports its status. If a node has not been heard from within some time, then the link is considered as broken whereupon some action must be taken. A node can in this case choose to either perform a link repair, which implies sending a new discovery request for the destination node in the direction of the broken link, or otherwise report a route error in the opposite direction of the broken link.

AODV further includes a feature for discarding inactive routes. During route establishment, each intermediate node stores a timeout value together with its local time in a table. Every time a packet is sent via the route, the local time of the table entry is updated, which thereby refreshes the status of the route. Whenever a node detects that its local time subtracted by the time stored in an

entry is longer than the timeout, then the node may discard the route [Perkins, 2003].

4.5.3 Route Discovery

Even though AODV includes many convincing features for establishing and maintaining a route, it is still not well suited, in its pure form, for a BLE mesh network. It has been discovered in simulations that a route discovery request typically reaches its destination, whereas the response, conversely, rarely reaches back to the originator. Although AODV has a blacklisting mechanism for excluding poor links, it is not applicable to use this feature in the SMP. The main reason is because of the mechanism's poor performance in networks with frequent topology changes. Furthermore, AODV is intended for IP routing which is not intended for the SMP. Consequently, it has been proposed to test a modified version of AODV that strengthens the algorithm for the BLE mesh networks. The modified algorithm uses the same concept of on-demand route discovery, but includes tougher constrains on link establishment. For the modified algorithm, a link can only be established if a discovery request is received with signal strength higher than some minimal threshold, $P_{th,route}$. Moreover, the modified algorithm requires route maintenance by end-to-end packet exchange, in other words pinging. If a ping message is not received by the route originator within some time, then the path is considered as broken and must be reconstructed from the originator. Any intermediate node may discard a route that has not transported a ping message within some time.

Ping messages can further be used to acknowledge package receptions from the other end node. One way of doing this is to include a monotonically increasing sequence number in every package that is routed from one end to the other. The ping message should then include a field that indicates that highest cohesive received sequence number from the other node, and in this way let the packet transmitter know if a packet was lost somewhere along the path.

To establish a route in SMP the originator node sends out a RREQ (Section 4.2.17) to the destination using flooding. Any intermediate node that supports routing must update its routing table according to the AODV specification [Perkins, 2003] using the sequence numbers of the packet. Before retransmitting the RREQ, each intermediate device should update the `VIA` field in the RREQ to its own address so that the receiving nodes can build a path to the originator during route discovery. When the RREQ reaches the destination, then the destination should update its neighbor table similarly to the intermediate nodes. The destination node should then wait for some time, T_{RREP}, before sending back a RREP to the originator using its own routing table. The waiting time should be written into the RREP Delay field in the RREP (Section 4.2.18). If the originator node does not receive a RREP within the RREQ timeout, ξ_{RREQ}, then the route discovery is considered as unsuccessful.

The RREP is a routed message that uses the routing table of the intermediate devices between the destination and the originator. A node that receives a RREP

should compare its address to the NEXT field in order to determine whether the RREP was intended for the node. If there is a match, then the node is considered to be an intermediate node. Any intermediate node should use the contained information to update its neighbor table according to the AODV specification. The VIA field in the RREP should then be updated to the node's address before retransmitting it to the next node (whose address is retrieved from the routing table). The lifetime of all active routing entries for one particular route should be updated upon reception of a RREP. The lifetime should be set to a lifetime factor, ξ_{PING}, times the delay between the processed RREQ and RREP for the particular route. The delay is calculated as the time between transmitting the RREQ and receiving the corresponding RREP, minus the time in the RREP_DELAY field in the RREP.

4.5.4 Message Routing

If there is an active route between two mesh nodes, then message routing can be used to exchange data. The message routing runs on the information stored in each node's routing table. The format of a routing packet is shown in Figure 4.18.

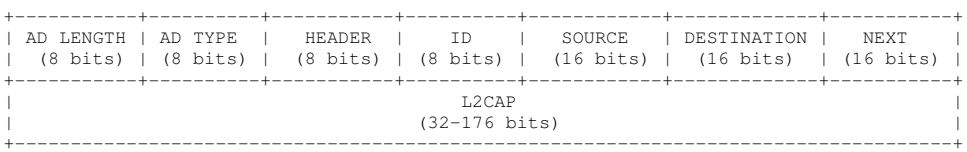


Figure 4.18: *The format of a SMP L2CAP routing packet*

The HEADER of the routing packet consists of a MLID and TRSP field (Section 4.2.5). If the L2CAP payload is the start of a new L2CAP packet then MLID should be set to 10b. If the L2CAP payload is a continuation of a fragmented L2CAP packet then MLID should instead be set to 01b. The transport of a routing message should always be set to indirect (TRSP = 01b). The packet identifier (ID) should monotonically incremented in each mesh node for every packet sent to the same destination. The packet ID can in this way be used to reconstruct the original L2CAP packet if the packet parts arrive in inconsistent order.

The SOURCE and DESTINATION field are the two octet addresses of the corresponding mesh nodes. The NEXT field is used to indicate the two octet address of the next mesh node on the route. The value of the NEXT field is obtained from the local routing table of each node.

Any additional data attached to the message shall be formatted as a L2CAP packet.

When a routed message is received, the mesh node should check if the packet ID from the provided source to the given destination has already been handled within some time; in this case the packet shall be ignored. The mesh node should in addition check if the NEXT address is unequal to its own address, whereas this should also result in ignoring the packet. Subsequently, the mesh node should check if the message DESTINATION is equal to its own address. In this case, the

packet should be sent to an upper layer in the device. If not, then the mesh node should look in its routing table to see if there is a known next hop towards the destination address. If an address is found then the packet should be forwarded to the next mesh node.

5
Simulation

The mesh protocols that have been presented in this thesis were simulated at Ericsson Research in Linköping. This chapter introduces the system model for the BLE mesh simulator, and the configuration of some test cases that were used to obtain the system performance results.

5.1 System Overview

The simulator model is composed of one BLE Transmission Manager and a number of BLE Mesh Devices. Each BLE Device is further divided into a layer structure, whose internal structure depends on the simulated protocol. NSMP includes a Controller Layer, NSMP Mesh Layer and a Packet Logger. SMP includes a Controller Layer, Synchronization Layer, Adaptation Layer, Routing Layer and Packet Logger (Figure 5.1).

The transmission manager is used to handle the physical transportation of packets between multiple BLE devices. When a packet is transmitted from a BLE device, then the manager creates a wrapper instance that translates the logical data unit into a physical transmission. The physical transmission moreover contains a duration, a signal-to-noise ratio (SNR), a signal-to-interference-plus-noise ratio (SINR), a transmission power (P_{TX}), and a received signal strength (P_{RX}). The probability of successfully receiving a packet is based upon the parameters of the transmission. A new transmission that is added to the transmission manager will affect the state of all other ongoing transmissions.

The multi-layer architectures of the BLE devices are used to simplify the construction of the mesh protocols and to verify the correctness of each part in the device.

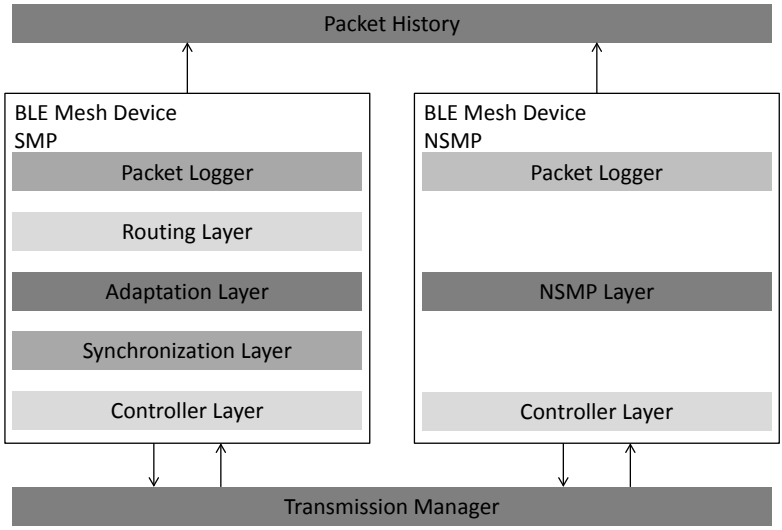


Figure 5.1: *An overview of the simulator.*

The controller layer acts as an actual BLE controller. The purposes of this layer are to handle advertising and scanning according to command, and also to log the power consumption of the device. The controller layer is further responsible for providing the physical position of the BLE device.

The NSMP layer implements the model for the NSMP according to the known behavior of this protocol.

The synchronization layer maintains reception and transmission windows by consuming and generating time beacons. This layer is also responsible for logging any information related to device synchronization, such as skewness, window duration and time beacon information.

The adaptation layer includes transmission power tuning, retransmission control and the time beacon rate. The adaptation layer is also responsible for consuming and generating time beacon acknowledgments, as well as maintaining the acknowledgment table.

The routing layer controls how data packets are transported in the mesh network. Any data packet that is addressed to the device is handled by the routing layer and delivered to a higher instance.

The topmost part of the device is the packet logger which is used to output any received data in a human readable way.

The system is simulated in free space with no interference from other wireless devices.

5.2 Models

This subsection presents some essential models which are used in the simulation.

5.2.1 Network

The network model is designed so that devices and transmissions are represented as mathematical nodes and edges. The physical appearance of a device is consequently a singular point in a 2-dimensional space, whereas a transmission is modeled as a line. The transmission manager is in a physical context the space that encapsulates the whole network (Figure 5.2).

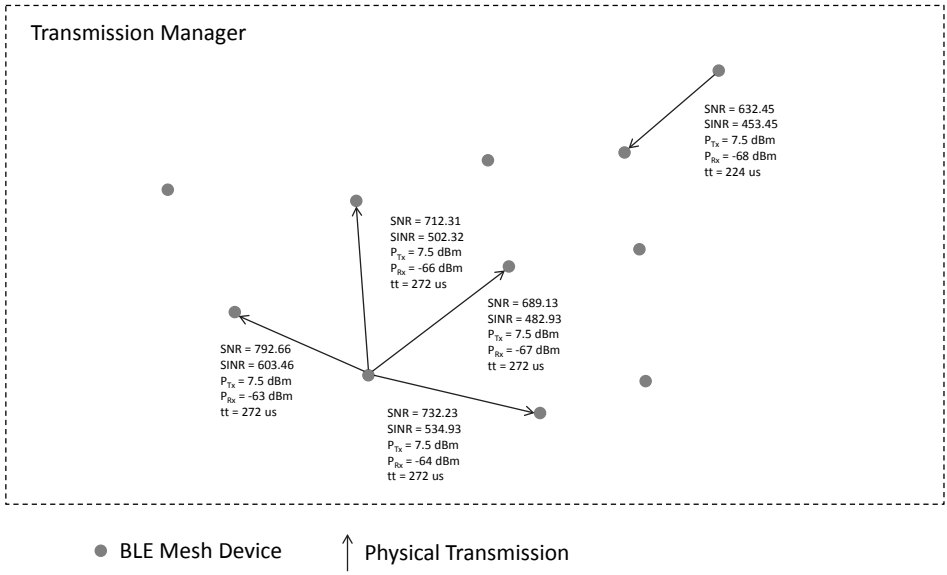


Figure 5.2: *Exemplary network in the simulator.*

5.2.2 Transmissions

A physical transmission is modeled as a set of edges between the transmitter and every receiver in the virtual space. Each receiver in the virtual space will obtain one unique edge for every new transmission. Initially, each transmission edge have

$$\begin{cases} P_{RX} = P_{Tx} - \text{pathLoss}(d) \quad [\text{dBm}] \\ \text{SNR} = \text{SINR} = P_{RX}/S_i \end{cases},$$

where P_{TX} is the transmission power, P_{RX} is the received signal strength (RSSI),

S_i is the receiver sensitivity and d is the distance between the receiver and the transmitter. The path loss of a transmission is calculated as

$$\text{pathLoss}(d) = \max(0, 20\log(f) + 10\eta\log(d) + \text{Pf}(\kappa) - 28) \quad [\text{dB}],$$

where f is the transmission frequency in MHz, η is the distance power loss coefficient, d is the distance in meters, and $\text{Pf}(\kappa)$ is the floor penetration factor for κ floors. The values of these coefficients are given in the table 5.1 [Salas, 2014, pp. 37–39]. Using this table, the path loss function can be simplified as

$$\text{pathLoss}(d) = \max(0, 40 + 25\log(d)) \quad [\text{dB}].$$

The model that is used to determine whether or not a transmission can be received is to check the block error probability (BLEP). A transmission can only be received if

$$\text{BLEP}(\text{BER}(\sigma), n) < X,$$

where σ is the SINR, n is the number of bits in the message, X is a uniformly distributed random value between 0 and 1, and $\text{BER}(\sigma)$ is the bit error rate. The values of $\text{BER}(\sigma)$ and $\text{BLEP}(p_e, n)$ are calculated as

$$\text{BER}(\sigma) = \frac{1}{2}\left(1 - \frac{1}{\sqrt{1 + \frac{1}{\sigma}}}\right),$$

$$\text{BLEP}(p_e, n) = 1 - (1 - p_e)^n,$$

where p_e is the bit error probability for a fixed SINR [Ghnimi, 2011].

Table 5.1: *The path loss coefficients used in the simulator.*

Parameter	Value
f	2400 MHz
η	25
$\text{Pf}(\kappa)$	0 (single floor assumed)

5.2.3 Interference

When multiple BLE devices transmits data at the same time they will, according to the system model, interfere with each other. This effect is the only source of interference that is included in the simulator. When a new packet is added to the transmission manager during an ongoing transmission, then the SINR value of all simultaneous transmissions is updated. First the interference power, I, of a particular transmission edge is obtained by adding the received signal strength, P_{RX} for all other edges that are connected to the same receiver. The SINR value of

that specific edge is then set to

$$\text{SINR} = P_{RX}/(I + S_i),$$

where P_{RX} is the edge's signal strength on the receiver side, and S_i is the sensitivity of the receiving node. In this way, the probability of successfully receiving a transmission is rapidly reduced when more transmissions are added to the same channel.

5.2.4 Power Consumption

The power consumption of a device is modeled as the function

$$\mathrm{P}(t) = 1_{\mathrm{TX}}(t)\mathrm{P}_{\mathrm{adv}}(t) + 1_{\mathrm{RX}}(t)(1 - 1_{\mathrm{TX}}(t))\mathrm{P}_{\mathrm{scan}}(t),$$

where $\mathrm{P}_{\mathrm{adv}}(t)$ is the power consumption at time t when transmitting, and $\mathrm{P}_{\mathrm{scan}}(t)$ is the power consumption at time t when receiving[1]. The values of the functions $1_{\mathrm{TX}}(t)$ and $1_{\mathrm{RX}}(t)$ are set to 0 or 1 depending on the state of the transmitter and the receiver, respectively. That is, if the transmitter is turned on, then $1_{\mathrm{TX}}(t)$ is 1, otherwise it is 0. $\mathrm{P}_{\mathrm{adv}}(t)$ and $\mathrm{P}_{\mathrm{scan}}(t)$ are in the simulation calculated as the linear functions

$$\mathrm{P}_{\mathrm{adv}}(t) = P_{advBase} + C_{adv}\mathrm{P}_{\mathrm{TX}}(t),$$

$$\mathrm{P}_{\mathrm{scan}}(t) = P_{scanBase} + C_{scan}\mathrm{S}_{\mathrm{i}}(t),$$

where $P_{advBase}$ and $P_{scanBase}$ are the base cost for transmitting and receiving, respectively, C_{adv} and C_{scan} are two constant factors, $\mathrm{P}_{\mathrm{TX}}(t)$ is the transmitter output power and $\mathrm{S}_{\mathrm{i}}(t)$ is the receiver sensitivity. It is assumed that the power consumption of the device will display linear behavior for $P_{TX} \in [-20\text{ dBm}, 10\text{ dBm}]$ and $S_i \in [-100\text{ dBm}, 70\text{ dBm}]$ which is the working range of BLE [BLE, 2014a, pp. 16–22].

The values of $P_{advBase}$, $P_{scanBase}$, C_{adv} and C_{scan} are specified in table 5.2.

Table 5.2: *The power consumption coefficients used in the simulator.*

Parameter	Value
$P_{advBase}$	70 mW
$P_{scanBase}$	55 mW
C_{adv}	15
C_{scan}	-4.65 $\cdot 10^9$

The values of $P_{advBase}$, $P_{scanBase}$, C_{adv} and C_{scan} are derived from datasheets by Texas Instruments [Tex, 2013a] [Tex, 2013b].

[1] The reader should notice that the power consumed by the receiver is zero while the device is transmitting. This is because the device cannot receive and transmit at the same time.

5.3 Definitions

This subsection presents some definitions that are useful in the context of the simulation.

5.3.1 Packet Delivery Ratio

The packet delivery ratio (PDR) is defined as the number of packets received by all destinations, N_r, over the number of packets sent by all sources, N_s.

$$PDR = \frac{N_r}{N_s}$$

The PDR will later be used as a metric for transmission reliability.

5.3.2 Message Collision

A message collision is defined as a transmission edge that fails to be received due to interference from other ongoing transmissions. A message collision is said to have taken place if a transmission is received with

$$\text{BLEP}(\text{BER}(\sigma), n) \geq p,$$

and

$$\text{BLEP}(\text{BER}(\sigma), n) < p,$$

where σ is the SINR, n is the number of bits in the message, p is set to X, and X is a uniformly distributed random value between 0 and 1.

5.3.3 Collision Ratio

The collision ratio is defined as the ratio between received transmissions with

$$\text{BLEP}(\text{BER}(\sigma), n) \geq p,$$

and transmissions with

$$\text{BLEP}(\text{BER}(\sigma), n) < p,$$

where σ is the SINR, n is the number of bits in the message, p is set to X, and X is a uniformly distributed random value between 0 and 1. In other words, the collision ratio is the number of transmissions that fails because of interference, over

the number of transmissions that would be receivable if interference is ignored. The collision ratio will later be used as a metric for data traffic load.

5.3.4 Power Delay Product

To measure the latency between different protocols, or configurations, the power delay product, D, is used. This product is defined as

$$D = l \cdot \overline{P},$$

where l is the massage latency, and $\overline{P}$ is the average power consumption of a device. This metric enables fair comparisons between two protocols, or configurations. For a responsive, low power system, the target is to have as small power delay product as possible.

5.4 Assumptions

All simulations are performed in the Ericsson internal simulator for BLE. The parameters that are used in the simulator are summarized in Table B.1, which is found in Chapter B. The input to the simulator is provided by iterating over a set of values for one simulation parameter. All other deviations from the parameter summary will be explicitly mentioned in the simulation results. All transmissions for every simulation are sent over a single advertising channel.

5.5 Deployment

Two different deployments are tested in the simulator: grid deployment and random deployment. The grid deployment is used to measure the performance of the mesh protocol in an industrial building. For this scenario, 36 devices had been perfectly aligned in a grid so that the whole building had access to the network. The random deployment is targeted to demonstrate that the mesh protocol can still provide good performance even though the nodes are not perfectly aligned. The random deployment contains the same amount devices.

5.5.1 Grid

The grid deployment is composed of 36 devices which are uniformly distributed in a rectangular 6x6 grid. The distances between two adjacent devices are set to 16.67 m so that the grid covers a 100x100 m building. Each device is initially assigned a unique address between 0 and 35. The addresses are assigned in ascending order from the lower-left corner to the upper-right corner (Figure 5.3).

5.5.2 Random

The random deployment is also composed of 36 devices. The total area is, however, reduced to 75x75 m. Each device is randomly deployed at the coordinate

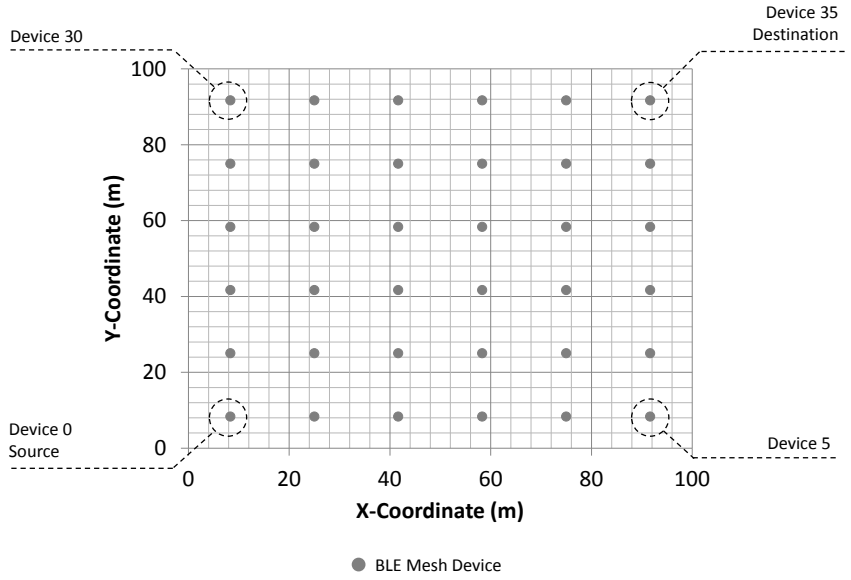


Figure 5.3: *An overview of the grid deployment.*

(X,Y), where X and Y are two uniformly distributed random variables[2]. Each device is initially assigned a unique address between 0 and 35. The exact deployment and the device addresses are shown in Figure 5.4.

5.6 NSMP Tuning

The first objective is to find a suitable scanning interval (T_i) and duty cycle (D_{scan}) for the non-synchronized mesh protocol. This procedure is required to enable fair comparisons between the non-synchronized and the synchronized mesh protocol. The approach for this is to launch two related simulations. The first simulation is designed to show the PDR for different selections of T_i. The T_i that provides the highest PDR will then be used for the subsequent simulations. D_{scan} is in this simulation fixed at 20 %.

The next simulation is targeted to show how the PDR varies for different selections of D_{scan}. T_i is here fixed to the best value obtained from the first simulation. The output from this simulation is later used to design three different configurations that will be compared to the SMP.

These two simulations are only run in the grid deployment. The PDR is measured by sending 8 flooding messages from device 0 to device 35. The packets are sent 5 seconds apart, starting 80 seconds after the simulation start. Each simulation

[2]The random deployment is generated only once.

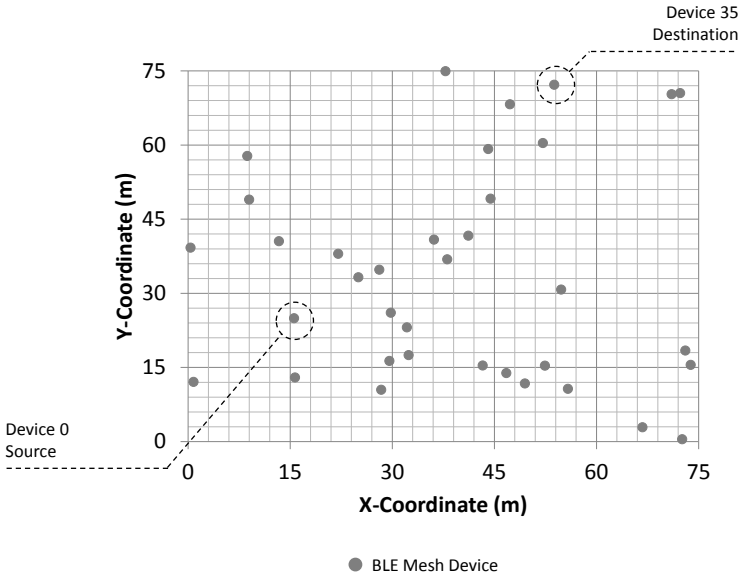


Figure 5.4: *An overview of the random deployment.*

runs for 120 seconds with 50 different seeds[3].

5.7 Synchronization

The next objective is to evaluate the synchronization layer of the synchronized mesh protocol. The simulation input is the transmission window duration (T_{TX}), time beacon reciprocal rate (λ_{TB}), sleep clock drifting (ρ) and transmission power (P_{TX}). The wanted results are the collision ratio, power consumption and reception window duration.

Each simulation runs for 120 seconds with 10 different seeds. The simulations only utilizes the grid deployment. The mesh initiator is the device with address 0.

5.8 Adaptation

The third objective is to evaluate the adaptation layer with various configurations. The simulation input is the desired probability, p, to reach the desired number of receivers, u. The output is the transmission power, the number of retransmissions, the collision ratio, and the PDR.

When measuring the PDR, 8 packets are sent across the network using flooding.

[3] For each seed, the messages were sent according to the described procedure.

The packets are sent 5 seconds apart, starting 80 seconds after the mesh is initialized. The sources of the packets are `[0,5,30,35,0,5,30,35]` and the destinations are `[35,30,5,0,35,30,5,0]`. Each simulation runs for 120 seconds with 25 different seeds. Only the grid deployment is used for these simulations.

5.9 Message Flooding

The fourth objective is used to evaluate the performance of message flooding in different scenarios.

First, the PDR and the power consumption is measured for different selections of T_{TX} in the SMP with the adaptation layer turned on. The same measurement is then performed with the adaptation layer turned off. Together, these results will lay the groundwork for the upcoming comparisons between the synchronized and the non-synchronized mesh protocol. This initial simulation is only run with the grid deployment.

Next, the collision ratio, the power consumption, the message latency, the number of hops, and the PDR are compared using different configurations. Here the simulation inputs are the selection of mesh protocol (SMP/NSMP), the state of the adaptation layer (enabled/disabled), the original size of the scanning window (T_{TX} and D_{scan}), the receiver sensitivity (S_i), and the selection of deployment (grid/random).

The PDR is measured by sending 8 packets from device 0 to device 35. Each simulation runs for 120 seconds run with 50 different seeds.

5.10 Message Routing

The final objective is to demonstrate the performance of packet routing with the AODV based routing algorithm. This simulation is executed by letting device 0 establish a route to device 35, and then let a packet be sent via the active route to the destination. The input for this simulation is to change the minimal accepted signal strength ($P_{th,route}$), the state of the adaption (enabled/disabled), and the selection of deployment (grid/random). The output of the simulation is the discovery latency, the lifetime of a route, the message latency, and the PDR.

The PDR is measured by sending 8 packets from the device 0 to the device 35 after a route is established. Each simulation runs for 120 seconds run with 75 different seeds.

6 Results

This chapter contains the results from the simulations that are described in chapter 5. This chapter will moreover discuss some nontrivial and unexpected results that are presented.

6.1 NSMP Tuning

The initial simulation of the NSMP shows that the PDR stays over 90 % when T_i is set within the range $T_{i,min}$ = 5 ms and $T_{i,max}$ = 100 ms. D_{scan} is here set to 20 % as mentioned in Section 5.4. As shown in Figure 6.1 it is hard to distinguish how small changes of T_i will affect the PDR, principally because of the severe up and downs in the chart. The closing decision for the following simulations is to set T_i = 45 ms because of the supreme value near this point[1]. The supreme value is marked with a dashed line in Figure 6.1. Higher resolution of the horizontal axis can possibly present event better values of T_i. However, the effect of fine-tuning T_i is not expected to produce major changes in the PDR.

The next simulation shows that the NSMP can provide high PDR even for outstandingly low values of the duty cycle. Figure 6.2a shows the resulting PDR as a function of the duty cycle, D_{scan}. This simulation also shows that there is a clear linear relation between the D_{scan} and the power consumption. This relation is expected since the total power consumption must be $P = D_{scan}P_{RX} + D_{adv}P_{TX}$, where D_{scan} is the scanning duty cycle and D_{adv} is the advertising duty cycle[2]. The actual result can be seen in Figure 6.2b.

[1] The supreme value is not necessarily the maximal value.

[2] The reader should note that the value of D_{adv} is unknown.

Table 6.1 shows the power consumption and the PDR for some selections of D_{scan} that is used for later evaluations.

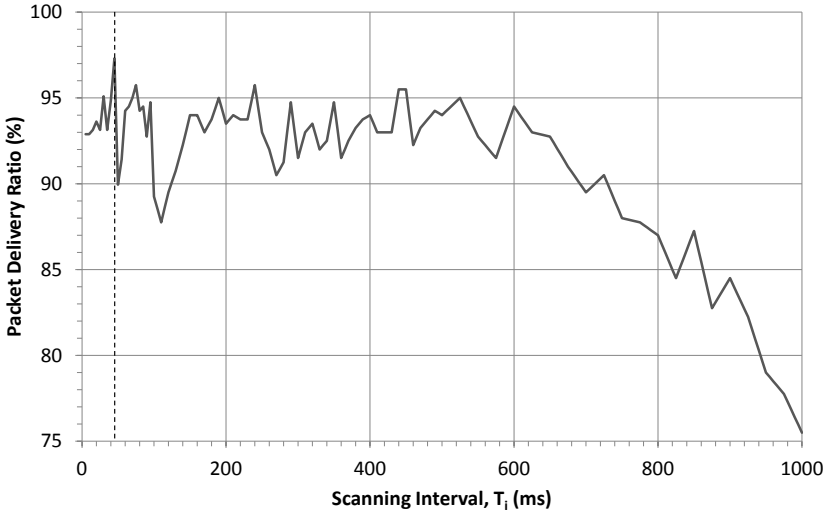
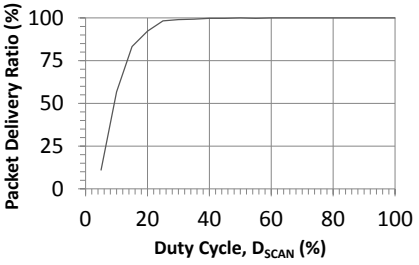
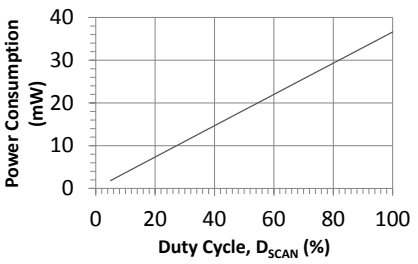


Figure 6.1: *A graph showing scanning interval (T_i) vs. packet delivery ratio (PDR).*



(a) *A graph showing packet delivery ratio (PDR) vs. duty cycle (D_{scan}).*



(b) *A graph showing power consumption vs. duty cycle (D_{scan}).*

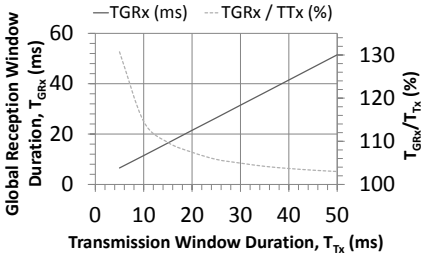
Figure 6.2: *Two graphs; one showing the power consumption, the other showing the PDR, for different selection of the duty cycle (D_{scan}).*

Table 6.1: *The NSMP configurations used for later evaluations.*

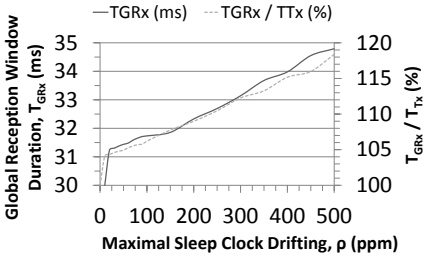
T_i	D_{scan}	**PDR**	**Power consumption**
45 ms	10 %	56.8 %	3.68 mW
45 ms	20 %	92.3 %	7.35 mW
45 ms	30 %	99.0 %	11.0 mW

6.2 Synchronization

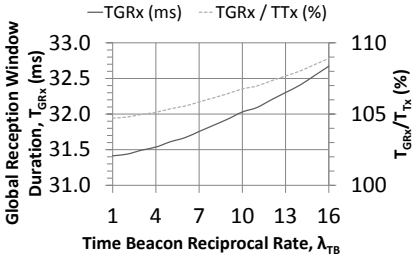
The first simulations of the SMP shows how the global reception window (T_{GRX}) is affected by various configurations. The solid line in Figure 6.3a shows that the widest global reception window among all devices is always less than 1.5 ms longer than the transmission window regardless of the selected T_{TX}. This result verifies that periodical timing information is received and processed correctly as described in Section 4.3. The dashed line shows the duration of the T_{GRX} relative to the T_{TX}. The clock drifting in each device is here assumed to be less than 50 ppm, whilst the time beacon reciprocal rate, λ_{TB}, is set to 4.



(a) *Global reception window duration (T_{GRX}) vs. transmission window duration (T_{TX}).*



(b) *Global reception window duration (T_{GRX}) vs. maximal sleep clock drifting.*



(c) *Global reception window duration (T_{GRX}) vs. time beacon reciprocal rate (λ_{TB}).*

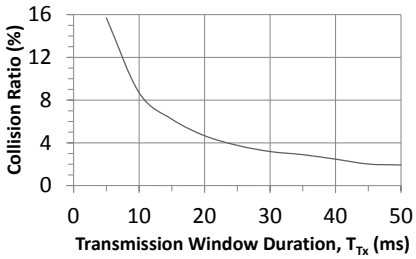
Figure 6.3: *Three graphs showing the global reception window duration for different inputs.*

Next is to investigate the effect of having an imprecise sleep clock. Figure 6.3b shows the duration of the global reception window as a function of the maximal clock drifting (solid line). This simulation exposes a linear relation between the duration of the global reception window and the maximal clock drifting. The exception is when the clock drifting is 0 ppm, since this removes the need of window translation.

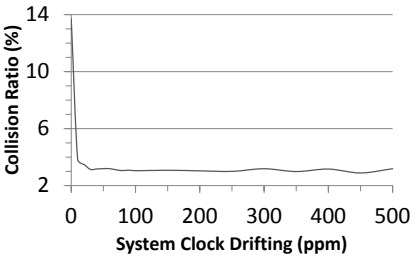
The simulation further inspects how different values of λ_{TB} affect the global reception window. It can be seen on the solid line in Figure 6.3c that this result displays a similar behavior as the preceding result. This is the expected outcome since the effect of not receiving TBs within some time is analog to having a more imprecise sleep clock.

In addition to the above results, it is also examined how the network collision ratio is affected for the stated configurations. The results of the collision ratio are shown in Figure 6.4. The conclusions drawn from this figure are:

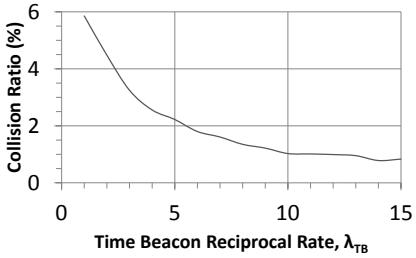
1. There is an inverse relationship between the transmission window duration, T_{TX}, and the collision ratio (Figure 6.4a).
2. The lower bound of the collision ratio is obtained when the clock drifting is non-zero (Figure 6.4b).
3. There is an inverse relationship between the time beacon reciprocal rate, λ_{TB}, and the collision ratio (Figure 6.4c).



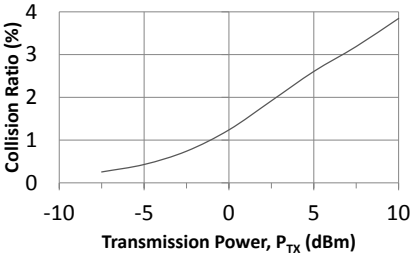
(a) Transmission window duration (T_{TX}) vs. collision ratio.



(b) Sleep clock drifting vs. collision ratio.



(c) Time beacon reciprocal rate (λ_{TB}) vs. collision ratio.



(d) Transmission power (P_{TX}) vs. collision ratio.

Figure 6.4: *Four graphs showing the collision ratio for different inputs.*

The first partial result is intuitive, since the effect of halving the T_{TX} is likely to double the collision rate. This is a typical traffic jam situation that occurs in many practical contexts where the user reduces the upper bound of a traffic flow through a shared channel. The next partial result can be explained by the

protocol properties of SMP, where the node is only allowed to send messages in discrete time slots (see Section 4.2.1). Clock imprecision allows nodes to transmit data between the discrete time slots in the TXWIN of another node, and thereby reduces the collision rate. The final partial result can be explained as a reduction of outgoing transmissions per node when the time beacon rate decreases.

The next simulation is targeted to display the relation between the transmission powers and the collision ratio. The result of this simulation is shown in Figure 6.4d. One conclusion from this simulation is that the collision ratio can be reduced by lowering the transmission power of all devices. This is basically the motive for including an adaptation layer in the SMP architecture.

This simulation further shows that the selection of transmission power has little effect on the overall power consumption. The difference in total power consumption when using $P_{TX} = -7.5$ dBm (lowest possible for this scenario) compared to $P_{TX} = 10$ dBm is less than 1 %, which can be seen in Figure 6.5. Figure 6.6 also reveals that the major power consumption is consumed by the receiver.

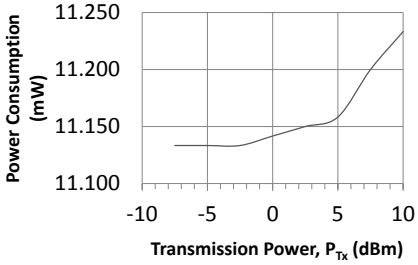
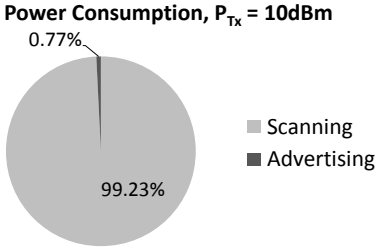
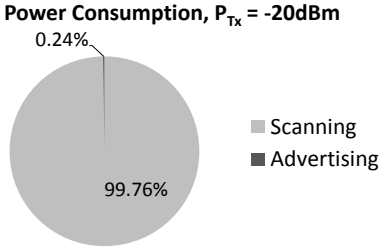


Figure 6.5: *A graph showing power consumption vs. transmission power.*



(a) *Power consumption distribution of SMP with $P_{TX} = 10$ dBm.*



(b) *Power consumption distribution of SMP with $P_{TX} = -20$ dBm.*

Figure 6.6: *The distribution between advertising power consumption and scanning power consumption of SMP.*

The configuration that is used for the following simulations is $T_{cycle} = 150$ ms, $\lambda_{TB} = 3$ and $P_{TX} = 7.5$ dBm. The effect of varying the transmission window (T_{TX}) is further investigated in Section 6.4.1.

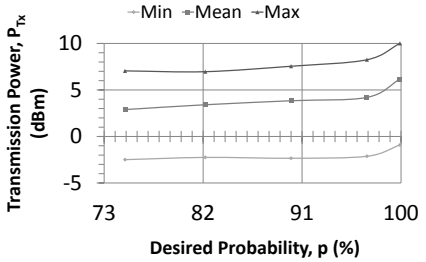
6.3 Adaptation

The simulation of the adaptation layer substantially shows that it is possible to adjust the PDR by changing the desired number of neighbors, u, and the probability of reaching u neighbors, p. Any variation of these parameters also affects the transmission power and the number of retransmissions of each device. This further affects the collision ratio and therefore the overall performance of the network. Small values of u and p are thereby preferred in a highly dense network, whereas a sparse network would require a higher value of u and p.

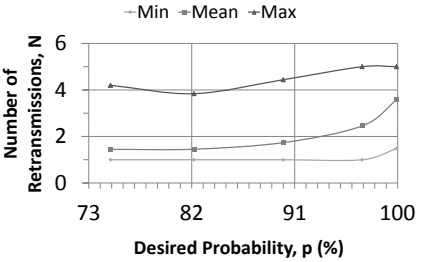
The transmission power, number of retransmissions, and collision ratio for different selections of p are shown in Figure 6.7a, Figure 6.7b, and Figure 6.7c, respectively. The value of p is here set to

$$p = 1 - \sqrt[n]{1 - P}$$

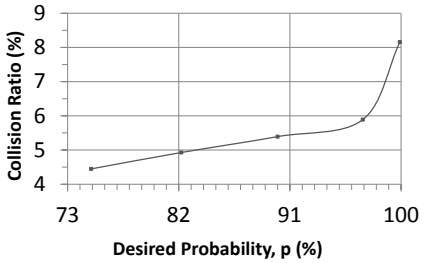
where $P = 0.999$ and $n \in \{1, 2, 3, 4, 5\}$.



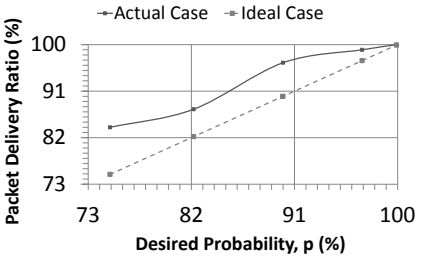
(a) Desired probability (p) vs. transmission power (P_{TX}).



(b) Desired probability (p) vs. number of retransmissions (N).



(c) Desired probability (p) vs. collision ratio.



(d) Desired probability (p) vs. packet delivery ratio (PDR).

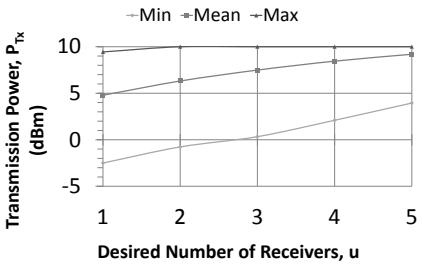
Figure 6.7: *Some relations between desired probability, p, and various simulator outputs.*

Figure 6.8a, Figure 6.8b and Figure 6.8c show the transmission power, the num-

ber of retransmissions and the collision ratio for different selections of u.

One interesting consequence of selecting $u = 2$ is that an almost linear relation between p and the PDR is obtained (Figure 6.7d). The relation between u and the PDR is, however, not as trivial. This relation is shown in Figure 6.8d.

The configuration that is selected for the upcoming simulations is $u = 2$. The motivation for this choice was mainly because of the linear relation between p and the PDR. This allows the system to provide a PDR close to the selected value of p. If the user accepts a low PDR in the system, then p can be given a low value. This can be useful if the target is low channel occupancy. If, on the other hand, the user requires the system to provide a high PDR, then p can be set to a high value, such as $p = 0.999$, which is the case for the upcoming simulations.



(a) Desired number of neighbors (u) vs. transmission power (P_{TX}).

(b) Desired number of neighbors (u) vs. number of retransmissions (N).

(c) Desired number of neighbors (u) vs. collision ratio.

(d) Desired number of neighbors (u) vs. packet delivery ratio (PDR).

Figure 6.8: *Some relations between desired number of neighbors, u, and various simulator outputs.*

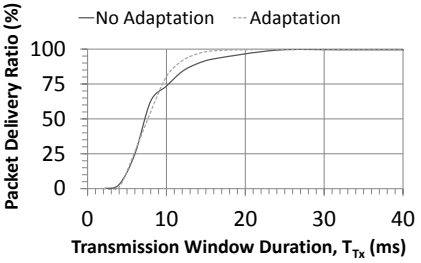
6.4 Message Flooding

This section presents the results from the evaluation of message flooding.

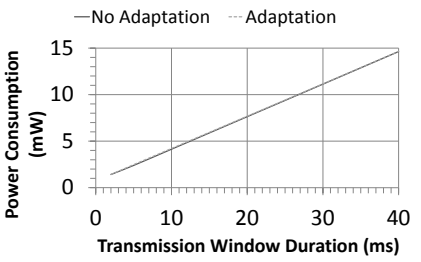
6.4.1 SMP Transmission Window Tuning

As stated in Section 5.9, the first objective is to measure the power consumption and the PDR for different selections of T_{TX} in the SMP. Figure 6.9a and Figure 6.9b show the results of this simulation with and without the adaptation layer.

Notably, the PDR stays above 99 % even for T_{TX} = 15 ms when the adaptation layer is active. This implies that the SMP provides an equally high PDR compared to the NSMP even when the power consumption is approximately one third (recall Section 6.1).



(a) Packet delivery ratio (PDR) for different selections of T_{TX}.



(b) Power consumption for different selections of T_{TX}.

Figure 6.9: *The packet delivery ratio and power consumption for different configurations of SMP within the grid deployment.*

6.4.2 Configuration Selection

The next objective is to compare the performance of message flooding between the synchronized and the non-synchronized mesh protocol. For this, nine different configurations are defined: six for the SMP and three for the NSMP. The SMP is simulated with T_{TX} set to [7.5 ms, 15 ms, 30 ms] and, in addition, with the adaptation layer set to enabled or disabled. The selection of T_{TX} corresponded to a scanning window duty cycle of approximately 5 %, 10 % and 20 %. The NSMP is simulated with the configurations stated in Table 6.1. The scanning duty cycles of NSMP is set to [10 %, 20 %, 30 %]. This is to show that the NSMP can only provide an equally high PDR as the SMP if the scanning duty cycle is increased. All other parameters are set to the assumed values in Section 5.4 The simulation results are presented in Section 6.4.3 to Section 6.4.6.

6.4.3 Grid Deployment with High-Gain Receiver Mode

The first simulation is in the grid deployment using high-gain receiver mode (see Section 5.4). Table 6.2 shows the simulation results for the NSMP and the SMP

with the adaptation layer disabled.

Table 6.2: *The simulation results from flooding simulation, using grid deployment with high-gain receiver mode. SMP adaptation layer is disabled.*

Result	**NSMP**			**SMP**		
Duty Cycle (%)	10	20	30	5	10	20
Collision Ratio (%)	3.40	4.87	5.46	26.2	21.1	14.9
Power Consumption (mW)	3.68	7.35	11.0	2.34	3.97	7.35
Latency (ms)	482	325	242	901	501	259
Number of Hops	4.86	4.12	3.76	3.33	2.95	2.82
PDR (%)	56.8	92.3	99.0	55.1	91.0	99.4
Power delay product (mWs)	1.77	2.39	2.66	2.11	1.99	1.90

The results show that the SMP outperforms the NSMP in terms of power consumption and PDR[3]. Interestingly, the SMP can provide an almost equal PDR even with half the power consumption.

In terms of message latency in respect to power consumption, the results are very similar. It should be noticed that the NSMP has slightly lower latency with respect to the power consumption for short duty cycles, which is seen on the power delay product. It is, however, possible to slightly reduce the message latency of the SMP by decreasing T_{cycle}. This will, on the other hand, affect the length of the transmission window if the duty cycle is kept. A reduced transmission window conversely increases the message latency as shown in Table 6.2. This conflict makes it hard to determine whether or not the power delay product can be improved for the SMP. It must also be recalled that the NSMP violates one of the timing constraints of BLE, as stated in Section 3.4. For now, this issue will be ignored since the timing violation only occurs between the last retransmissions. The message latency is therefore not expected to significantly increase even if this violation is rectified.

The next results show a similar comparison, but with the SMP adaptation layer set in enabled or disabled state Table 6.3.

For this particular scenario, the adaptation layer does not contribute to any major increase of PDR. It is however shown that the adaptation layer increases the collision ratio drastically. This is expected since more packets are sent, and hence the number of transmission edges are generated. The adaptation layer also makes the devices consume slightly more power. The reason behind the increased power consumption is most likely because of the transmitted TBAs, along with the increased transmission power. This simulation however indicates that the message latency is reduced when the adaptation layer is enabled. This also gives a slightly lower power delay product, compared to when the adaptation is disabled. The

[3]The reader should notice that the interesting values for the comparison is the PDR with respect to the power consumption.

Table 6.3: Comparision between simulation results of adaptive, and non-adaptive SMP flooding, using grid deployment with high-gain receiver mode.

Result	Non-adaptive			Adaptive		
Duty Cycle (%)	5	10	20	5	10	20
Collision Ratio (%)	26.2	21.1	14.9	31.0	27.1	19.3
Power Consumption (mW)	2.34	3.97	7.35	2.39	4.04	7.40
Latency (ms)	901	501	259	743	402	218
Number of Hops	3.33	2.95	2.82	2.30	2.16	2.14
PDR (%)	55.1	91.0	99.4	48.0	97.1	100
Power delay product (mWs)	2.11	1.99	1.90	1.78	1.62	1.61
P_{TX} (dBm)	7.5	7.5	7.5	9.94	9.64	9.52
Number of Retransmissions	3	3	3	2.78	4.55	4.70

reason for this is that the number of required hops between the source and the destination is reduced, whereas the latency per hop is practically fixed.

6.4.4 Grid Deployment with Standard Receiver Mode

The next simulation is also in the grid deployment but is instead using the standard receiver mode (see Section 5.4). For this scenario the NSMP practically fails to work because of the low PDR, as seen in Table 6.4. The SMP however provides acceptable PDR even for short duty cycles. Table 6.4 compares the NSMP and the SMP with the adaptation layer disabled.

Table 6.4: The simulation results from flooding simulation, using grid deployment with standard receiver mode. SMP adaptation layer is disabled.

Result	NSMP			SMP		
Duty Cycle (%)	10	20	30	5	10	20
Collision Ratio (%)	0.143	0.455	0.768	9.43	6.75	4.77
Power Consumption (mW)	3.02	6.04	9.06	1.94	3.30	6.10
Latency (ms)	-	800	673	2160	1370	683
Number of Hops	-	7.30	7.37	8.02	7.30	7.18
PDR (%)	0.750	8.75	34.8	42.2	88.1	98.4
Power delay product (mWs)	-	4.83	6.10	4.19	4.52	4.17

The next results from this simulation further shows that the adaptation layer can give a major performance boost in some situations; especially for the shorter selections of T_{TX}. Table 6.5 compares the simulation results with, and without, the adaptation layer.

Table 6.5: Comparision between simulation results of adaptive, and non-adaptive SMP flooding, using grid deployment with standard receiver mode.

Result	Non-adaptive			Adaptive		
Duty Cycle (%)	5	10	20	5	10	20
Collision Ratio (%)	9.43	6.75	4.77	13.1	10.7	9.07
Power Consumption (mW)	1.94	3.30	6.10	2.01	3.36	6.22
Latency (ms)	2160	1370	683	1980	1110	496
Number of Hops	8.02	7.30	7.18	5.75	5.19	4.69
PDR (%)	42.2	88.1	98.4	37.3	97.1	100
Power delay product (mWs)	4.19	4.52	4.17	4.00	3.73	3.09
P_{TX} (dBm)	7.5	7.5	7.5	9.94	9.25	9.13
Number of Retransmissions	3	3	3	3.20	4.78	5.00

6.4.5 Random Deployment with High-Gain Receiver Mode

The following simulation is in the random deployment with the high-gain receiver mode. The results of this simulation are shown in Table 6.6.

The row that should receive extra attention is the PDR. It can be observed that the PDRs for the NSMP are considerably higher than results presented in Section 6.4.3. The reason for this can partly be explained by the decreased deployment area, which directly increases the network density around the source and the destination. The increased PDR can moreover be explained by the decreased number of hops between the two end-nodes. Each hop count, n, will introduce a particular packet loss probability, p_i, where i is the hop index. The total risk of losing a packet along a path of n hops can thereby be calculated as

$$\mathrm{P}(n) = 1 - \prod_{i=1}^{n}(1 - p_i)$$

It may further be shown that $\mathrm{P}(n)$ increases with n as long as $p_i > 0$:

$$\mathrm{P}(n+1) = 1 - \prod_{i=1}^{n+1}(1 - p_i) = 1 - (1 - p_{n+1})\prod_{i=1}^{n}(1 - p_i) =$$

$$= 1 - \prod_{i=1}^{n}(1 - p_i) + p_{n+1}\prod_{i=1}^{n}(1 - p_i) = \mathrm{P}(n) + p_{n+1}\prod_{i=1}^{n}(1 - p_i) \geq \mathrm{P}(n)$$

The reader should however take into account that the value of each separate p_i may have bigger impact on the total packet loss than the number of hops n.

The following results compares the performance of SMP with, and without, the

Table 6.6: The simulation results from flooding simulation, using random deployment with high-gain receiver mode. SMP adaptation layer is disabled.

Result	NSMP			SMP		
Duty Cycle (%)	10	20	30	5	10	20
Collision Ratio (%)	6.65	8.14	9.50	39.3	31.9	22.1
Power Consumption (mW)	3.68	7.35	11.0	2.34	3.97	7.35
Latency (ms)	298	197	147	454	269	144
Number of Hops	3.28	2.66	2.19	1.45	1.27	1.28
PDR (%)	92.5	99.5	100	94.5	99.5	100
Power delay product (mWs)	1.11	1.45	1.62	1.06	1.07	1.06

adaptation layer. These results are presented in Table 6.7.

Table 6.7: Comparision between simulation results of adaptive, and non-adaptive SMP flooding, using random deployment with high-gain receiver mode.

Result	Non-adaptive			Adaptive		
Duty Cycle (%)	5	10	20	5	10	20
Collision Ratio (%)	39.3	31.9	22.1	65.4	53.5	28.1
Power Consumption (mW)	2.34	3.97	7.35	2.40	4.02	7.42
Latency (ms)	454	269	144	445	272	194
Number of Hops	1.45	1.27	1.28	1.00	1.12	1.73
PDR (%)	94.5	99.5	100	96.8	100	100
Power delay product (mWs)	1.06	1.07	1.06	1.07	1.09	1.44
P_{TX} (dBm)	7.5	7.5	7.5	9.96	9.07	8.96
Number of Retransmissions	3	3	3	4.92	4.73	4.20

6.4.6 Random Deployment with Standard Receiver Mode

The following simulation shows the performance of message flooding with the receiver set to standard mode, still using the random deployment. The results of this simulation are shown in Table 6.8.

These results once again show that the SMP can provide an equally high PDR compared to the NSMP; even with one-third of the power consumption. The message latency for this scenario is, however, considerably higher for the SMP.

Table 6.9 shows the results of the same simulation, but compares the SMP with, and without, the adaptation layer.

The PDR for the shortest selection of T_{TX} was in this scenario significantly improved by the adaptation layer. The adaptation algorithm however fails to improve the PDR for the longest T_{TX} because of the same problem that is discussed

Table 6.8: The simulation results from flooding simulation, using random deployment with standard receiver mode. SMP adaptation layer is disabled.

Result	NSMP			SMP		
Duty Cycle (%)	10	20	30	5	10	20
Collision Ratio (%)	1.77	2.56	2.90	17.4	13.4	8.92
Power Consumption (mW)	3.02	6.04	9.06	1.99	3.29	6.07
Latency (ms)	522	402	313	1062	642	299
Number of Hops	4.73	4.39	4.07	3.75	3.31	3.10
PDR (%)	25.3	75.0	87.3	72.0	96.5	100
Power delay product (mWs)	1.58	2.43	2.84	2.11	2.11	1.81

Table 6.9: Comparision between simulation results of adaptive, and non-adaptive SMP flooding, using random deployment with standard receiver mode.

Result	Non-adaptive			Adaptive		
Duty Cycle (%)	5	10	20	5	10	20
Collision Ratio (%)	17.4	13.4	8.92	34.0	22.7	11.1
Power Consumption (mW)	1.99	3.29	6.07	2.00	3.34	6.10
Latency (ms)	1062	642	299	1096	651	408
Number of Hops	3.75	3.31	3.10	2.82	3.00	3.21
PDR (%)	72.0	96.5	100	83.5	99.0	100
Power delay product (mWs)	2.11	2.11	1.81	2.19	2.17	2.49
P_{TX} (dBm)	7.5	7.5	7.5	9.99	9.51	9.22
Number of Retransmissions	3	3	3	4.93	4.90	4.32

in Section 6.4.5. It is also possible to observe that the message latency is somewhat increased when the adaptation layer is enabled.

6.5 Message Routing

This section presents the results from the evaluation of message routing using the AODV-based routing algorithm in SMP.

6.5.1 Grid Deployment with High-Gain Receiver Mode

The first routing simulation is executed using the grid deployment with the high-gain receiver mode selected. The results presented in Table 6.10 are obtained when the adaptation layer is disabled.

One interesting consequence of using the proposed routing algorithm is that the PDR always stays at 100 %. Still, the above results also show that a multi-hop connection sometimes fails to be discovered. The side effect of such events is that

Table 6.10: The simulation results from routing simulation using grid deployment with high-gain receiver mode. SMP adaptation layer is disabled.

Result							
Minimal RSSI (dBm)	-76	-74	-72	-70	-68	-66	-64
RREQ success rate (%)	72.0	100.0	100.0	98.0	96.0	96.0	96.0
RREQ number of hops	3.06	3.64	4.45	5.25	7.13	8.07	9.00
APP route latency (ms)[1]	478	493	504	515	505	500	495
APP packet hops[2]	2.08	2.09	2.10	2.23	2.23	2.22	2.22
PDR (%)	100	100	100	100	100	100	100

no packets were sent, which can overrate the PDR statistics.

It should be pointed out that the message latency is nearly unaffected by the minimal accepted signal strength. The reason for this is doubtless because of the routing algorithm implementation which allows application packets to perform multiple hops in a single transmission. It should further be noticed that the packet latency is not considerably longer than when using message flooding (Section 6.4.3).

The route discovery latency and route lifetime are presented in the below CDF plots, Figure 6.10a and Figure 6.10b, respectively. The conclusion that can be drawn from the CDF plots is that high minimal signal strength will lead to long route lifetime, but also to slightly longer route discovery time.

The following results, presented in Table 6.11, show the same simulation but with the adaptation layer set to enabled.

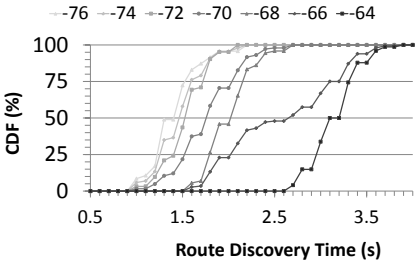
Table 6.11: The simulation results from routing simulation using grid deployment with high-gain receiver mode. SMP adaptation layer is enabled.

Result							
Minimal RSSI (dBm)	-76	-74	-72	-70	-68	-66	-64
RREQ success rate (%)	32.0	64.0	66.7	88.0	84.0	48.0	8.0
RREQ number of hops	3.13	3.56	4.44	5.50	6.90	7.58	9.0
APP route latency (ms)	703	711	696	678	782	734	699
APP packet hops	2.28	2.23	2.34	2.29	2.52	2.33	2.34
PDR (%)	100	100	100	100	100	100	100

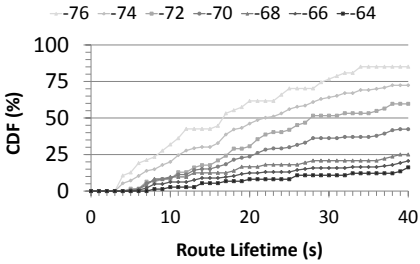
These results illustrates that the adaptation layer significantly reduces the performance of message routing. Figure 6.10c and Figure 6.10d further show that the route discovery time and route lifetime are also deteriorated by the adaptation layer.

[1] Packet latency for application packet.
[2] Number of hops for application packet.

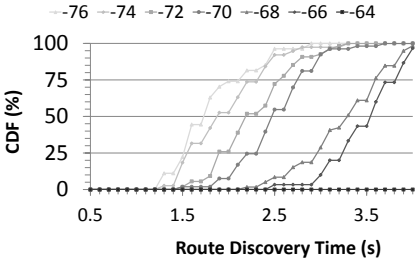
As seen in Figure 6.10d the route lifetime is less than 30 seconds for at least 50 % of the simulation seeds; regardless of the minimal accepted signal strength.



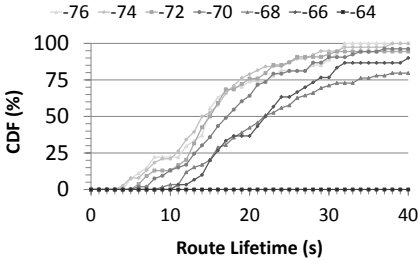
(a) CDF of the route discovery time without adaptation layer using different minimal accepted RSSI.



(b) CDF of the route lifetime without adaptation layer using different minimal accepted RSSI.



(c) CDF of the route discovery time with adaptation layer using different minimal accepted RSSI.



(d) CDF of the route lifetime with adaptation layer using different minimal accepted RSSI.

Figure 6.10: *The route discovery time and route lifetime using different configurations in SMP within the grid deployment scenario.*

It is very rational that the adaptation layer reduces the performance of message routing. The adaptation algorithm described in Section 4.4 works by collecting TBAs from all neighbors of the adapting mesh node. The adapting mesh node later weights all TBAs equally to tune its transmission power and number of retransmissions. For flooding, this strategy is successful because the message propagation is not directed via any specific neighbor. For routing, however, the message propagation strongly depends on whether some particular neighbors can be reached or not. This conflict is probably one of the causes of the displayed performance loss. The results can potentially be improved by introducing custom weights for the collected TBAs when running the adaptation algorithm. This was however not tested in this thesis.

6.5.2 Random Deployment with High-Gain Receiver Mode

The next simulation is executed with the random deployment, still using the high-gain receiver mode. The simulation result with the adaptation layer disabled is presented in Table 6.12.

Table 6.12: *The simulation results from routing simulation using random deployment with high-gain receiver mode. SMP adaptation layer is disabled.*

Result							
`Minimal RSSI (dBm)`	-76	-74	-72	-70	-68	-66	-64
`RREQ success rate (%)`	77.0	92.0	95.9	91.8	97.3	98.7	98.7
`RREQ number of hops`	1.82	2.01	2.07	2.39	3.01	3.42	4.62
`APP route latency (ms)`	304	304	317	300	267	284	299
`APP packet hops`	0.94	0.94	0.94	0.97	0.92	0.96	0.94
`PDR (%)`	100	100	100	100	100	100	100

The above results are very similar to the results presented in Table 6.10. The number of hops and the message latency are however lower in this simulation primarily because of the reduced size of the deployment area. Moreover, the message latency is nearly unaffected by the minimal accepted signal strength; very much alike the previous section. It is further shown in Figure 6.11a and Figure 6.11b that the route discovery time and route lifetime has resembling characteristics as the equivalent simulation in the grid deployment.

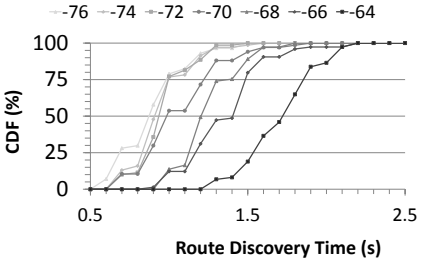
The final results display the output of the same simulation as above, but with the adaptation layer enabled. As shown in Table 6.13 the route discovery success rate was very poor compared to the case without the adaptation layer. In fact, even the discovery number of hops and the message latency were slightly impaired.

It should further be noticed that the route lifetime is extremely low in this simulation. Figure 6.11d shows that over 80 % of all routes break within 40 seconds regardless of the minimal accepted signal strength. The route discovery time, presented in Figure 6.11c, also shows some performance loss compared to the case with the disabled adaption layer.

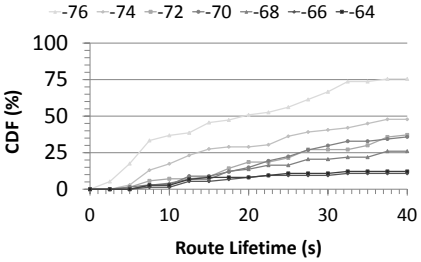
The above results confirm that the current adaptation algorithm does not play well together with message routing. Addition work has to be put into the adaptation layer in order to provide support for the AODV-based routing algorithm.

Table 6.13: *The simulation results from routing simulation using grid deployment with high-gain receiver mode. SMP adaptation layer is enabled.*

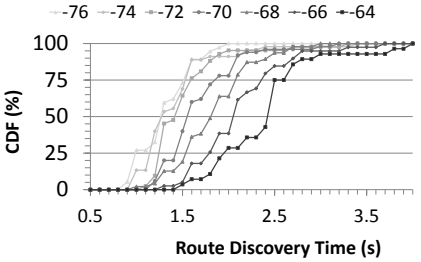
Result							
Minimal RSSI (dBm)	-76	-74	-72	-70	-68	-66	-64
RREQ success rate (%)	54.4	65.2	59.2	68.5	64.4	52.0	37.3
RREQ number of hops	2.30	2.69	3.07	3.50	3.91	4.64	5.29
APP route latency (ms)	486	508	500	482	508	453	443
APP packet hops	1.53	1.76	1.54	1.48	1.54	1.36	1.31
PDR (%)	100	100	100	100	100	100	100



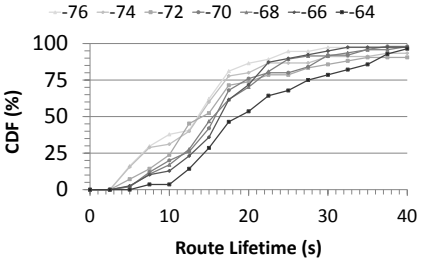
(a) *CDF of the route discovery time without adaptation layer using different minimal accepted RSSI.*



(b) *CDF of the route lifetime without adaptation layer using different minimal accepted RSSI.*



(c) *CDF of the route discovery time with adaptation layer using different minimal accepted RSSI.*



(d) *CDF of the route lifetime with adaptation layer using different minimal accepted RSSI.*

Figure 6.11: *The route discovery time and route lifetime using different configurations in SMP within the random deployment scenario.*

7
Conclusion

The goal of this chapter is to conclude the report by giving a summary of the BLE mesh protocol study. The chapter will thoroughly answer the questions stated in Section 1.2 and, in addition, present some other essential differences between NSMP and SMP. Finally the report will be closed by presenting some ideas of future work that strongly relates to this thesis.

7.1 Summary

The goal of this thesis is initially to study the possibility of establishing a mesh network using BLE communication. The study leads to the conclusion that two separate protocols can be used: NSMP and SMP. The first protocol, NSMP, allows each device to communicate asynchronously via bridges by using message flooding. The second protocol, SMP, conversely defines a synchronization technique, which increases the chance of receiving messages from neighbor nodes. This protocol also defines a flooding algorithm, similar to the one used in NSMP. Moreover, SMP also defines a multi-hop routing technique based on AODV. Additional differences between the protocols are shown in Table 7.1.

The secondary goal of the thesis is to evaluate the performance of the two protocols. This is done by measuring three distinct qualities in a BLE simulator: power consumption, packet delivery ratio and packet latency. The results are obtained by using a simulation model for each corresponding protocol. The model for NSMP is defined in Chapter 3, whereas the model for SMP is defined in Chapter 4. The system model for the whole network is explained in Chapter 5.

The first simulations aim at adjusting the parameters of the NSMP to tune its performance for a grid deployment scenario. The next simulations perform an

Table 7.1: Functional differences between NSMP and SMP.

Feature	NSMP	SMP
Application data format	Any	L2CAP
Communication	Asynchronous	Synchronous
Flooding support	Yes	Yes
Routing support	No	Yes
Scenario adaptive	No	Locally-adaptive

equivalent fine-tuning for the SMP. The protocols are then compared by letting a device flood a message over two different mesh layouts: a grid deployment and a random deployment. Finally, the AODV-based routing algorithm is evaluated in SMP. This simulation also runs in both deployments.

The executed simulations provides satisfying results which can be used to answer the questions of this thesis. A recap of the results for each individual question in Section 1.2 is listed below.

1. *What protocols can be used to establish a mesh network using BLE communication?*

   This report presents two different protocols that are intended for BLE mesh networks: NSMP and SMP. NSMP is a fully asynchronous mesh protocol, whereas SMP is not. A detailed description of the both protocols can be found in Chapter 3 and Chapter 4.

2. *How well do the individual mesh protocols work for various scenarios?*

   The results from Section 6.5.1 and Section 6.5.2 shows that the performance of NSMP and SMP drops severely when the receiver sensitivity is increased. Solving this problem requires node-wise redeployment, altered transmission power or using additional retransmissions. It is typically hard to improve the performance by changing the deployment. BLE devices are usually attached to other objects, which in most cases are placed at specific locations for a reason. The transmission power and number of retransmission are however easy to configure; at least if this is allowed by the device. In NSMP the user can manually change the transmission power to tune the device for a particular scenario. There are however no support for automatic parameter tuning. The number of retransmissions is static in NSMP. In SMP the user is allowed to manually change both the number of retransmissions and the transmission power. This protocol further introduces an adaptation technique which can automatically tune these parameters at run-time. Section 6.5.1 and Section 6.5.2 show that the PDR is radically improved for some scenarios when the adaptation layer is enabled.

3. *How much power is consumed by running the individual mesh protocols?*

It is shown in Section 6.4 that SMP can provide a PDR over 90 % with less than 4 mW average power consumption in both a grid deployment (Section 5.5.1) and a random deployment (Section 5.5.2). To achieve a PDR over 99 % the SMP however requires an average power consumption of around 4-7.5 mW. The NSMP requires more power to offer the same performance. From Section 6.1 it can be derived that the 90 % PDR is reached in the grid deployment when the power consumption is about 3.5-7.5 mW. A PDR over 99 % is not obtained until the average power consumption is around 7.5-11 mW. The random deployment is however beneficial for the NSMP which is shown in Section 6.4.5. The 90 % PDR is then already reached at 3.7 mW, whereas the 99 % limit is reached at around 7.4 mW. It is, however, clear that SMP consumes less power than NSMP in all the simulated flooding scenarios. More detailed information can be found in Section 6.4.3 to Section 6.4.6.

4. *How long is the message latency for the individual mesh protocols?*

   The latency for message flooding evidently depends on the scanning duty cycle for both NSMP and SMP. Section 6.4.3 shows that the SMP adds around 200-400 ms per hop in a grid deployment (Section 5.5.1), whereas NSMP only adds around 100 ms per hop. In Section 6.4.5 these values are confirmed when the mesh arrangement is switched to a random deployment (Section 5.5.2). However, Section 6.4.3 to Section 6.4.6 show that the SMP has significantly lower power delay product compared to NSMP. Section 6.5 later displayed that routed communication with SMP had almost identical packet latency as flooded communication.

5. *Is there any support for multi-hop message routing in the individual mesh protocols?*

   It is demonstrated in Section 4.5 that the SMP supports an AODV-based routing algorithm that can be used to establish a route. Once a route has been established the SMP allows any application data to be exchanged between the two end-devices using the L2CAP. It is also shown that SMP supports a packet exchanging technique to check the aliveness of a route and, in addition, acknowledges the reception of application packets. The simulation results from Section 6.5.1 and Section 6.5.2 shows that the lifetime of a route strongly depends on the minimal accepted signal strength during route discovery. A high value of the minimal accepted signal strength makes the route very robust, but consequently requires more time to establish. The typical latency for establishing a route in the grid deployment (Section 5.5.1) is 1.5-3.5 seconds, whereas the latency in the random deployment (Section 5.5.2) is 1.0-3.0 seconds. It is further shown that the packet delivery ratio is practically 100 % when message routing is used. The latency for routed messages is principally the same as for flooded messages.

7.2 Future Work

This report has displayed various techniques that can be used to establish and maintain a BLE mesh network, and to increase its overall performance. It has also been presented that some of these techniques work very well in particular situations, while they in other situations fail. The underlying reasons are occasionally a bit diffuse and would therefore require deeper investigation. Additional work must also be put into the SMP specification since it ignores some fundamental problems for distributed networks, such as address distribution and a solution for self-adaptive message routing. Some suggestions of future work that closely relates to this thesis are given below.

- *Power improvements based on local collision ratio estimations*
  It has been shown that the proposed adaptation layer in SMP can detect whether the transmission power and the number of retransmissions is sufficient to reach neighbor devices. It has further been shown that a device can act upon these detections in order to improve the network performance. One way of taking this technique to its next level is to use local estimations of the collision ratio in order to adapt the size of the transmission window, and in this way save power when the traffic is low.
- *Investigation of improved routing algorithms for self-adapting networks*
  When the AODV-based routing algorithm in SMP was run simultaneous as the adaptation layer in Section 6.5.1 and Section 6.5.2, it was shown that they strongly interfered with each other. It would be useful to do a deep investigation of possible improvements to the routing algorithm to make it work better with self-adapting mesh nodes.
- *Design of an access profile for* BLE *mesh networks*
  To control the mesh layer of SMP would require some standard procedures that can be used from mesh applications. One adequate way of doing this is to introduce a mesh access profile that handles the configurations of the synchronization layer, adaptation layer, and routing layer of SMP. This profile could potentially act as a complement to the GAP, but for mesh network communication.
- *Address assignment in distributed mesh networks*
  It was promptly assumed in the thesis that each BLE device had a unique address already at the start of the simulation. In reality this address assignment must, of course, be taken care of by the mesh protocol. There are in fact a lot of techniques that can be used to assign unique addresses throughout a multi-hop network. Unfortunately most of them are based on the assumption that there is a central unit that assures address conflict-freeness. Further investigation is required in order to move this technique to a fully decentralized system.

Appendix

A

HCI Commands

This chapter presents some HCI commands that are relevant for this thesis. The HCI commands are further explained in the Bluetooth Core Specification [BLE, 2014b, pp. 962–1042].

A.1 HCI Advertising Control

This section presents some HCI commands used to control the advertising in the LL of a BLE device. The commands are presented in table A.1.

Table A.1: *Some important HCI advertising commands.*

Code	Command	Parameters
0x0006	LE Set Advertising Parameters	ADVERTISING_INTERVAL_MIN, ADVERTISING_INTERVAL_MAX, ADVERTISING_TYPE, OWN_ADDRESS_TYPE, PEER_ADDRESS_TYPE, PEER_ADDRESS, ADVERTISING_CHANNEL_MAP, ADVERTISING_FILTER_POLICY
Used by the host to set advertising parameters. ADVERTISING_INTERVAL_MIN and ADVERTISING_INTERVAL_MAX are the interval of the periodical advertising. ADVERTISING_TYPE must be one of the following values: ADV_IND (0x00), ADV_DIRECT_IND (0x01), ADV_SCAN_IND, high duty cycle (0x02), ADV_NONCONN_ID (0x03), ADV_DIRECT_IND, low duty cycle (0x04). ADVERTISING_INTERVAL_MIN must be greater or equal to 20ms, and greater or equal to 100ms if ADV_SCAN_IND or ADV_NON_CONN_IND is selected.		
0x0008	LE Set Advertising Data	ADVERTISING_DATA_LENGTH, ADVERTISING_DATA
Used by the host to set the data for the advertising packet. ADVERTISING_DATA_LENGTH must be within the range 0x00-0x1F. ADVERTISING_DATA must be formatted according to the Advertising Data (AD) format.		
0x000A	LE Set Advertise Enable	ADVERTISING_ENABLE
Used by the host to enable/disable advertising. ADVERTISING_ENABLE = 0x01 enables advertising, whereas ADVERTISING_ENABLE = 0x00 disables it.		

A.2 HCI Scanning Control

This section presents some HCI commands used to control the scanning in the LL of a BLE device. The commands are presented in table A.2.

Table A.2: *Some important HCI scanning commands.*

Code	Command	Parameters
0x000B	LE Set Scan Parameters	LE_SCAN_TYPE, LE_SCAN_INTERVAL, LE_SCAN_WINDOW, OWN_ADDRESS_TYPE, SCANNING_FILTER_POLICY
Used by the host to set scan parameters. LE_SCAN_TYPE must have one of the values: NO_SCAN_REQ (0x00) (passive scanning) or SCAN_REQ (0x01) (active scanning). LE_SCAN_INTERVAL is the interval of the engendered scanning windows. LE_SCAN_WINDOW is the length of the scanning windows.		
0x000C	LE Set Scan Enable	LE_SCAN_ENABLE, FILTER_DUPLICATES
Used by the host to enable/disable scanning. LE_SCAN_ENABLE = 0x01 enables scanning, whereas LE_SCAN_ENABLE = 0x00 disables it.		

B

Simulation Parameters

This chapter summarizes the parameters used for the BLE mesh simulator described in Chapter 5. The parameters are listed in Table B.1.

Table B.1: *A list of system parameters used in the simulator.*

Parameter	Value
Controller layer	
Transmission power, P_{TX}	7.5 dBm
Receiver sensitivity (high-gain mode), S_i	-93 dBm
Receiver sensitivity (standard mode), S_i	-86 dBm
NSMP layer (controller)	
Minimal delay, first transmission, $T_{1,min}$	0 ms
Maximal delay, first transmission, $T_{1,max}$	0 ms
Minimal delay, second transmission, $T_{2,min}$	95 ms
Maximal delay, second transmission, $T_{2,max}$	110 ms
Minimal delay, third transmission, $T_{3,min}$	0 ms
Maximal delay, third transmission, $T_{3,max}$	20 ms
NSMP layer (relay)	
Scanning interval, T_i	45 ms
Scanning duty cycle, D_{scan}	20 %
Minimal delay, first transmission, $T_{1,min}$	15 ms
Maximal delay, first transmission, $T_{1,max}$	35 ms
Minimal delay, second transmission, $T_{2,min}$	95 ms
Maximal delay, second transmission, $T_{2,max}$	110 ms
Minimal delay, third transmission, $T_{3,min}$	0 ms
Maximal delay, third transmission, $T_{3,max}$	20 ms
Synchronization layer	
Cycle time, T_{cycle}	150 ms
Transmission window, T_{TX}	30 ms
Sleep clock drifting, ρ	31-50ppm
Time beacon reciprocal rate, λ_{TB}	4
Adaptation layer	
Adaptation interval, T_{adapt}	5 s
Desired probability, p	99.9 %
Desired number of neighbors, u	2
Number of retransmissions, N	3
Routing layer	
Minimal accepted RSSI, $P_{th,route}$	-68 dBm
RREQ timeout, ξ_{RREQ}	5000 ms
RREP delay, T_{RREP}	1000 ms
Ping timeout factor, ξ_{PING}	3

List of Figures

List of Tables

Bibliography

Spike in Bluetooth Technology Penetration in Hub Devices Leads to Increased Bluetooth Smart Adoption. Bluetooth Special Interest Group, 4 2014. URL http://www.bluetooth.com/Pages/Press-Releases-Detail.aspx?ItemID=209. Accessed: 2015-08-24. Cited on page 1.

Ericsson. Ericsson mobility report. Technical report, 2014. Cited on page 1.

More Than 30 Billion Devices Will Wirelessly Connect to the Internet of Everything in 2020. ABI Research, 5 2013. URL https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne/. Accessed: 2015-08-24. Cited on page 1.

Skocir Branko. Multi-hop communication in bluetooth low energy ad hoc network. Master's thesis, Jozef Stefan International Postgraduate School, 2014. Cited on page 1.

K Mikhaylov. Multihop data transfer service for bluetooth low energy. Technical report, IEEE, 11 2013. Cited on page 1.

Kevin Townsend. *Getting Started with Bluetooth Low Energy*. O'Reilly Media, Inc, 2014. Cited on pages 2, 10, 11, 13, and 15.

Wireless mesh network concepts and best practices guide, 2010. Cited on page 2.

Andrews S. Tanenbaum. *Computer Networks*. Pearson Education, Inc., 2011. Cited on pages 2, 45, and 46.

Rohal Pankaj. Study and analysis of throughput, delay and packet delivery ratio in manet for topology based routing protocols (aodv, dsr and dsdv). *International Journal for Advance Research in Engineering and Technology*, 2013. Cited on page 2.

Sabih H. Gerez. Implementation of digital signal processing: Some background on gfsk modulation. Technical report, Department of Electrical Engineering, 2013. Cited on page 5.

Robin Heydon. *Bluetooth Low Energy - The Developer's Handbook*. Pearson Education, Inc, 2013. Cited on pages 5, 6, 8, 11, 14, 16, and 39.

Specification of the Bluetooth System, 4.2, volume 6. Bluetooth Special Interest Group, 2014a. Cited on pages 8, 9, 10, and 55.

Specification of the Bluetooth System, 4.2, volume 2. Bluetooth Special Interest Group, 2014b. Cited on pages 10, 18, 22, and 85.

Specification of the Bluetooth System, 4.2, volume 3. Bluetooth Special Interest Group, 2014c. Cited on pages 10, 12, 13, 15, and 16.

Bluetooth Low Energy Functions. Microsoft, 2015. URL `https://msdn.microsoft.com/en-us/library/windows/hardware/hh450825(v=vs.85).aspx`. Accessed: 2015-08-24. Cited on page 15.

BluetoothGatt. Google Inc., 4 2015. URL `https://developer.android.com/reference/android/bluetooth/BluetoothGatt.html`. Accessed: 2015-08-24. Cited on page 15.

Specification of the Bluetooth System, 4.2, volume 1. Bluetooth Special Interest Group, 2014d. Cited on page 16.

Amit Gupta. Heart rate profile. Technical report, Bluetooth Special Interest Group, 2011. Cited on page 16.

Specification of the Bluetooth System, 4.0, volume 6. Bluetooth Special Interest Group, 2010. Cited on page 16.

Specification of the Bluetooth System, 4.1, volume 6. Bluetooth Special Interest Group, 2013. Cited on page 16.

POSIX and Olson time zone formats. IBM China, 12 2012. URL `http://www.ibm.com/developerworks/aix/library/au-aix-posix/au-aix-posix-pdf.pdf`. Accessed: 2015-08-24. Cited on page 29.

Ulrich Windl. Clock quality, 11 2006. URL `http://www.ntp.org/ntpfaq/NTP-s-sw-clocks-quality.htm`. Accessed: 2015-08-24. Cited on page 38.

Low-power serial real-time clock (RTC). STMicroelectronics, 2 2013. URL `http://www.st.com/web/en/resource/technical/document/datasheet/CD00019849.pdf`. Accessed: 2015-08-24. Cited on page 39.

DS1302. Maxim Integrated Products, 3 2015. URL `http://datasheets.maximintegrated.com/en/ds/DS1302.pdf`. Accessed: 2015-08-24. Cited on page 39.

C Perkins. *Ad hoc On-Demand Distance Vector (AODV) Routing*. University of California, 2003. Cited on page 47.

Agustí Corbacho Salas. Indoor positioning system based on bluetooth low energy. Technical report, Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona, 2014. Cited on page 54.

Said Ghnimi. Ber perfermance of gmsk modulation under radio. Technical report, IEEE, 2011. Cited on page 54.

CC2540. Texas Instruments, 6 2013a. URL `http://www.ti.com/lit/gpn/cc2540`. Accessed: 2015-08-24. Cited on page 55.

CC2541. Texas Instruments, 6 2013b. URL `http://www.ti.com/lit/gpn/cc2541`. Accessed: 2015-08-24. Cited on page 55.