MASTER THESIS

TITLE: Performance evaluation of IPv6 over Sigfox: focusing on SCHC fragmentation

MASTER DEGREE: Master's degree in Applied Telecommunications and Engineering Management (MASTEAM)

AUTHOR: Antonios Platis

ADVISORS: Carles Gómez Montenegro, Rafael Vidal Ferré

DATE: March, 20 2021

Title: Performance evaluation of IPv6 over Sigfox**:** focusing on SCHC fragmentation

Author: Antonios Platis

Advisors: Carles Gómez Montenegro, Rafael Vidal Ferré

Date: March 20, 2021

Abstract

IoT is emerging nowadays, and new technologies are entering the market. There are various technology categories in IoT, one of which is Low-Power Wide Area Networks (LPWANs). It consists of technologies with a large range, relatively low data rate and lightweight payload.

A challenge that those technologies are facing is the support of the IPv6 protocol, in order to be able to support Internet connectivity. The IETF has recently defined Static Context Header Compression and fragmentation (SCHC), an adaptation layer defined by the IETF to support IPv6 over LPWAN technologies.

However, as SCHC is a new protocol, its performance has not yet been evaluated. The purpose of this Master Thesis is to evaluate the performance of IPv6 over Sigfox, a LPWAN technology that is growing over the last years, through SCHC. In particular, this Master Thesis is on SCHC fragmentation performance over Sigfox.

A testbed has been created for that evaluation, using a device that supports Sigfox and a cloud server. Several performance metrics have been chosen, like the transmission duration of a packet (which is evaluated both theoretically and experimentally), the amount of uplink and downlink messages exchanged, and the energy performance.

The findings of the evaluations show that sending larger packets through Sigfox network using SCHC fragmentation is feasible and the transmission duration may vary from some seconds, up to hours, depending on the size of the packet and the transmission errors. Finally, a device's lifetime when transmitting SCHC Packets may range from 2 months, up to more than 4 years, depending on the SCHC Packet size and the period T of transmitting the packet.

Acknowledgement

First of all, I would like to thank my professors and advisors in this Master Thesis, Carles Gómez Montenegro and Rafael Vidal Ferré, who guided me from the beginning until the end of the project. In addition, I would like to thank Sergio Aguilar Romero, who was always available to help me, and everyone else involved in the SCHC over Sigfox colaboration. Finally, my fellow colleagues and friends of the Master, my family, and everyone else that was by my side while working on the Master Thesis, especially during that tough time of Covid-19.

CONTENTS

Introduction

The Internet recently experiences a tremendous evolution and is part of the daily life of everyone. The amount of traffic in the Internet is in the scale of Exabytes / month and will continue to increase for the next years.

The Internet of Things (IoT) is a term very popular today. Its meaning is that not only personal computers and laptops are connected to the Internet, but also objects, like lamps, alarms, temperature and humidity sensors and many more can be connected and do not require any human interaction.

There are various IoT networks today and as this sector grows, more will enter the market. However, there are several issues that need to be addressed and considered when thinking about the IoT technologies and their connectivity to the Internet. More specifically, regarding the wireless technologies, the more popular ones today are the ones that are part of the Wireless Personal Area Networks (WPANs) and the Low Power Wide Area Networks (LPWANs).

However, the Internet was not initially designed to support such devices. One of their main characteristics is low consumption and small amount of data sent. That would mean that transmitting a larger amount of data, as required to support IPv6, is not always possible, in principle. During the last years, special adaptation layers were proposed to support larger packets through networks with restrictions in data size transmission, like WPANs or LPWANs. More specifically, one of them is called Static Context Header Compression and fragmentation (SCHC), which was developed to support header compression and fragmentation for LPWANs.

The performance of the SCHC protocol, and in particular, its fragmentation functionality, over Sigfox network is the main topic of this Master Thesis. More in detail, metrics like packet transmission duration, uplink and downlink Sigfox messages sent, average current consumption and device lifetime are evaluated.

The present Master Thesis is divided in 5 chapters. The first one is an overview of LPWAN technologies. The second one discusses the topic of IPv6 over IoT technologies, and the adaptation layers needed to support IPv6. The third one is focused on the evaluation of the transmission durations for different packet sizes. The fourth one presents an energy consumption evaluation of SCHC over Sigfox. Finally, the final chapter presents the conclusions of the evaluation.

1. IoT Overview

Despite the fact that IoT as a term is a global trend nowadays, it was mentioned for the first time in 1999 by Kevin Ashton who saw the potential of an object being connected to the Internet. Going even more back in time, networks that included communications between machines, the so-called M2M communications, can be found even in the 1970's, where there were systems to monitor meters at the electrical grid. In the 1990's, there were a lot of systems and networks that included M2M communications, but were more for industrial use and application-specific, thus not being connected to the Internet through IP.

In the most recent days, there is a big spectrum of technologies having a focus on M2M communication. Some of them already support internet connectivity and for some others it is an ongoing effort. The majority of the use cases for those technologies are "smart" applications like: home automation, building automation, smart cities, smart grid, smart agriculture, smart factory, smart health, smart supply chain and many others.

Regarding the categorisation of those technologies, and more specifically the wireless ones, some characteristics that distinguishes them is the range of radio links, the energy consumption, the message sizes and required data rates, transmission latency, etc. The two main categories of wireless IoT technologies are Wireless Personal Area Networks (WPAN) and Low Power Wide Area Networks (LPWAN). In this chapter, LPWAN technologies are presented, with a focus on the Sigfox technology, which is the main technology of this Master Thesis. For a state-of-the-art analysis of current WPAN technologies, see Annex 1.

1.1 LPWAN technologies

The LPWAN main characteristic and difference compared to other categories of wireless networks is its long coverage and transmission range. In Figure 1.1 we can observe a qualitative comparison of LPWAN with other IoT wireless technologies:

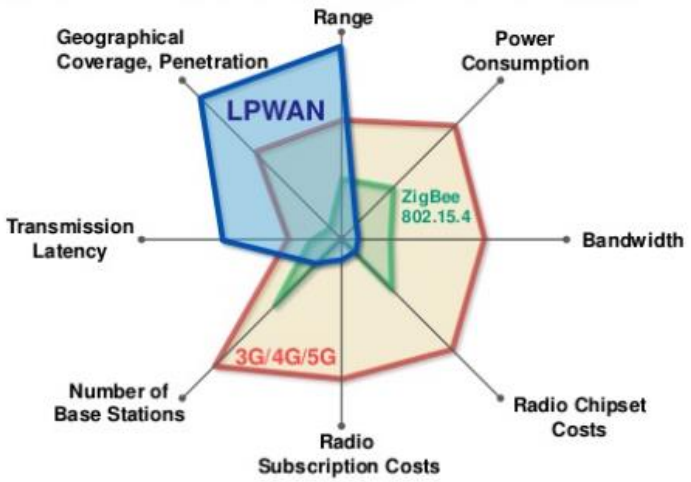

Figure 1.1: Qualitative comparison of IoT technologies [1]

Apart from the above mentioned characteristics, it is important to mention that LPWANs are very low cost technologies and they have extremely low power consumption. These characteristics make them ideal for IoT application because considering an application of a big quantity of sensors in a large geographical area would for sure require:

- low deployment costs, in order to allow the quantity needed for the application,
- low power consumption, in order to allow the devices to have a long lifetime and
- long range, in order to take measurements even in difficult to access places, and reduce the amount of network infrastructure required.

The drawback that LPWAN have is their relatively big transmission latency and small bandwidth. This results in data rates smaller than other IoT technolgies, such as WPAN ones. However, as there are applications where devices only need to send a few data per day, the data rates offered are enough to cover the needs in those cases.

The three main LPWAN technologies are Sigfox, LoRa/LoRaWAN and NB-IoT. Figure 1.2 presents a qualitative comparison between them:

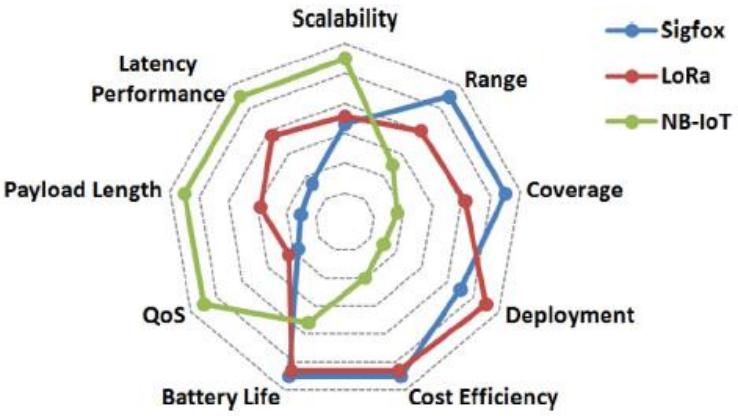

Figure 1.2: Comparison of LPWAN Technologies [2]

As seen in Figure 1.2 and expained in [2], Sigfox provides the better coverage and transmission range, which is basically because of having its own infrastructure network, while also having the longest battery life and cost efficiency, together with LoRa. In terms of latency performance and QoS, NB-IoT performs in a better way, as it uses licensed bands and higher bit rate. It also supports a much larger payload length than the others and a much larger number of devices, making it the most scalable. Finally, LoRa and Sigfox are more mature technologies, and thus being deployed in more countries, being around 100 countries for LoRa and 72 countries for Sigfox.

In the following sections, these technologies will be discussed further and especially Sigfox, which is the main technology that this Master Thesis focuses on.

1.1.1 LoRa

LoRa or LoRaWAN is a low rate and long range wireless communication network. It operates on the Industrial, Scientific and Medical (ISM) bands, having different frequency bands depending on the region.

As the autors of [3] mention, LoRa defined three classes of devices, which are called end-devices, all of which support bidirectional communication:

- Class A, which are battery powered sensors
- Class B, which are battery powered actuators, and have some additional listening windows compared to Class A,
- Class C, which are main powered actuators, and have even more listening windows.

Regarding the protocol stack, LoRaWAN defines the network elements, while LoRa physical layer defines the radio links of the end devices to the network. Figure 1.3 shows LoRaWAN protocol stack:

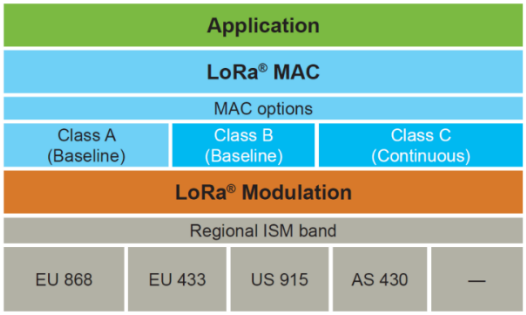

Figure 1.3: LoRaWAN protocol stack [3]

LoRa has a star-of-stars network topology. This means that between the end-devices and the radio gateways, also known as concentrators, the topology is star. In addition, between the concentrators and the network server, a link which is IP-based, the topology is also star. Finally, the network server is also connected to application servers, where the data is being stored or processed. We can have a more visual representation of the aforementioned in Figure 1.4:

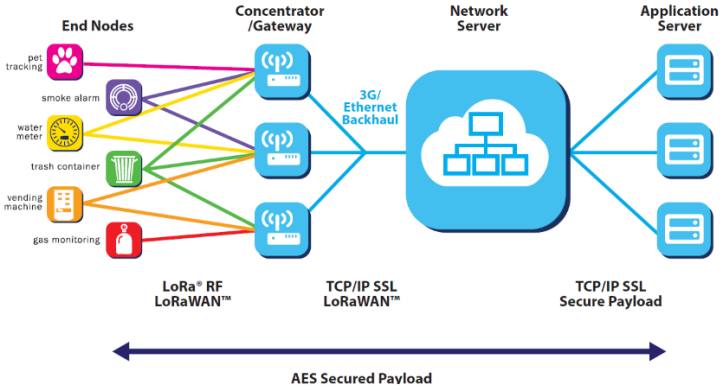

Figure 1.4: Network topology of LoRa [3]

As mentioned before, the different classes of devices have different receive settings. For example, in the case of class A devices, there are two listening windows after the transmission. As a result, the device can receive downlink messages during those listening windows. The delay between the sending window and the listening one is a setting defined for different regions because it should follow the local regulations.

In the physical layer, LoRa works at the sub-1 GHz layer, depending on the region, e.g. 433 and 868 in Europe, 915 in North America, etc. As explained in [4], the modulation used is called chirp spread spectrum (CSS). In CSS, bits of payload are represented as chirps. The ratio between the symbol rate and the chirp rate is called spreading factor (SF) and can range from 7 to 12, 7 being the one with the higher data rate. Additionally, a higher SF means more airtime or transmission time. In Figure 1.5 below there is a visual representation of the relation between the SF, the bitrate and the airtime, thus energy consumption, of a LoRa communication.

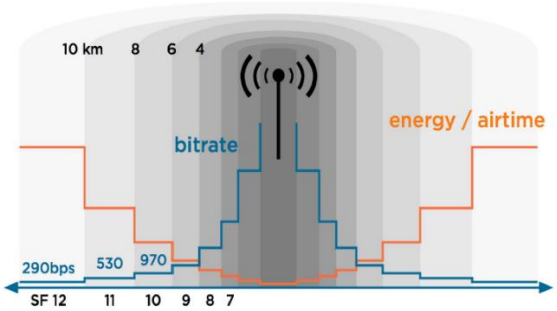

Figure 1.5: Bitrate and airtime for different SF on LoRa [4]

The data rates achieved can start from 270 bps and can go up to 50 kbps, when operating in Europe's frequency bands.

1.1.2 NB-IoT

NB-IoT is another LPWAN technology. It is specified by 3GPP and the original specification was in 2016. The main difference between NB-IoT and the rest of LPWAN technologies is that it operates on licensed bands. More specifically, it coexists with LTE and GSM as they are all specified by 3GPP.

A NB-IoT communication has a bandwidth of 200 kHz. There are three different methods of operation for the NB-IoT communications. The first one is called stand alone, where the current GSM frequency bands will be used. The second one is the guard band operation, where the unused resource blocks of the LTE guard band will be used. The last one is the in-band operation, where resource blocks of the LTE communications will be used. In Figure 1.6 below those operations are shown:

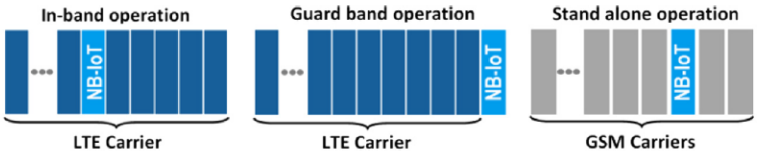

Figure 1.6: Operation modes of NB-IoT [2]

As mentioned above, NB-IoT is part of 3GPP standard, and more specifically of the LTE standard. However, it has several differences. As it is an IoT technology, it does not implement energy consuming mechanisms such as dual connectivity, handover, channel quality and others. It makes use of the Quadrature Phase Shift Keying (QPSK) modulation with Orthogonal frequency-division multiplexing (OFDM) or Single-carrier frequency-division mutiple access (SC-FDMA). The data rates achieved are 235 kbps in the Downlink (DL) and 205 kbps in the Uplink (UL).

In terms of network topology, as explained in [5], NB-IoT uses the one of LTE, known as Evolved Packet System (EPS), while having some optimizations called cellular IoT. These optimizations are responsible for finding the best path for the data transmitted. The protocol stack used is also the one of LTE reduced to the minimum requirements. Figure 1.7 shows a visual representation of the protocol stack of NB-IoT, where the UE is the device, eNB is the LTE receiver and MME is the mobility management unit.

Due to the fact that NB-IoT makes use of the current infrastructure of LTE and GSM, it can be easily deployed. It is though limited to the area supported by the LTE and GSM operators. Today, it is deployed in 52 countries and continues to grow.

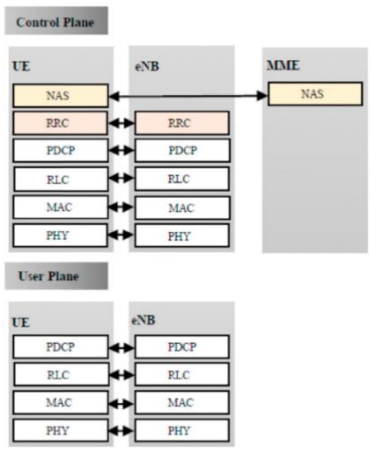

Figure 1.7: NB-IoT Protocol stack for control and user plane [5]

1.1.3 Sigfox

As explained before, Sigfox is the main LPWAN technology this Master Thesis is based on and thus will be explained with more details. It is another LPWAN technology like the ones described previously, having main characteristics as low power consumption and long range. It is already deployed in 72 countries and continues expanding. Figure 1.8 shows some important numbers regarding the the scale of Sigfox's deployment:

Figure 1.8: Sigfox scale [6]

Sigfox was founded in 2010 in France by Ludovic Le Moan and Christophe Fourtet. The vision was to be able to connect the objects of the physical world to the digital one. After 10 years, Sigfox has the largest IoT ecosystem and has deployed its own network infrastructure.

Sigfox also operates in the ISM bands. More specifically, as explained in [6], it has defined eight different radio configuration zones (RCZ) for the different regions of the world. For example, RCZ1 is used in Europe and operates in the 868 MHz band.

In order to achieve the best performance at the physical layer, Sigfox's network is called 3D-UNB, which stands for 3-dimensional ultra narrow band. First of all, Sigfox communications are ultra narrow band, which means that they have a very small bandwidth, 100 Hz. The great advantage of having a very narrow bandwidth is that it is very robust to interference, as it covers a very small part of the spectrum.

The modulation used for uplink messages is differential binary phase shift keying (D-BPSK). Sigfox uses this modulation because it is easy to implement and results in low cost components, while there is no need for a very high data rate. The data rates achieved are 100 bps or 600 bps, depending on the region. For example, in RC1, both 100 and 600 bps are used. For the downlink, Sigfox uses Gaussian phase shift keying (GFSK) having a bitrate of 600 bps.

Getting back to the 3D-UNB, the 3 dimensions are time, space and frequency. The combination of the three dimensions contributes to having a more robust network which has as few communication errors as possible. Each device that uses the Sigfox network sends a frame three times. This provides the time diversity of the Sigfox network. Duplicating and sending again the message

decreases the chances of an error in the receiving end. In addition to that, each transmission is done in a different frequency, which provides the frequency diversity. This takes place to prevent a noisy frequency channel, by hopping on nearby ones. This technique is generally called frequency hopping (FH). Finally, Sigfox messages are not transmitted to a specific receiving base station. The devices are not associated with one base station. Alternatively, all base stations that are within the range of the sending device will receive the message sent. This also provides an extra layer of error prevention and explains the space diversity. Using the mechanisms described in this section, Sigfox results in a very robust and error-prone network.

Regarding the communication types, Sigfox has defined two different such types, called U-procedure, which stands for unidirectional procedure, and B-procedure, which stands for bidirectional procedure. In the first one, the device only sends data towards the Sigfox network, while in the second one, it transmits and then waits to receive a downlink message. There is no other way that a Sigfox device can receive a message. Figure 1.9 shows the sequence of messages sent in a Sigfox B-procedure. Note that three replicas of the same message are sent in a different freqquency, followed by the DL message.

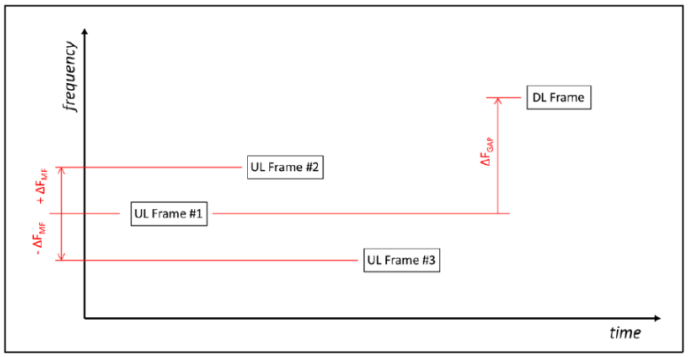

Figure 1.9: Sigfox B-procedure with multiple frames and FH [6]

Regarding the frame sizes, Sigfox supports message payloads of up to 12 bytes in the UL and exactly 8 bytes in the DL. It is obvious that these messages are small compared to those of other wireless communication technologies. However, devices using the Sigfox network are of low consumption and do not need to process and send big amounts of data. As a result, these lightweight messages are indeed serving their purposes. For an UL message of 12 bytes of payload, the total Sigfox frame size will be 26 bytes. Figure 1.10 shows the full communication stack from application to modulation in the UL:

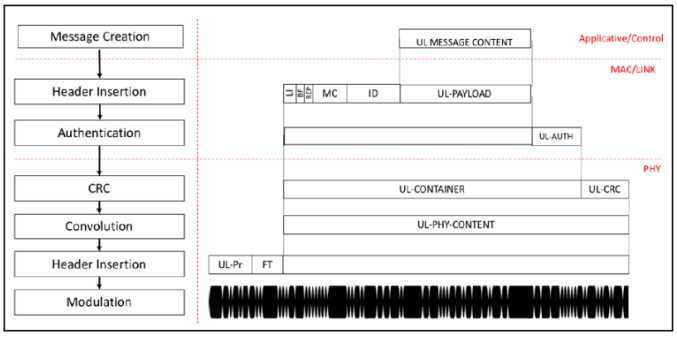

Figure 1.10: Sigfox communication stack in UL [6]

Observing Figure 1.10 we can see the sequence of mechanisms implemented, both in the MAC and the PHY layer, after the creation of the message. Each mechanism inserts additional bits for its respective purpose or processes the frame until the modulation translates the message to the RF domain.

As mentioned above, Sigfox has created a proprietary network infrastructure. This infrastructure is composed of a huge amount of base stations in the 72 countries where the network is implemented. All those base stations are receiving the uplink messages of the devices, as well as sending back downlink data when needed, through the Sigfox radio as explained in the previous sections. Then, they establish IP links with the Service Center, which is the backend cloud-based network of Sigfox. There, all the data is concentrated and forwarded to a user application or server, if requested. Finally, another entity in the Sigfox network is the Registration Authority. This entity is responsible for user authentication, making sure that no unauthorised devices have access. Figure 1.11 shows a visual representation of the Sigfox network:

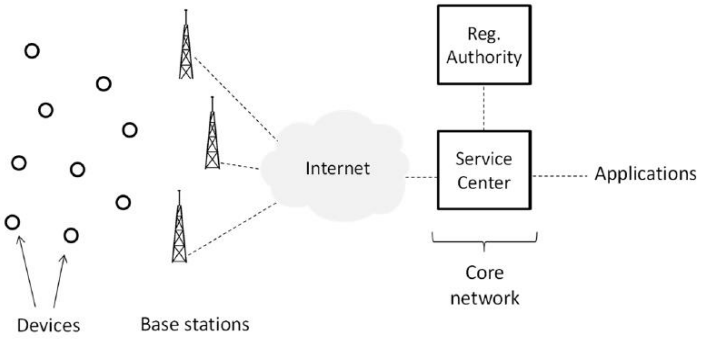

Figure 1.11: Sigfox network architecture [7]

Regarding the user applications and servers, the connection between these and the core network of Sigfox is done through callbacks. Callbacks are APIs configured by the users that can send their data arriving in the core network to their servers. Sigfox provides guidelines on how to use these callbacks specifically with Amazon AWS or Google Cloud Platform. More details on Sigfox callbacks can be found on [8].

When a device is new to the Sigfox network, it needs to be registered in order to have the required access rights and be able to transmit data to the base stations. This is done in the Sigfox backend. Through its user interface, apart from registering devices, the user can check the Sigfox service map, keep track of all the messages sent to his devices, manage device types and manage callbacks.

2. IPv6 over IoT

One of the main concepts of IoT is connectivity among devices. This connectivity can be achieved using the networks mentioned in the previous chapter, like the ones of Sigfox or LoRa. However, the need for a universal network is not satisfied using those dedicated networks. Therefore, at that point comes the need for connecting those networks to the Internet. This can be done through the IPv6 protocol, which is the most recent version of the Internet Protocol (IP), a protocol defined by the Internet Engineering Task Force (IETF). One of the most important differences of IPv6, which make it more suitable for communications between small and constrained devices, is that it has an address space of 2^{128} addresses, enabling the use of almost unlimited devices in the Internet. In other words, every small device, from a sensor, to an air conditioner or a fridge, could have its own IPv6 address and handle its own messages. However, networks like the ones presented in the previous chapter, like Sigfox and LoRa, were not initially designed to handle IPv6 packets. On the other hand, LPWANs support smaller message payloads, as they target to be lightweight. As a result, there has to be an adaptation layer between the IPv6 layer and the lower levels (MAC, PHY) of those technologies. Those adaptation layers will take care of the fragmentation and the reassembly of larger packets, if it is not supported by the low level technology itself. In addition, they provide other mechanisms, such as neighbour discovery, header compression and more. In the following sections, adaptation layers will be discussed in further detail. 6LoWPAN and 6Lo, which are some of the most common adaptation layers in IoT, are presented in Annex 2. In the next section, the main adaptation layer examined in the Master Thesis, SCHC, is presented.

2.1 Static Context Header Compression

Static Context Header Compression (SCHC) is a generic framework defined by the IETF in [9], which provides a header compression/decompression mechanism and an optional fragmentation/reassembly mechanism. It is designed to be used by LPWANs, like Sigfox, LoRa and NB-IoT. As in the case of 6LoWPAN, SCHC is an adaptation layer, which lies between the IPv6 layer and the lower layers of LPWAN technologies. This is illustrated in Figure 2.1:

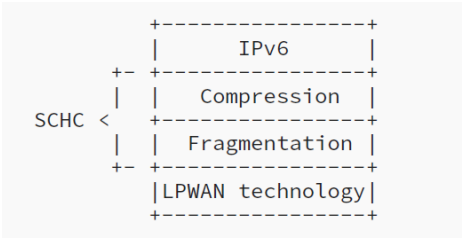

Figure 2.1: Protocol stack of SCHC used in a LPWAN technology [9]

The information described is based on the framework definition in [9]. In this section, the header compression mechanism will be analysed, while fragmentation, which is the main mechanism evaluated in the Master Thesis, will be discussed in a separate section.

When transmitting a packet, the compression will be done firstly, followed by the fragmentation. In the receiver end, the packet reassembly will be done, followed by the decompression, as shown in Figure 2.2:

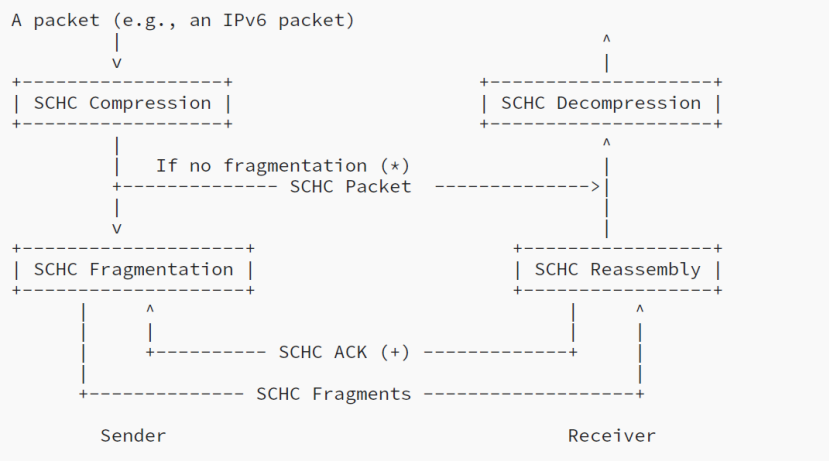

Figure 2.2: SCHC mechanism sequence in a packet transmission [9]

A SCHC Packet is the data unit produced by the SCHC compression. It consists of the payload and the compressed header of an IPv6 packet. The latter is composed of a RuleID and a compression residue.

There are several parameters that SCHC defines. Some of them are whether fragmentation will be used, or which will be the size of the RuleID, how many Rules there will be, the maximum packet size, the size of the Layer 2 (L2) word size and how will padding be handled (‘1’ or ‘0’). This set of parameters is defined in a profile, which should be shared among both the device and the network infrastructure, no matter which the sender and receiver are.

In addition, a set of Rules, used either in a compression/ decompression (C/D) process, or in a fragmentation/reassembly (F/R) process, define a Context. The scope of a Context is device specific. The use of a RuleID in C/D is to identify the compression’s rule, while in F/R, its role is to identify all the settings of fragmentation used. The RuleID in F/R is also divided in uplink and downlink.

In SCHC compression/ decompression, the RuleID is sent as part of the compressed header. The purpose of sending only a RuleID, instead of all the parameters needed for the C/D, is to save space from the packet length, which

makes C/D more efficient. The sender and the receiver are then able to associate the specific RuleID with the parameters of the Rule.

In Figure 2.3, the different fields of a Rule inside a Context are illustrated:

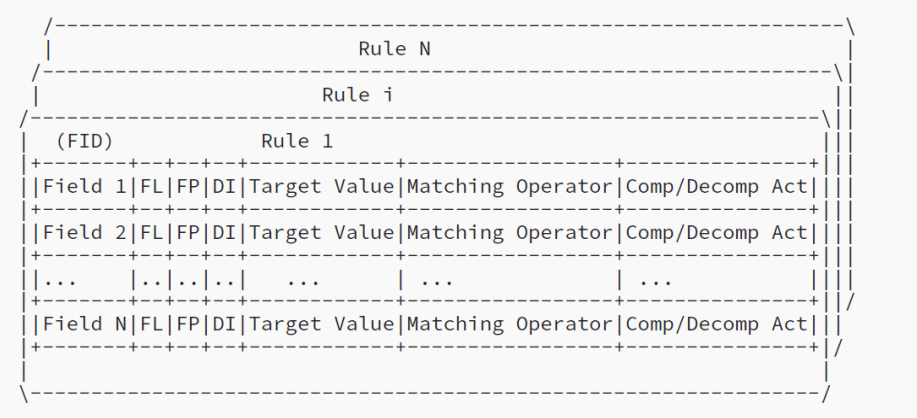

Figure 2.3: A SCHC C/D Context [9]

There are several fields, that are described below:

- Field Identifier (FID) defines a protocol and a field.
- Field Length (FL) defines the length of the original field.
- Field Position (FP) defines the position of the first occurrence of a field in a packet.
- Direction Indicator (DI) defines the direction (uplink, downlink, bidirectional).
- Target Value (TV) is the value used to match with the packet header field.
- Matching Operator (MO) is the operator which will be used to match the target value with the field value.
- Compression/ Decompression Action (CDA) is the pair of actions used to compress and decompress the header field.

As mentioned above, the result of a header compression is a SCHC Packet, which can be divided in the fields shown in Figure 2.4:

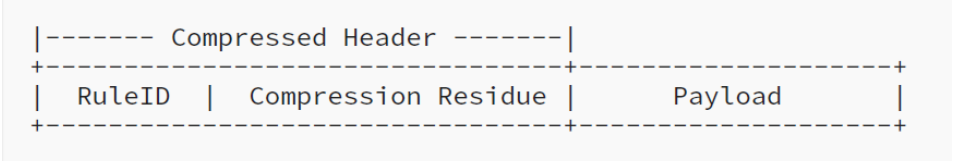

Figure 2.4: SCHC Packet [9]

The compressed header consists of the RuleID chosen and the compression residue as a result of the compression, if any. If there are more than one headers to be compressed, their respective residues are concatenated in the compression residue. Regarding the CDA, the Table 2.1 shows the actions that

are taken by the sender and the receiver, when each of the actions denoted is chosen.

Table 2.1. SCHC Header Compression CDAs

Action	Compression	Decompression
not-sent	elided	use TV stored in Rule
value-sent	send	use received value
mapping-sent	send index	retrieve value from TV list
LSB	send least significant bits (LSB)	concatenate TV and received value
compute-*	elided	recompute at decompressor
DevIID	elided	build IID from L2 Dev addr
AppIID	elided	build IID from L2 App addr

As seen in Table 2.1, sometimes, there are no actions taken from the compression part, and the decompressor constructs the field value from a computation, or from an address, like the AppIID or the DevIID.

2.2 SCHC fragmentation

SCHC fragmentation is the mechanism which fragments a large packet, e.g. an IPv6 packet of 1280 that does not fit in a L2 payload. A typical LPWAN technology has an L2 MTU of some tens of bytes. For example, Sigfox has a maximum uplink payload size of 12 bytes, and a maximum downlink payload size of 8 bytes. Some of those technologies, like Sigfox, do not have an internal fragmentation and reassembly mechanism, so SCHC F/R should be used in order to be able to send or receive larger packets.

There are currently three different modes of sending fragmented SCHC Packets: No-ACK, ACK-Always and ACK-on-Error. These modes will be discussed separately in the next subsections.

First of all, some of the most important parameters of F/R will be introduced. SCHC F/R defines several different messages. A message is a L2 transmission or reception. These messages are:

- SCHC Fragment: A message that carries part of the fragmentation result of a SCHC Packet.
- SCHC ACK: An acknowledgement sent by the receiver to the sender. It indicates a correct or wrong reception.
- SCHC ACK REQ: A message from the sender requesting a SCHC ACK
- SCHC Sender-Abort: A message from the sender indicating that the SCHC Packet sending is aborted.
- SCHC Receiver-Abort: A message from the receiver indicating the the SCHC Packet reception is aborted.

A SCHC Packet is fragmented in tiles. A SCHC Packet may contain one tile, but may also contain more than one tile. Tiles can be grouped in windows. All windows must have the same number of tiles, except the last one, which may have less. The tiles within a window are numbered in a decreasing way, starting from window size minus 1 up to 0. For example, for a window size of 7, tiles within a window should be numbered from 6 down to 0. Windows are used in the ACK-Always and ACK-on-Error modes.

Another important concept is the one of bitmaps. Bitmaps are bound to a specific window. They indicate whether a tile in a window is correctly received or not. For example, if bit 2 of a window is set to 1, it means that tile 2 of that window is correctly received.

There are also two important timers and a counter that are used by the different modes:

- The inactivity timer, that is used from the receiver. When this timer is over, the receiver aborts reception.
- The retransmission timer, which is used by the sender to abort waiting for a SCHC ACK.
- The attempts counter, which is counting how many ACKs have been requested and has a maximum of *MAX_ACK_REQUESTS*.

When all tiles are received on the reception side, the receiver should do an integrity check to make sure that the reception of a SCHC Packet was done successfully. The recommended way for doing an integrity check is using a Reassembly Check Sequence (RCS).

A SCHC message is composed of a SCHC Header and the rest of the message, which could be payload, bitmap, or padding, depending on the type of the message. The SCHC Header also has several fields depending on the message type. The header fields used are:

- RuleID: Specifies the RuleID used in the F/R.
- DTag: Differentiates SCHC Packets using the same RuleID.
- Window: Indicates the number of window.
- Fragment Compressed Number (FCN): Indicates the index of the tile within a window. For all windows, except the last one, FCN starts from *(WINDOW_SIZE - 1)* and goes until 0. For the last window, FCN starts from *(WINDOW_SIZE - 1)* and is *WINDOW_SIZE* for the last tile. The SCHC Fragment holding the last tile of the last window is called All-1 Fragment, while the SCHC fragments holding the last tile of the previous windows are called All-0 Fragments. The rest of the fragments are called Regular Fragments.
- RCS: Appears only in All-1 Fragments.
- C: This field indicates whether the integrity check was successful or not
- Compressed bitmap: This field is present optimally in SCHC ACK messages and shows the compressed bitmap.

Table 2.2 indicates which fields are present in each message type:

Table 2.2. SCHC Header fields present in different message types

Message Type	Header Field							
	Name	RuleID	DTag	Window	FCN	RCS	C	Compressed Bitmap
	Size	Defined by Profile	T bits	M bits	N bits	U bits	1 bit	Defined by Profile
SCHC Fragment		✓	✓	✓	✓	✓		
SCHC ACK		✓	✓	✓			✓	✓
SCHC ACK REQ		✓	✓	✓	✓			
SCHC Sender-Abort		✓	✓	✓	✓			
SCHC Receiver-Abort		✓	✓	✓			✓	

As mentioned previously, there are three types of SCHC Fragments: Regular, All-0 and All-1. In a SCHC Fragment message, the fragment payload follows the header, while padding is added if needed, as shown in Figure 2.5:

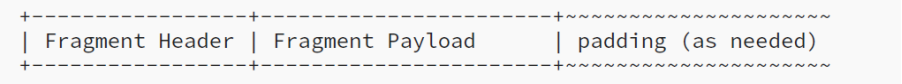

Figure 2.5: Generic format of SCHC Fragment [9]

In a more specific format, a Regular (as well as All-0) and All-1 Fragment is divided into the following parts shown in Figures 2.6 and 2.7:

Figure 2.6: Regular & All-0 SCHC Fragment format [9]

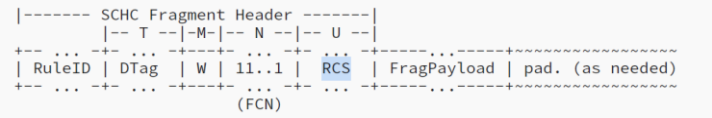

Figure 2.7: All-1 SCHC Fragment format [9]

All-1 Fragments are called so because the FCN field is filled with ones. The RCS field is also present.

An example of SCHC ACK messages formats, in both successful and unsuccessful reception, is shown in Figures 2.8 and 2.9:

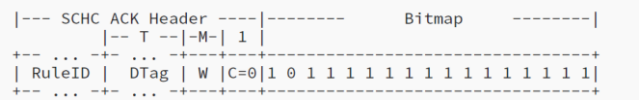

Figure 2.8: Unsuccessful SCHC ACK format [9]

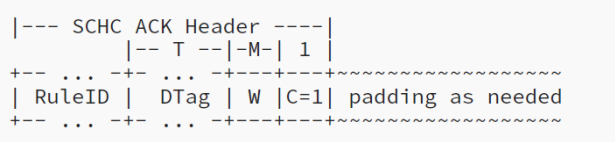

Figure 2.9: Successful SCHC ACK format [9]

Lastly, the formats of SCHC ACK REQ, Sender-Abort and Receiver-Abort are shown in Figures 2.10-2.12:

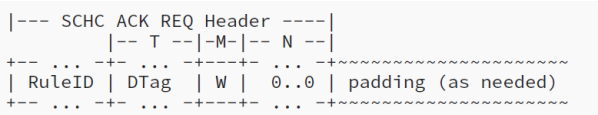

Figure 2.10: SCHC ACK REQ format [9]

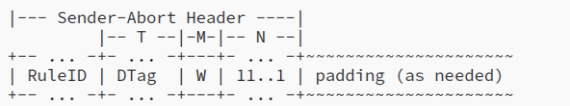

Figure 2.11: SCHC Sender-Abort format [9]

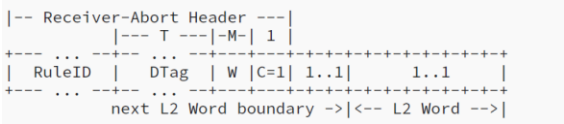

Figure 2.12: SCHC Receiver-Abort format [9]

2.2.1 No-ACK Mode

The first mode under examination is No-ACK. In this mode, SCHC ACK messages are not used. Therefore, this mode does not provide reliability. The most important characteristics of that mode are:

- It is mainly used in scenarios that do not require bi-directional links. Unidirectional links could work fine.
- Windows are not used.
- Each fragment should have one tile.

The sender transmits the fragments one by one, without waiting for an ACK. All fragments are Regular Fragments, except the last one which is an All-1. The receiver receives all fragments, until the All-1. After that, the receiver performs the final integrity check.

2.2.2 ACK-Always Mode

In this mode, SCHC ACKs are sent from the receiver. The main characteristics are:

- It requires bi-directional links.
- Windows are used.
- Each fragment should have one tile.

In this mode, the sender starts to transmit the fragments of the first window. After sending the last fragment, it waits for the reception of a SCHC ACK. If the ACK specifies that all fragments were correctly received, the sender proceeds to the next window. In any other case, it resends the unsuccessfully sent fragments. This would mean that the sender will not proceed to the next window until the whole previous window was correctly received. The same applies to the receiver. After the reception of the last fragment, which would be the All-1 Fragment, the receiver performs the integrity check, that will indicate a successful or unsuccessful reception of the SCHC Packet. If the reception was correct, it will send the final SCHC ACK to the sender. Otherwise, the transmission will continue until the successful reception of all fragments.

2.2.3 ACK-on-Error Mode

The last mode is ACK-on-Error. In that mode, SCHC ACKs are sent from the receiver, but not always. The more important characteristics are:

- It requires bi-directional links.
- Windows are used.
- Each fragment may have more than one tiles.
- All tiles, except the ultimate and the penultimate, should have the same size, called regular size.
- A SCHC Fragment may carry more than one contiguous tile.

An example of a SCHC Packet composition in tiles, windows and SCHC Fragment is shown in Figure 2.13:

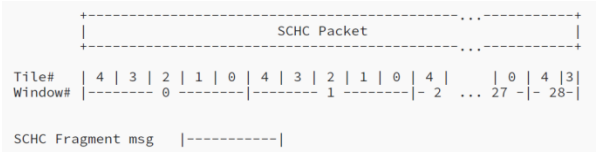

Figure 2.13: Composition of a SCHC Packet in ACK-on-Error [9]

In ACK-on-Error, the sender initially transmits all fragments of the first window. The receiver, upon receiving an All-0 Fragment, which corresponds to the last fragment of the window (except for the last window), checks the bitmap of that window. If all fragments are correctly received, it does not send the SCHC ACK message. In any other case, it sends the SCHC ACK message which indicates in the bitmap the unsuccessfully sent fragments. The sender, on the other side, upon receiving the SCHC ACK, will resend the requested fragments. If there is no ACK received, it proceeds to the next window, assuming a correct reception.

Another important characteristic is that after resending the fragments, the sender proceeds to the next window, without knowing if the resending was successful. That will get clear after the next reception of a SCHC ACK. That is why the W field of the SCHC ACK is very important. The receiver will send a SCHC ACK, having a W field of the first window that has an error, even though there may be more errors in the next windows. That is the one of the core differences with ACK-Always, where a window should be successfully sent before proceeding to the next one.

Finally, upon reception of an All-1 message, the receiver is first checking if there are any errors and if not, performs the integrity check. If the check is successful, it sends the final SCHC ACK to the sender and the SCHC Packet reception is successful. Otherwise, the transmission continues by sending the SCHC ACK with the bitmap and waits for the resending of fragments, until all fragments are correctly received and the integrity check is performed successfully.

As specified in [10], when ACK-on-Error is used over Sigfox, there are two recommended categories:

1. SCHC Packets whose size is less than 300 bytes.
2. SCHC Packets whose size is greater or equal to 300 bytes.

2.2.3.1 ACK-on-Error SCHC Packet Category 1

For packets smaller than 300 bytes, a single byte SCHC Header is used. The following header fields are used:

- Rule ID size: 3 bits
- DTag size (T): 0 bits
- Window index size (W): 2 bits
- FCN size (N): 3 bits
- MAX_ACK_REQUESTS: 5
- WINDOW_SIZE: 7 fragments
- Regular tile size: 11 bytes
- Penultimate tile size: 11 bytes
- Fragment size: 1 tile (Payload) plus header
- Retransmission Timer: 45 seconds
- Inactivity Timer: 200 seconds
- RCS: not used

In addition, as already mentioned, there are three different fragment types: Regular, All-0 and All-1. The size of those fragments in that category would be as shown in Table 2.3:

Table 2.3. SCHC Packet Category 1 fragment size per type

Fragment type	Header size	Payload size	Total size
Regular	1 byte	11 bytes	12 bytes
All-0	1 byte	11 bytes	12 bytes
All-1	1 byte	1-11 bytes	2-12 bytes

2.2.3.2 ACK-on-Error SCHC Packet Category 2

For packets larger or equal to 300 bytes, a two-byte SCHC Header is used. The following header fields are used:

- Rule ID size: 8 bits
- DTag size (T): 0 bits
- Window index size (W): 3 bits
- FCN size (N): 5 bits
- MAX_ACK_REQUESTS: 5
- WINDOW_SIZE: 31 fragments
- Regular tile size: 10 bytes
- Penultimate tile size: 10 bytes
- Fragment size: 1 tile (Payload) plus header
- Retransmission Timer: 45 seconds
- Inactivity Timer: 200 seconds
- RCS: not used

Same as above, the respective fragment size per type would be as shown in Table 2.4:

Table 2.4. SCHC Packet Category 2 fragment size per type

Fragment type	Header size	Payload size	Total size
Regular	2 bytes	10 bytes	12 bytes
All-0	2 bytes	10 bytes	12 bytes
All-1	2 bytes	1-10 bytes	3-12 bytes

3. Evaluation of SCHC Packet transmission duration

This chapter is devoted to the analysis of the transmission durations of sending SCHC Packets. First of all, it focuses on errorless transmissions, which means that all fragments are transmitted and received successfully. In the first subsection, a theoretical model for the errorless transmission duration is presented. In the second, the transmission durations are measured experimentally and compared to the theoretical values. In the third section, controlled errors in transmissions are evaluated experimentally, while in the last section, random errors in transmissions are also evaluated.

3.1 Theoretical Model for errorless transmissions

This section will focus on analyzing the transmission duration of a SCHC Packet theoretically. A SCHC Packet is divided in fragments, which could be either Regular, All-0 or All-1. First of all, the duration of Sigfox U-procedure, which corresponds to regular fragments, is analyzed, followed by the one of Sigfox B-procedure, which corresponds to All-0 and All-1 fragments. Finally, the duration of sending SCHC Packet is analyzed for packet size up to 2250 bytes. The fragmentation time is not included in the total time.

3.1.1 Duration of a U-procedure:

During the U-procedure, three transmissions take place, the original message and two replicas of it. There are three phases during the procedure:

1. Transmission phase
2. Wait next transmission phase
3. Cooldown phase

The different phases are illustrated in the Figure 3.1:

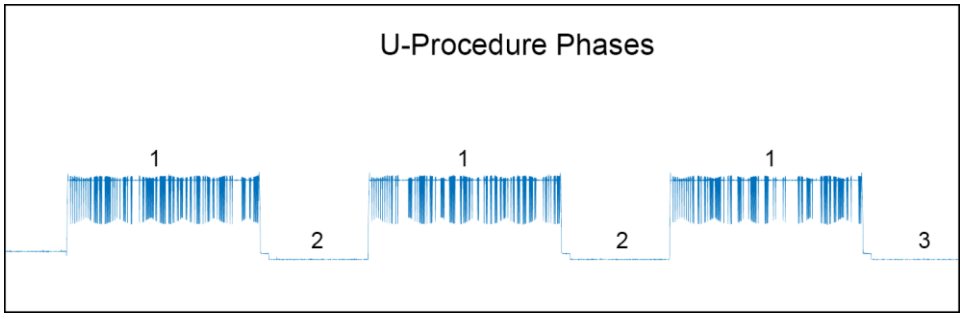

Figure 3.1: Phases of a Sigfox U-procedure

The transitions between the phases can be seen in Figure 3.1. Note that phase one happens 3 times and phase two happens 2 times.

The duration of each phase is shown in Table 3.1:

Table 3.1. Duration of U-procedure phases

Phase	Notation	Duration (ms)
1	T_{Tx}	[1120,2080]
2	T_{WaitTx}	1000
3	T_{Cool}	1000

T_{Tx} is the only phase whose duration is variable. The actual transmission time, T_{Tx}, depends on the Sigfox frame length and the bitrate used. More specifically, the duration is based on Equation (3.1) shown below:

$$T_{Tx}(sec) = \frac{L_{SIGFOX-FRAME}}{Bitrate} \quad (3.1)$$

In order to find the length of the frame, we need to take into account that Sigfox radio specifications define an uplink frame as shown in Table 3.2:

Table 3.2. Sigfox uplink frame fields

Size (bits)	19	29	32	0-96	16-40	16
Frame field	Preamble	Frame synch and header	Device ID	Payload (SCHC Fragment)	Message authentication code (MAUTH)	FCS

According to Sigfox, depending on the payload size, the Message Authentication Code (MAUTH) will be as shown in Table 3.3:

Table 3.3. Sigfox MAUTH size

Payload size (bytes)	0	1	2	3	4	5	6	7	8	9	10	11	12
Mess. auth. code size (bytes)	2	2	4	3	2	5	4	3	2	5	4	3	2

That means that the total length of the frame will be between 112 (0-byte payload) and 208 bits (12-byte payload).

Regarding the bitrate, Sigfox defines two different ones, 100 bps and 600 bps. In Europe, the one defined is 100 bps. As a result, the transmission time will be:

$$T_{Tx}(sec) = \frac{96 + MAUTH + Payload}{100} \quad (3.2)$$

Summing up all the aforementioned, the total duration of a Sigfox U-procedure (in seconds) would be:

$$T_{U-Total}\ (sec) = 3 * T_{Tx} + 2 * T_{WaitTx} + T_{Cool} = 3 * \frac{96 + MAUTH + Payload}{100} + 2 * 1 + 1$$

$$T_{U-Total}\ (sec) = 3 + 3 * \frac{96 + MAUTH\ +\ Payload}{100} \quad (3.3)$$

In the worst case, when the maximum payload size is 12 bytes, the duration will be 9.24 seconds. In the best case, when the payload size is 0 bytes, the duration will be 6.36 seconds.

3.1.2 Duration of a B-procedure:

This procedure includes a downlink message, which follows an uplink transmission. The procedure includes the following phases:

1. Transmission phase
2. Wait next transmission phase
3. Wait next reception phase
4. Reception phase
5. Confirmation transmission phase
6. Cooldown phase

The different phases and the transitions are illustrated in Figure 3.2:

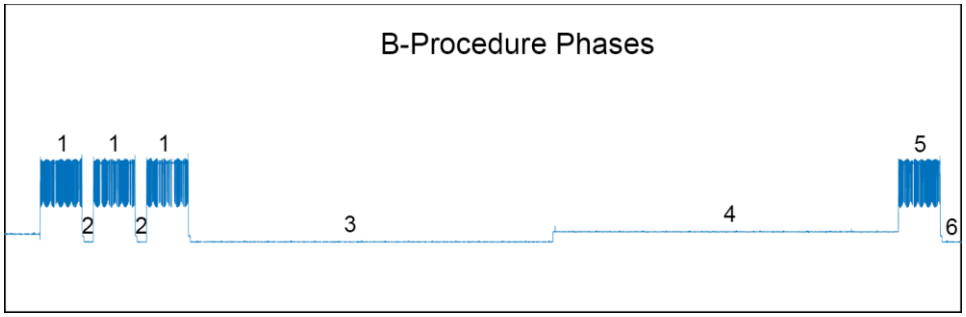

Figure 3.2: Phases of a Sigfox B-procedure

The duration of the phases are shown in Table 3.4:

Table 3.4. Duration of Sigfox B-procedure phases

Phase	Notation	Duration (ms)
1	T_{Tx}	[1120,2080]
2	T_{WaitTx}	475
3	T_{WaitRx}	15556
4	T_{Rx}	[387,25000]
5	T_{Conf}	1799
6	T_{Cool}	1000

Phases 1,2 and 6 are the same as the U-procedure. The durations of phases 3 and 5 are constant. The duration of the reception phase is the only variable one. The duration value shown in Table 3.4 is a mean value of the possible values it may have. More specifically, theoretically, the base station may transmit to the device at any time between the start of the reception window and its end, which is $T_{Window\_Rx\_Max}$ = 25 s. The actual duration of the reception is calculated as shown in Equation (3.4):

$$T_{Rx-DL}(sec) = \frac{L_{SIGFOX-FRAME}}{Bitrate} \quad (3.4)$$

Sigfox radio specifications define a downlink frame as shown in Table 3.5:

Table 3.5. Sigfox Downlink Frame fields

Size (bits)	91	13	32	64	16	16
Frame field	Preamble	Frame synch and header	ECC	Payload	Message authentication code (MAUTH)	FCS

By observing Table 3.5 above, we can extract that the total length is 232 bits.

In addition, the Sigfox bitrate in the downlink is 600 bps. So, we conclude that:

$$T_{Rx-DL}(sec) = \frac{232}{600} = 0.387 \quad (3.5)$$

The reception time can be any value between 0.387 and 25 seconds. However, it is observed experimentally that the reception happens between 14 and 15 seconds after the opening of the reception window, in our considered scenario. As a result, we can take the mean of those values as the average T_{Rx}:

$$T_{Rx-Mean}\ (sec) = \frac{14 + 15}{2} = 14.5 \quad (3.6)$$

Summing up, the total transmission time of a Sigfox B-procedure (in seconds) would be:

$$T_{B-Total}\ (sec) = 3 * T_{Tx} + 2 * T_{WaitTx} + T_{WaitRx} + T_{Rx} + T_{Conf} + T_{Cool}$$
$$T_{B-Total}\ (sec) = 3 * \frac{96 + MAUTH\ +\ Payload}{100} + 2 * 0.475 + 15.556 + T_{Rx} + 1.799 + 1$$
$$T_{B-Total}\ (sec) = 19.305 + 3 * \frac{96 + MAUTH\ +\ Payload}{100} + T_{Rx} \quad (3.7)$$

Taking into account Equation (3.6), Equation (3.7) can be updated as:

$$T_{B-Total-Mean-Window}\ (sec) = 33.805 + 3 * \frac{96 + MAUTH\ +\ Payload}{100} \quad (3.8)$$

There is one special case, where the downlink message is lost or never transmitted from the base station. In that case, we will assume that the device never passes from phases 5 and 6, that correspond to the transmission of the final confirmation of the device to the base station and the cooldown phase. In addition, as the downlink message is not received, we can be sure that the device was waiting for the maximum reception window, which is 25 seconds. That means that the total time will be:

$$T_{B-Total-no-DL}\ (sec) = 3 * T_{Tx} + 2 * T_{WaitTx} + T_{WaitRx} + T_{Window-Rx-Max}$$
$$T_{B-Total-no-DL}\ (sec) = 3 * \frac{96+MAUTH + Payload}{100} + 2 * 0.475 + 15.556 + 25$$
$$T_{B-Total-no-DL}\ (sec) = 41.506 + 3 * \frac{96+MAUTH + Payload}{100} \quad (3.9)$$

3.1.3 Transmission duration of SCHC Packets

In this section, the SCHC Packet transmission duration is analysed. The total duration corresponds to the cumulative duration of all fragments sent.

The most important parameters, assumptions and choices of the analysis are :

- ACK-on-Error mode of transmission is used. In this mode:
  - Windows are used
  - B-procedure is used for the last fragment of each window:
    - If it is not the last window, an ACK will be sent from the Sigfox cloud only if there is a transmission loss (*All-0 SCHC Fragment*).
    - If it is the last window, an ACK will be sent whether there is a loss or not (*All-1 SCHC Fragment*).
  - U-procedure is used for all the fragments of the window, except the last one (*Regular SCHC Fragment*).
- 7 fragments per window are used for SCHC Packets until 300 bytes, and 31 fragments per window for SCHC Packet larger than 300 bytes.
- All fragments have the maximum payload available, which is 12 bytes, except for the last one, that may have less fragments.
- We assume that there are no transmission losses, both in the uplink and the downlink.
- The analysis is done for 2 cases:
  - (a). In the first case, the transmission duration of only the awake part of the device is shown.
  - (b). In the second scenario, the duty cycle is taken into consideration. That would mean that the sleep time will also be considered in the transmission duration. More specifically, the device will send 6 fragments, go to sleep for the rest of the hour, before waking up again to send the next fragments.
- The figures showing the results presented in the analysis are done with 1-byte granularity of packet sizes, ranging from 0 to 2250 bytes.

3.1.3.1 Transmission duration without duty cycle restriction

In this scenario only the awake part of the transmission of the fragments is taken into consideration.
The transmission duration for each fragment type would be:

- Equation (3.3) is used for the Regular SCHC fragments, as they are initiating a U-procedure. In addition, as we will always have a payload of 12 bytes, the transmission duration will be constant and will be:

$$T_{U-Total}\ (s) = 3 * T_{Tx} + 2 * T_{WaitTx} + T_{Cool} = 3 + 3 * \frac{96 + 2 * 8\ +\ 12 * 8}{100} = 9.24\ s$$

- Equation (3.9) is used for the All-0 fragments, as in that case, taking into account that we have no losses, there will be no response from the cloud. In addition, as the payload will always be 12 bytes, the transmission duration will be constant and will be:

$$T_{B-Total-no-DL}\ (s) = 3 * T_{Tx} + 2 * T_{WaitTx} + T_{WaitRx} + T_{Window-Rx-Max}$$
$$= 41.506 + 3 * \frac{96 + 2 * 8\ +\ 8 * 12}{100} = 47.746\ s$$

- Equation (3.8) is used for the All-1 SCHC fragments, as there will always be a response from the cloud.

The transmission duration will not be constant for all packets, as it will depend on the payload size of the last fragment.

$$T_{B-Total-Mean-Window}\ (s) = 33.805 + 3 * \frac{96 + MAUTH\ +\ Payload}{100}$$

To sum all those, Equation (3.10) describes the theoretical errorless transmission duration of a SCHC Packet:

$$T_{SCHC-Packet}\ (s) = Regular * 9.24\ +\ All0 * 47.746 + 33.805 + 3 * \frac{All1_{size}(bits)}{100} \quad (3.10)$$

In Figure 3.3 the packet transmission duration is plotted for the range of packet sizes between 0 and 2250 with 1-byte granularity:

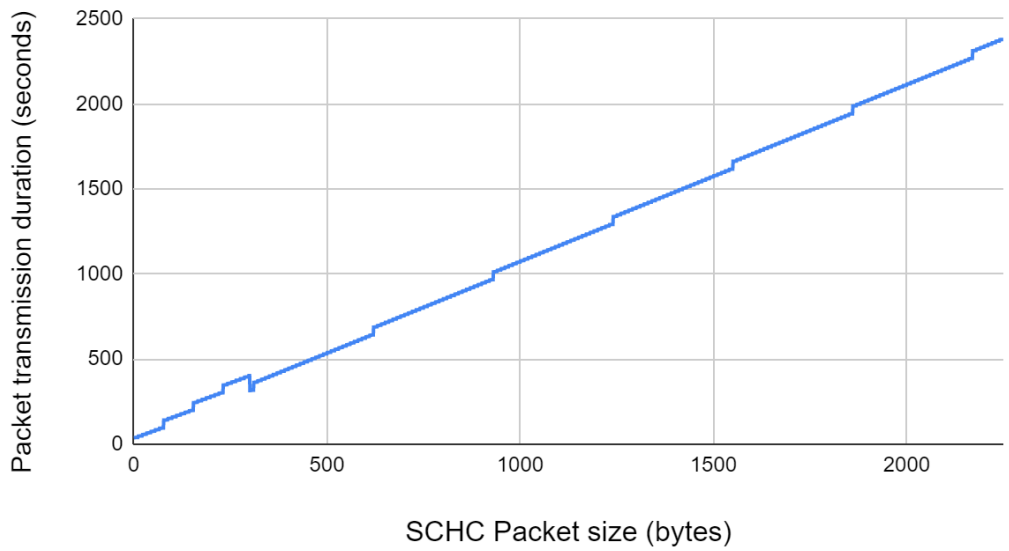

Figure 3.3: Theoretical errorless transmission duration of SCHC Packet (0-2250 bytes)

In order to have a better look on the first 512 sizes, Figure 3.4 presents that region in zoom:

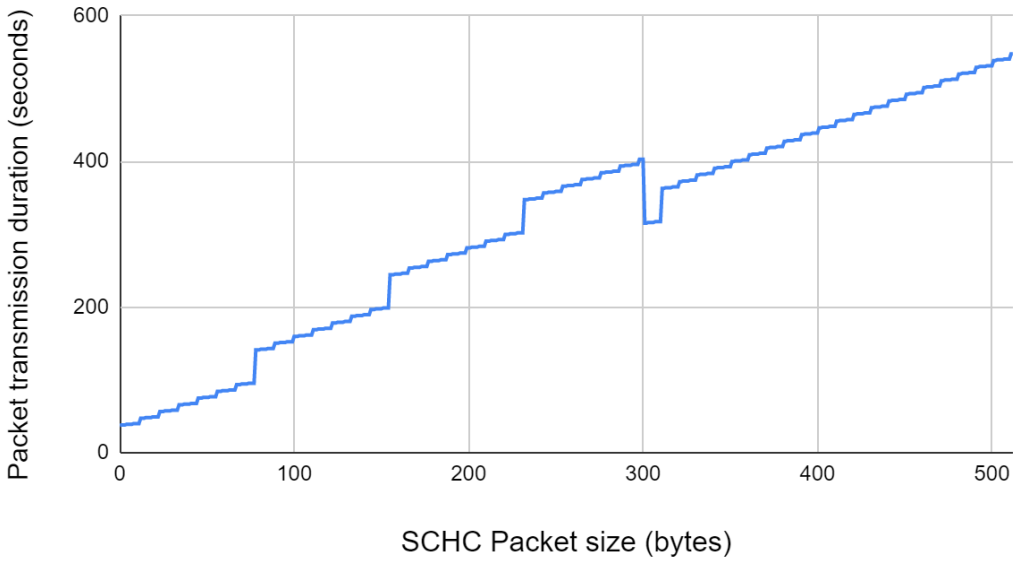

Figure 3.4: Theoretical errorless transmission duration of SCHC Packet (0-512 bytes)

The following observations can be concluded from the calculations and the figures:

- The behaviour is relatively linear, which means that the transmission duration increasses accordingly to the packet length.
- Several steps can be observed in Figure 3.4. This happens when a packet needs one more window. More specifically, around 45 additional seconds are added to the total duration.

- There is a special point in Figure 3.4 where a sudden decrease can be observed. That happens in the region of 300 bytes. At 301 bytes size, windows are composed of 31 fragments instead of 7, which results in having less windows for the case of 301 than the one of 300 bytes. Having less windows results in less All-0 fragments, and consequently less total packet transmission duration.
- The minimum transmission duration, which corresponds to the smallest SCHC Packet size consisting of only 1 fragment, is 37.405 seconds. The maximum transmission duration, which corresponds to the largest SCHC Packet size of 2250 bytes and is fragmented in 225 fragments, is 39 minutes and 39 seconds

3.1.3.2 Transmission duration with duty cycle restriction

In that case, as mentioned above, the duty cycle is taken into consideration in the transmission duration. That would mean that for each 6 fragments sent, the device should sleep for the rest of the hour. In other words, the total duration would be:

$$T_{SCHC-Packet}\ (s) = ceiling(Fragments/6)\ * 3600 \qquad (3.11)$$

Figure 3.5 shows the packet transmission duration for the range of packet sizes between 0 and 2250 with 1-byte granularity:

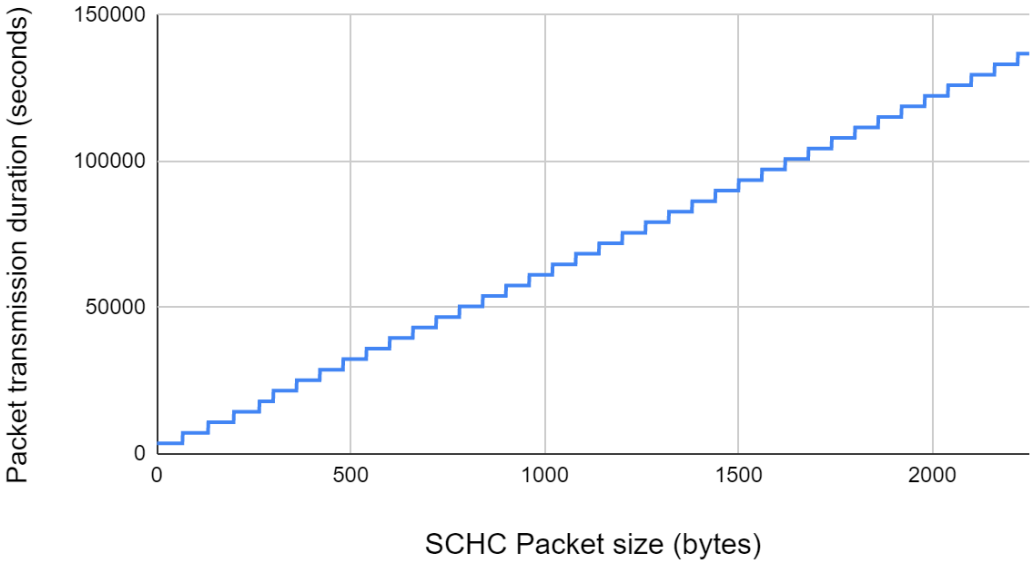

Figure 3.5: Theoretical errorless transmission duration of SCHC Packet taking into consideration duty cycle

Looking at Figure 3.5, we can conclude the following:

- The behavior is a bit different than case (a). The steps in Figure 3.5 are observed every 6 additional fragments, as that would mean an additional hour of device sleep.
- The decrease at 300 bytes though is not observed. On the other hand, an increase is observed. This happens because now the number of windows is not relevant, as the parameter affecting the result is the number of fragments.
- The total time needed for the transmission of large packet sizes is considerably large. The smallest SCHC Packet would need 1 hour, while the largest packet of 2250 bytes, would need 38 hours.
- Comparing the 2 scenarios, we observe that the duration in scenario (b) is 40 to 95 higher than scenario (a). It is obvious that this is caused by the duty cycle restriction.

3.2 Experimental scenario and transmission durations of errorless transmissions

This section describes the testbed used for the experimental tests of SCHC over Sigfox. In addition, the results of the tests for errorless tranmissions are presented afterwards.

3.2.1 SCHC over Sigfox evaluation testbed

In order to evaluate the model and to measure experimentally the transmission durations in real scenarios, a testbed was created. This testbed was used for all the experimental scenarios described in the Master Thesis.

3.2.1.1 Testbed Hardware

The hardware used for the evaluation is:

- Pycom LoPy4 (34.95 €)
- LoRa & Sigfox Antenna Kit (9 €)
- Expansion Board 3.0 (16 €)

LoPy4 is a Pycom Micropython-programmable quadruple bearer board. It is compatible with LoRa, Sigfox, WiFi and Bluetooth. Through the Expansion Board, it can be connected to a USB port. More information about LoPy4 can be found in its datasheet at [11]. The development of the board was done through a plugin of Visual Studio Code [12], called Pymakr [13], which supports micropython.

3.2.1.2 Testbed Architecture

For the sake of the evaluation of SCHC over Sigfox, a testbed was created, in collaboration with the Sigfox company and the University of Chile. Its architecture is illustrated in Figure 3.6:

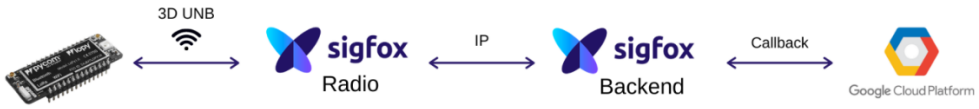

Figure 3.6: SCHC over Sigfox testbed architecture

The device used for the evaluation is the Pycom LoPy4 using a LoRa & Sigfox Antenna Kit. The device has the role of the sender/fragmenter, while the receiver/reassembler is in the Google Cloud Platform (GCP). More information about GCP can be found on its documentation at [REF].

Following the sequence illustrated, the device transmits a message through its antenna, which is received from the Sigfox base stations through the 3D-UNB network. Then, through IP links the message is sent to the Sigfox backend. Although the message is accessible through the Sigfox backend platform, it can not be processed. That is why it triggers a callback, which sends the message to the GCP. The callback triggers a Cloud Function, which decodes the message and does the required processing. In addition, it saves the message to Cloud Storage.

Initially, the device performs the fragmentation of the SCHC Packet. Following that, it sends the fragments one by one. On the other side, GCP, after receiving all messages successfully, reassembles the initial SCHC Packet.

Some examples of how Sigfox backend and GCP are working are shown in Figures 3.7 and 3.8:

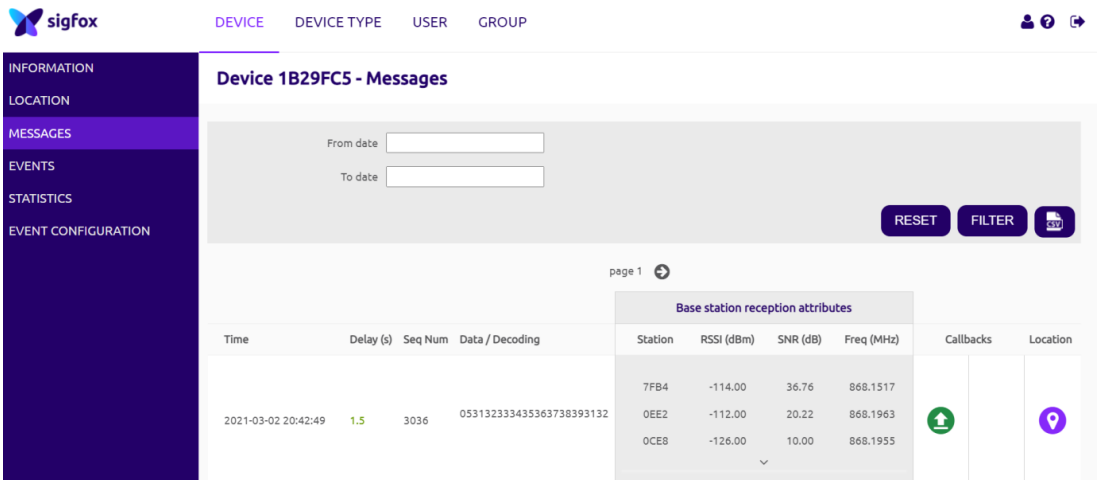

Figure 3.7: A message in the Sigfox backend

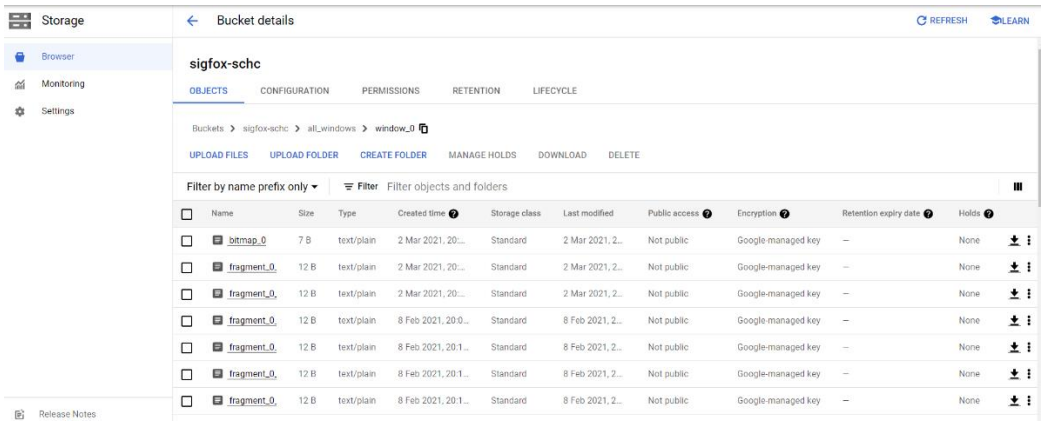

Figure 3.8: SCHC Fragments saved in Cloud Storage

3.2.2 Experimental transmission durations of errorless transmissions

For the evaluation of the theoretical model, some SCHC Packet sizes were selected. The packet sizes are shown in Table 3.6 below:

Table 3.6. Packet sizes chosen for the experimental analysis

SCHC Packet length (bytes)	0	11	20	22	77	90	150	231	233	512	1280	2250
SCHC Header (bytes)	1	1	1	1	1	1	1	1	1	2	2	2
Fragments	1	1	2	2	7	9	14	21	22	52	128	225
Windows	1	1	1	1	1	2	2	3	4	2	5	8

The tests were done using the LoPy4 presented in section 3.2.2.1. The location of the experiment was Barcelona, Spain. The coverage in the area of the experiments can be considered reliable. More specifically, no UL errors were observed, while on DL the error rate was around 1%.

Showing the results considering the duty cycle would not be very useful, as due to the sleep time the duration depends only on the number of fragments, the results would be identical to the theoretical ones.

So, the results only include the duration of the awake part of the device transmission. In Table 3.7 below the results are shown and compared to the theoretical ones.

Table 3.7. Experimental errorless SCHC Packet transmission duration

SCHC Packet length (bytes)			0	11	20	22	77	90	150	231	233	512	1280	2250
Fragments			1	1	2	2	7	9	14	21	22	52	128	225
Windows (7 or 31 fragments per window)			1	1	1	1	1	2	2	3	4	2	5	8
Transmission duration (seconds)	Regular Fragments	Count	0	0	1	1	6	7	12	18	18	50	123	217
		Total	0	0	9.36	9.36	56.11	65.52	112.54	169.63	169.02	469.22	1156.24	2040.25
		Mean	0	0	9.36	9.36	9.35	9.36	9.38	9.42	9.39	9.38	9.4	9.40
		St. deviation	0	0	0	0	0.00	0.03	0.01	0.05	0.02	0.02	0.03	0.03
	All-0 Fragments	Count	0	0	0	0	0	1	1	2	3	1	4	7
		Total	0	0	0	0	0	47.42	47.43	94.91	142.28	47.43	189.71	332.12
		Mean	0	0	0	0	0	47.42	47.43	47.45	47.43	0	47.43	47.45
		St. deviation	0	0	0	0	0	0	0	0.04	0.00	0	0.00	0.03
	All-1 Fragments	Total	37.49	38.74	38.98	39.24	38.07	37.42	39.03	38.97	37.47	37.89	38.04	38.37
		Mean	38.30752917											
		St. deviation	0.670224817											
	Total transmission duration		37.49	38.74	48.33	48.60	94.18	150.36	198.99	303.51	348.77	554.54	1383.99	2410.74
Network messages exchanged			1 UL 1 DL	1 UL 1 DL	2 UL 1 DL	2 UL 1 DL	7 UL 1 DL	9 UL 1 DL	14 UL 1 DL	21 UL 1 DL	22 UL 1 DL	52 UL 1 DL	128 UL 1 DL	225 UL 1 DL
Theoretical result			37.41	40.05	49.29	49.29	95.49	150.55	197.71	301.856	347.68	547.87	1367.55	2379.35
Theoretical - Experimental difference			0.08	-1.31	-0.95	-0.69	-1.31	-0.19	1.28	1.65	1.08	6.67	16.45	31.39
Theoretical - Experimental difference %			0.22%	3.26%	1.93%	1.40%	1.37%	0.13%	0.65%	0.55%	0.31%	1.22%	1.20%	1.32%

Observing Table 3.7, the following can be concluded:

- The experimental transmission duration values are relatively close to the theoretical values. The differences, especially expressed as a percentage, do not exceed 3.5%.
- It can be observed that the average transmission duration of a Regular fragment is around 9.4 seconds, while the theoretical value was 9.24 seconds. That accumulates a difference of ~20 ms per regular fragment sent. For relatively big SCHC Packet sizes (bigger than 200 bytes) the number of fragments is large enough so that this difference could affect the result. That is why for large packets the experimental result tends to be higher. On the other hand, for smaller SCHC Packet sizes, it is observed that there are cases where the experimental value is greater and others where the theoretical one is greater. This variability comes from the variability of the reception of the All-1 fragment. As explained in section 5.1, the Sigfox downlink message arrival time has some variability and the average is taken as the theoretical value. This variability, together with the fact that there is not significant accumulated difference because of the regular fragments, causes the results to be sometimes higher experimentally and some other times higher theoretically for small SCHC Packets.

3.3 Experimental transmission durations with controlled errors

This section will focus on analysing the transmission duration experimentally of controlled uplink and downlink errors.

Controlled errors means that those errors were forced by the application. More specifically, the notation used below is “X (Y Windows)”. This means that X errors were introduced in the SCHC Packet transmission that were distributed in Y Windows. For example, “4 (2 Windows)” would mean that 4 errors happened and were distributed in 2 windows, so there were 2 errors per error window.
The section will be divided into three subsections: UL-only controlled errors, UL & DL controlled errors and DL-only controlled errors.

3.3.1 UL-only controlled errors

In that subsection, the errors happen only in the UL. The SCHC Packet sizes chosen are 77, 90, 150 and 231 bytes. Table 3.8 below sums up the results taken:

Table 3.8. Experimental results of transmission duration for controlled UL-Only errors

SCHC Packet length (bytes)			77			90			150					231						
Fragments			7			9			14					21						
Windows (7 fragments per window)			1			2			2					3						
Errors introduced			No errors	1 (1 win.)	2 (1 win.)	No errors	1 (1 win.)	2 (1 win.)	No errors	1 (1 win.)	2 (1 win.)	4 (1 win.)	4 (2 win.)	No errors	1 (1 win.)	4 (1 win.)	4 (2 win.)	6 (1 win.)	6 (2 win.)	6 (3 win.)
Transmission duration (seconds)	Regular Fragments	Count	6	7	8	7	8	9	12	13	14	16	16	18	19	22	22	24	24	24
		Total	56.39	65.80	75.47	65.52	0.00	84.17	112.54	122.6	130.91	150.54	149.69	169.63	177.68	205.73	205.80	224.44	224.42	224.5
		Mean	9.40	9.4	9.43	9.36	9.35	0.00	9.38	9.43	9.35	150.54	9.36	9.42	9.35	9.35	9.35	9.35	9.35	9.35
		St. deviation	0.02	0.03	0.03	0.02	9.42	9.35	0.01	0.03	0.00	150.54	0.02	0.05	0.00	0.00	0.02	0.00	0.00	0.02
	All-0 Fragments	Count	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
		Total	0	0	0	47.42	39.01	38.15	47.43	37.80	38.34	38.11	38.86	94.91	86.24	85.83	76.92	85.73	76.07	78.06
		Mean	0	0	0	47.42	39.01	38.15	47.43	37.80	38.34	38.11	38.86	47.45	43.12	42.91	38.46	42.87	38.03	39.03
		St. deviation	0	0	0	0	0	0	0	0	0	0	0	0.05	6.08	6.37	0.21	6.44	0.23	0.38
	All-1 Fragments	Count	1	2	2	1	1	1	1	1	1	1	2	1	1	1	1	1	1	2
		Total	38.58	78.39	77.47	37.42	37.97	37.54	39.03	38.08	38.32	37.84	76.25	38.97	38.52	38.23	39.19	38.36	37.82	78.17
		Mean	38.58	39.19	38.74	37.42	37.97	37.54	39.03	38.08	38.32	37.84	38.13	38.97	38.52	38.23	39.19	38.36	37.82	39.09
		St. deviation	0	0.31	0.10	0	0	0	0	0	0	0	0.26	0	0	0	0	0	0	0.13
	Total duration		94.70	144.19	152.94	150.36	151.79	159.86	198.99	198.49	207.57	226.49	264.80	303.51	302.44	329.79	321.90	348.53	338.31	380.73
	Time difference from no error		-	49.48	58.24	-	1.43	9.50	-	-0.51	8.58	27.50	65.81	-	-1.07	26.28	18.39	45.02	34.80	77.22
Network messages exchanged			7 UL	9 UL	10 UL	9 UL	10 UL	11 UL	14 UL	15 UL	16 UL	18 UL	19 UL	21 UL	22 UL	25 UL	25 UL	27 UL	27 UL	28 UL
			1 DL	2 DL	2 DL	1 DL	2 DL	2 DL	1 DL	2 DL	2 DL	2 DL	3 DL	1 DL	2 DL	2 DL	3 DL	2 DL	3 DL	4 DL

As seen above, when considering the duty cycle, there are small differences, only when a lot more fragments are sent. However, the most interesting part is analysing the differences between the duration of not including the duty cycle time.

Figures 3.9 and 3.10 are giving a more clear illustration of the results:

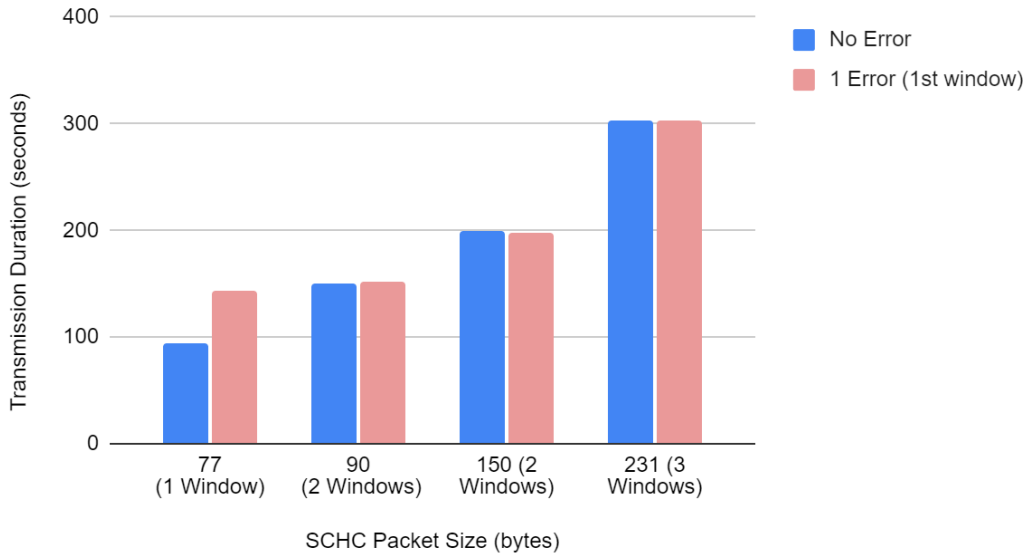

Figure 3.9: ACK-on-Error transmission duration of controlled UL errors (1 error)

Figure 3.9 shows the differences when only one error is introduced, compared to the case where no errors happen. It is observed that for the case of 77 bytes there is an increase of around 40 seconds. This happens because, as the error happens in the first and only window, an additional All-1 message, apart from the retransmission of the lost fragment, should be transmitted, resulting in that increase. However, in the rest of the cases, the transmission duration is not increased, but instead it is also a bit shorter in the cases of 150 and 231 bytes. That happens because, as the error happens in the first window, the All-0 message results in a received DL message with the ACK bitmap. When there is a DL message in the All-0 the duration of the All-0 message is around 10 seconds shorter. Those 10 seconds are compensated from the retransmission of the lost fragment. Finally, there is no need for an additional All-1 message (as in the case of 77 bytes), as the error happened in the first window. That is why the total transmission duration is more or less the same.

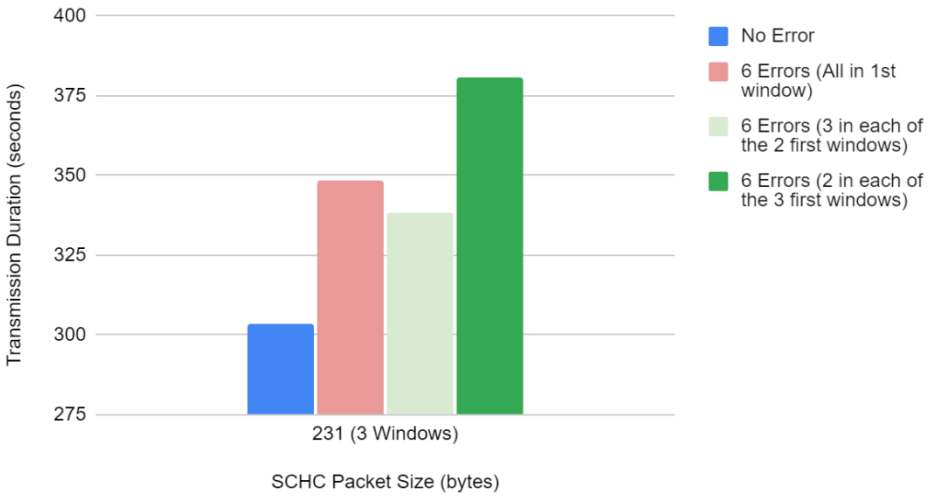

Figure 3.10: ACK-on-Error transmission duration of controlled UL errors (6 errors)

In Figure 3.10, the different cases of 6 errors are shown. The difference is the distribution of the errors in the different windows. We observe that the case where the errors happen in the first two windows is around 10 seconds shorter than the case where they all happen in the first window. That is why in the first one there are two received DL ACKs in response to the All-0 message, while in the second case there is only one received DL ACK, causing that difference of 10 seconds. In the last case, where the errors happen in all 3 windows there is a significant increase of around 40 seconds. This happens due to the fact that there is an error in the last window which results in an additional All-1 message.

3.3.2 UL & DL controlled errors

In this section the SCHC Packet size under consideration is the one of 231 bytes. More specifically, the tests included 1 UL error, 1 UL-1 DL error and 1 UL - 2 DL errors. The results are presented Table 3.9. and illustrated in Figure 3.11. In the first, second and third case the duration is pretty much the same. More specifically, in the second case, the explanation is the same as in the UL-only case. In the third case, although there is a DL error, there is no effect, as the lost DL of the first window will be resent and received successfully at the end of the second window. However, 2 DL errors will result in the first successfully received DL message being the response of the All-1 message. As a result, one additional All-1 message will be transmitted in the end, causing the additional 47 seconds.

Table 3.9. Experimental results of transmission duration for controlled UL & DL errors

SCHC Packet length (bytes)			231			
Fragments			7			
Windows (7 fragments per window)			1			
Errors introduced			No errors	1 UL	1 UL 1 DL	1 UL 2 DL
Transmission duration (seconds)	Regular Fragments	Fragments	18	19	19	19
		Total	169.63	177.68	177.78	177.69
		Mean	9.42	9.35	9.36	9.35
		St. deviation	0.05	0.00	0.00	0.00
	All-0 Fragments	Fragments	2	2	2	2
		Total	94.91	86.24	86.47	94.84
		Mean	47.45	43.12	43.24	47.42
		St. deviation	0.04	6.08	5.92	0.00
	All-1 Fragments	Fragments	1	1	1	2
		Total	38.97	38.52	39.68	78.82
		Mean	38.97	38.52	39.68	39.41
		St. deviation	0	0	0	0.18
	Total duration		303.51	302.44	303.93	351.35
	Time difference from no error		-	-1.07	0.42	47.84
Network messages exchanged			21 UL 1 DL	22 UL 2 DL	22 UL 3 DL	23 UL 4 DL

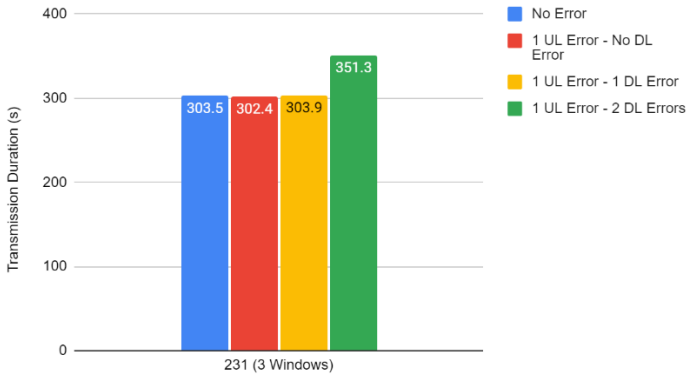

Figure 3.11: Transmission duration of controlled UL & DL errors (1 error)

3.3.3 DL-Only controlled errors

In this subsection the SCHC Packet size under consideration is the one of 231 bytes. More specifically, the tests included 1 UL error, 1 UL-1 DL error and 1 UL - 2 DL errors. The results are presented in Table 3.10 and illustrated in Figure 3.12.

In this case, as there are only DL errors, it means that all UL fragments were successfully received. That would mean that there is no need to send a DL ACK, until the final one in response to the All-1 message. Therefore, each DL error will cause an additional All-1 message to be sent, increasing the total transmission duration by around 45 seconds.

Table 3.10. Experimental results of transmission duration for controlled DL-only errors

SCHC Packet length (bytes)			231		
Fragments			21		
Windows (7 fragments per window)			3		
DL errors Introduced			No errors	1	2
Transmission duration (seconds)	Regular Fragments	Amount	18	18	18
		Total	169.63	168.33	168.42
		Mean	9.42	9.35	9.36
		St. deviation	0.05	0.01	0.01
	All-0 Fragments	Amount	2	2	2
		Total	94.91	94.84	94.85
		Mean	47.45	47.42	47.42
		St. deviation	0.04	0.00	0.00
	All-1 Fragments	Amount	1	2	3
		Total	38.97	86.77	134.86
		Mean	38.97	43.38	44.95
		St. deviation	0	5.71	4.28
	Total duration		303.51	349.94	398.12
	Time difference from no error		-	46.43	94.62
Network messages exchanged			21 UL 1 DL	22 UL 2 DL	23 UL 3 DL

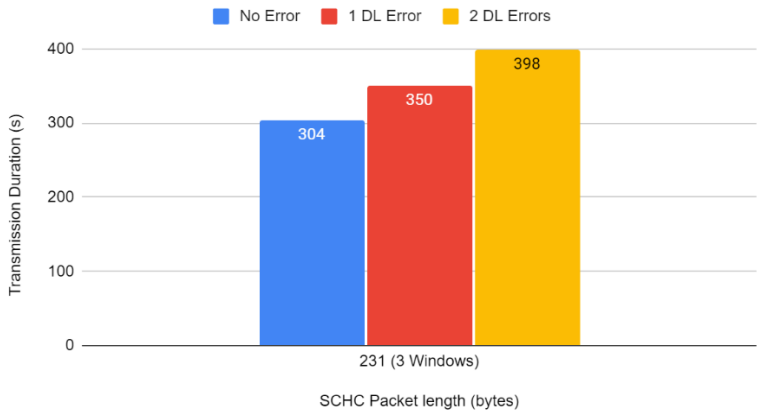

Figure 3.12: Transmission duration of controlled DL-only errors

3.4 Experimental transmission durations with random errors

This section will focus on analysing the experimental transmission duration of random uplink and downlink errors.

Random error means that a reference probability value is decided that indicates the probability of a message being lost. For example, 20% would mean that a message has a probability of 20% being lost.

The SCHC Packet sizes chosen for that analysis are 77 bytes, 150 bytes, 231 bytes and 512 bytes. In addition, the random error probabilities chosen are 10% and 20%. Given that we analyse also the cases of UL-only and UL-DL, in total there are 16 different cases (4 packet sizes x 2 error probabilities x 2 scenarios). For each case, 10 repetitions were done and the average of those is shown in the results. Table 3.11 shows the results for the case of UL-Only errors, while Table 3.12 shows the results for UL & DL Randrom errors. In Figures 3.13 - 3.16, 4 metrics are presented: the transmission duration with and without the consideration of the duty cycle, the UL messages sent and the DL messages sent.

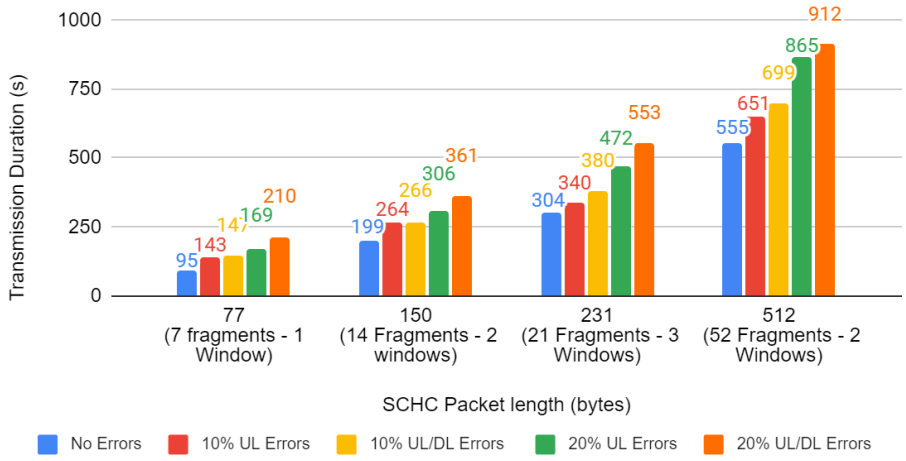

Figure 3.13: Random errors transmission duration (not considering duty cycle)

In Figure 3.13, the awake transmission time of the device is shown. As expected, when the error probability increases, the duration also increases. Another observation is that the larger the SCHC Packet is, the more % increase is observed for the same error probability. That is also expected, as larger packets will consist of more fragments, so there are more errors, which cause more All-1 messages to be sent and make the total transmission duration longer.

Table 3.11. UL-Only results of random errors

SCHC Packet length (bytes)			77			150			231			512		
Fragments			7			14			21			52		
Windows			1			2			3			2		
Case			No errors	10% UL errors	20% UL errors	No errors	10% UL errors	20% UL errors	No errors	10% UL errors	20% UL errors	No errors	10% UL errors	20% UL errors
Transmission duration (seconds)	Regular Fragments	Amount	6	6	6	12	12	12	18	18	18	50	50	50
		Sent	6	6.85	7.25	12	14	15.4	18	19.8	22.9	50	54.5	63.5
		Errors	0	0.85	1.25	0	2	3.4	0	1.8	4.9	0	4.5	13.5
		Total	56.39	64.07	67.81	112.54	130.96	144.03	169.63	185.19	214.2	469.22	509.83	593.91
		Mean	9.40	9.35	9.35	9.38	9.35	9.35	9.42	9.35	9.35	9.38	9.36	9.35
		St. Deviation	0.02	0.00	0.01	0.01	0.01	0.00	0.05	0.01	0.01	0.02	0.01	0.01
	All-0 Fragments	Amount	0	0	0	1	1	1	2	2	2	1	1	1
		Sent	0	0	0	1	1.1	1.2	2	2	2.4	1	1.1	1.2
		Errors	0	0	0	0	0.1	0.2	0	0	0.4	0	0.1	0.2
		Total	0	0	0	47.43	45.86	42.84	94.91	86.71	88.66	47.43	41.07	42.88
		Mean	0	0	0	47.43	43.02	37.16	47.45	43.36	37.84	47.43	38.23	37.20
		St. Deviation	0	0	0	0	2.69	5.38	0.05	4.67	9.98	0	0	5.38
	All-1 Fragments	Amount (No error)	1	1	1	1	1	1	1	1	1	1	1	1
		Amount needed to be sent (Accounting errors)	1	1.65	1.9	1	2.1	2.7	1	1.7	2.5	1	2.2	4.8
		Sent	1	1.85	2.45	1	2.2	3	1	1.7	4	1	2.5	5.7
		Errors	0	0.2	0.55	0	0.1	0.3	0	0	1.5	0	0.3	0.9
		Total	38.58	78.86	101.17	39.03	86.84	118.69	38.97	67.61	169.28	37.89	100.09	224.41
		Mean	38.58	39.98	40.68	39.03	39.42	38.92	38.97	39.56	41.72	37.89	39.73	39.83
		St. deviation	0	1.36	1.76	0	0.76	1.73	0	0.56	3.19	0	1.19	2.09
	Total duration (no duty cycle)		94.70	142.93	168.98	199	263.66	305.56	303.51	339.52	472.14	554.54	650.99	865.17
	% Increase from no error		0.00%	50.93%	78.43%	0.00%	32.50%	53.56%	0.00%	11.86%	55.56%	0.00%	17.39%	56.01%
	Total duration (duty cycle)		43200	43200	43200	64800	64800	86400	86400	86400	108000	194400	216000	259200
Total UL errors			0	1.05	1.8	0	2.2	3.9	0	1.8	6.8	0	4.9	14.6
UL errors %			0.00%	12.07%	18.56%	0.00%	12.72%	19.90%	0.00%	7.66%	23.21%	0.00%	8.43%	20.74%
Network messages exchanged		UL	7	8.7	9.7	14	17.3	19.6	21	24	29.3	52	58.1	70.4
		DL	1	1.7	1.9	1	2.3	3.5	1	3	3.9	1	3.1	5.5

Table 3.12. UL & DL Results of random errors

SCHC Packet length (bytes)			77			150			231			512		
Case			No errors	10% UL/DL errors	20% UL/DL errors	No errors	10% UL/DL errors	20% UL/DL errors	No errors	10% UL/DL errors	20% UL/DL errors	No errors	10% UL/DL errors	20% UL/DL errors
Transmission duration (seconds)	Regular Fragments	Amount	6	6	6	12	12	12	18	18	18	50	50	50
		Sent	6	7	7.3	12	14.2	15	18	20.6	23.5	50	56.9	64.5
		Errors	0	1	1.3	0	2.2	3	0	2.6	5.5	0	6.9	14.5
		Total	56.39	65.00	68.28	112.54	132.83	140.28	169.63	192.67	219.8	469.22	532.22	603.3
		Mean	9.40	9.35	9.35	9.38	9.35	9.35	9.42	9.35	9.35	9.38	9.35	9.35
		St. deviation	0.02	0.00	0.00	0.01	0.01	0.00	0.05	0.00	0.01	0.02	0.01	0.01
	All-0 Fragments	Amount	0	0	0	1	1	1	2	2	2	1	1	1
		Sent	0	0	0	1	1.1	1.1	2	2.4	2.5	1	1	1.4
		UL errors	0	0	0	0	0.1	0.1	0	0.4	0.5	0	0	0.4
		DL errors	0	0	0	0	0	0.2	0	0.3	0.1	0	0	0.1
		DL received	0	0	0	0	0.5	0.3	0	1	1.1	0	0.8	0.7
		Total	0	0	0	47.43	44.05	44.81	94.91	90.57	91.10	47.43	39.92	49.31
		Mean	0	0	0	47.43	41.22	41.97	47.45	38.61	38.27	47.43	39.92	38.91
		St. deviation	0	0	0	0	2.69	2.70	0.05	10.73	8.44	0	0	1.90
	All-1 Fragments	Sent	1	1.95	3.3	1	2.2	3.9	1	2.4	5.5	1	3.1	6.4
		UL errors	0	0.1	0.7	0	0	0.9	0	0.1	1.1	0	0.2	1.2
		DL errors	0	0.2	0.7	0	0.3	0.5	0	0.1	1.1	0	0.5	1
		DL received	1	1.7	1.9	1	1.9	2.5	303.5	2.2	3.3	1	2.4	4.2
		Total	38.58	81.88	144.64	39.03	88.79	175.93	0	96.89	242.38	37.89	126.69	266.41
		Mean	38.58	40.81	50.97	39.03	40.37	41.08	0	40.24	42.39	37.89	40.38	41.34
		St. deviation	0	1.93	3.06	0	1.02	2.98	0	1.92	4.23	0	2.24	3.72
	Total duration (no duty cycle)		94.7	146.88	209.64	198.99	265.67	361.03	303.51	380.13	553.28	554.54	698.82	912.22
	% Increase from no error		0.00%	55.10%	121.40%	0.00%	33.51%	81.43%	0.00%	25.24%	82.29%	0.00%	26.02%	64.50%
	Total Duration (duty cycle)		43200	43200	43200	64800	64800	86400	86400	108000	129600	194400	237600	280800
Total UL errors			0	1	2	0	2.3	4	0	3.1	7.1	0	7.1	16.1
Total UL errors %			0.00%	11.24%	18.87%	0.00%	13.14%	20.00%	0.00%	12.20%	22.54%	0.00%	11.64%	22.27%
Total DL errors			0	0.2	1	0	0.3	0.7	0	0.4	1.2	0	0.5	1.1
Total DL errors %			0.00%	10.53%	26.92%	0.00%	11.11%	20.00%	0.00%	11.11%	21.43%	0.00%	13.51%	18.33%
Network messages exchanged		UL	7	8.9	10.6	14	17.5	20	21	25.4	31.5	52	61	72.3
		DL	1	1.9	2.6	1	2.7	3.5	1	3.6	5.6	1	3.7	6

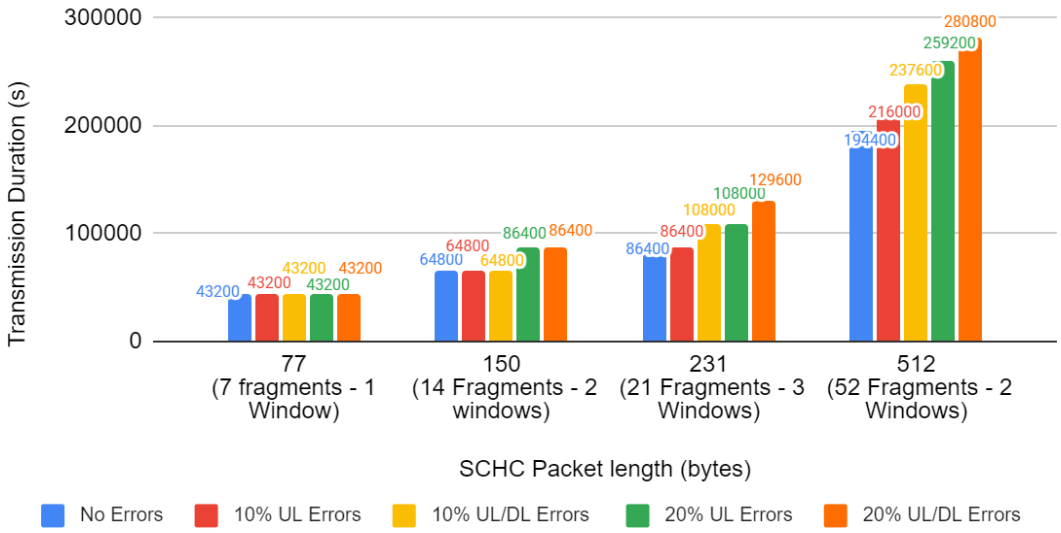

Figure 3.14: Random error transmission duration (considering duty cycle)

In Figure 3.14, we observe the time it would take including the sleep time for the duty cycle. The behavior is the same as without duty cycle. The main differences are that, of course, the total duration is much longer, and that the figures are more smoothened. For example, for the case of 77 fragments, the errors do not have any effect on the total time. That happens because, although more fragments need to be sent, they are not enough to trigger an additional hour of sleep in order to respect the duty cycle.

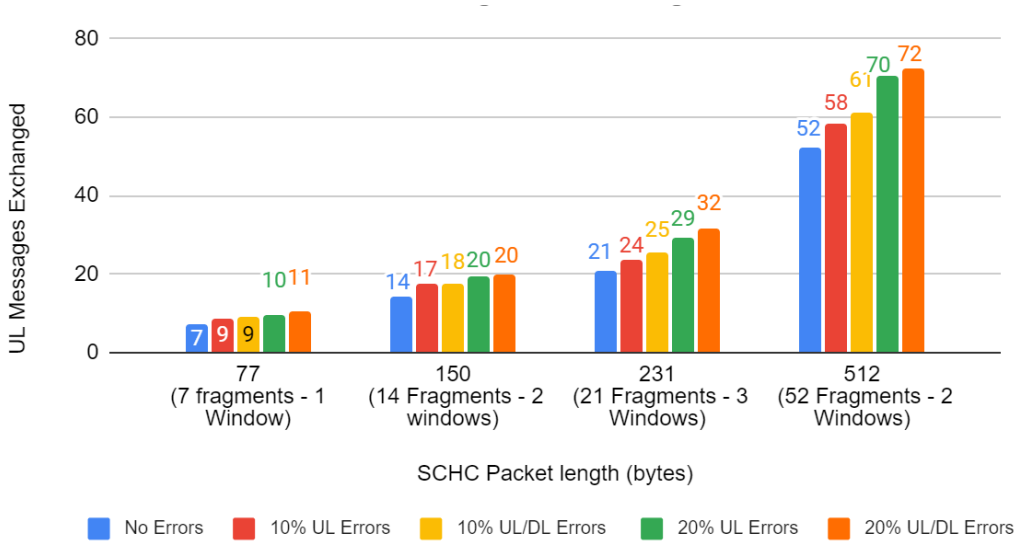

Figure 3.15: Random error UL messages

Figure 3.15 shows the amount of UL messages sent. An increase is observed as the error probability increases. In addition, DL errors also cause additional UL messages. That happens because after a lost DL message which was a response to an All-1 message, the device needs to resend an All-1 message.

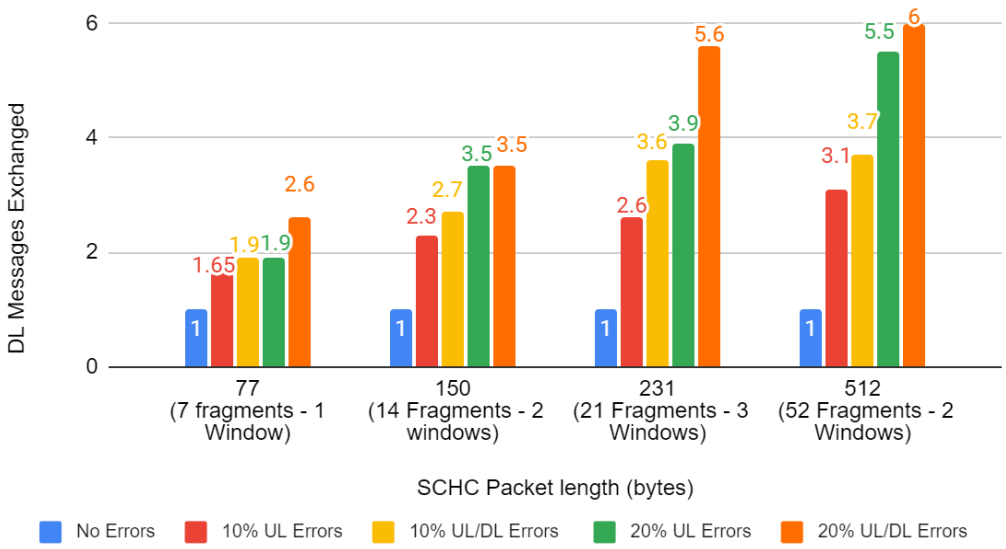

Figure 3.16: Random error DL messages

In Figure 3.16, the DL messages sent are observed. Note that those messages do not include only the DL message correctly received from the device, but also the ones lost. Again, for a bigger error probability, there are more DL messages sent. Another observation is the similarity between the cases of 150 and 512 bytes. That could be explained by the fact that they have the same number of windows. Finally, as expected, the larger the packet size is, the more DL messages need to be sent. That is explained by the fact that for larger packet sizes, there are more errors, which cause more All-1 messages to be sent (and therefore, more DL ACK responses) until all fragments are correctly received.

4. Energy Evaluation of SCHC Packet transmission

This section will focus on evaluating the energy performance of sending SCHC Packets using the LoPy4 device. In the first subsection, the energy consumption of each SCHC Fragment type will be analysed. In the second subsection, the energy consumption of sending a whole SCHC Packet will be studied, while in the last one, several graphs will be shown regarding the energy performance.

An important assumption that all the next subsections follow is the device, in order to respect the duty cycle restrictions, sleeps for the rest of the hour after sending 6 SCHC Fragments. In addition, the active part of sending a SCHC Packet is defined as the whole time needed to successfully send a SCHC Packet, including the actual transmission time and the intermediate sleep time. Another assumption is that the last fragment of the SCHC Packet sent is a full fragment (12 bytes). Finally, a last assumption is that the SCHC Packet transmission is errorless.

4.1 Energy Consumption of Regular, All-0 & All-1 SCHC Fragments

This subsection will focus on analysing the energy consumption of the different SCHC Fragment types, which will be the basis of calculating the total energy consumption of a SCHC Packet.

LoPy4 works at 3.3V, so that would be used as the reference operating voltage in all calculations. In addition, the time intervals of each phase of a transmission is based on the analysis done in section 3.1. In Table 4.2, the analysis is done per fragment type. The average current consumption, the power consumption and the energy consumption is calculated.

4.2 Energy Consumption of a SCHC Packet

This section intends to build a mathematical model of the power consumption of errorless SCHC Packet transmissions. According to the calculations made in section 4.1, we conclude in Table 4.1 that for each fragment type the current consumption is:

Table 4.1. Current consumption and transmission duration per fragment type

Fragment type	Transmission duration (s)	Average current consumption Iavg (mA)
Regular Fragment	T_{Reg}= 9.24	77.18
All-0 Fragment	T_{All0} = 47.75	48.83
All-1 Fragment	T_{All1} = 40.05	50.91

Table 4.2. Average consumption per fragment type

Regular Fragment						
Phase	**Occurrences**	**Duration (ms)**	**Current consumption (mA)**	**Average current consumption Iact (mA)**	**Power consumption (mW)**	**Energy consumption (mJ)**
Transmission	3	2080	97.8	**77.18**	322.74	671.30
Wait next transmission	2	1000	34.3		113.19	113.19
Cooldown	1	1000	34.3		113.19	113.19
Total	-	**9240**	-		-	**2353.47**
All-0 Fragment						
Phase	**Occurrences**	**Duration (ms)**	**Current consumption (mA)**	**Average current consumption Iact (mA)**	**Power consumption (mW)**	**Energy consumption (mJ)**
Transmission	3	2080	97.8	**48.83**	322.74	671.30
Wait next transmission	2	475	34.3		113.19	53.77
Wait reception	1	15556	34.3		113.19	1760.78
Reception	1	25000	46.2		152.46	3811.5
Total	-	**47746**	-		-	**7693.71**
All-1 Fragment (Full - 12 bytes)						
Phase	**Occurrences**	**Duration**	**Current consumption (mA)**	**Average current consumption Iact (mA)**	**Power consumption (mW)**	**Energy consumption (mJ)**
Transmission	3	2080	97.8	**50.91**	322.74	671.30
Wait next transmission	2	475	34.3		113.19	53.77
Wait reception	1	15556	34.3		113.19	1760.78
Reception	1	14500	46.2		152.46	2210.67
Confirmation (ACK)	1	1799	87.9		290.07	521.84
Cooldown	1	1000	34.3		113.19	113.19
Total	-	**40045**	-		-	**6727.91**

Each packet (of a given size Bytes, a given number of total fragments Frag and a given number of windows Win) consists of the following fragment transmissions:

- 1 All-1 Fragment
- (Win - 1) All-0 Fragments
- Frag - Win Regular Fragments

In addition, for each 6 fragments sent, the device needs to sleep for the rest of the hour in order to respect the duty cycle set in RCZ1. The sleep current consumption I_{Sleep} would be 40 uA. That would mean that the total transmission time T_{Total} will be *ceil(Frag/6) hours,* where ceil stands for the ceiling function.

The total average current consumption I_{act} of the SCHC Packet will be:

$$I_{act}\ (mA) = \frac{I_{avg-Reg} * T_{Total-Reg} + I_{avg-All0} * T_{Total-All0} + I_{avg-All1} * 1 * T_{All1} + T_{DC-Sleep} * I_{Sleep}}{T_{Total}} \quad (4.1)$$

where T_{Total} is the total transmission duration of the packet and $T_{Total\text{-}Reg}$, $T_{Total\text{-}All0}$, $T_{Total\text{-}All1}$, $T_{DC\text{-}Sleep}$ are the respective total durations of each fragment type and the sleep time. If we analyse that more:

$$I_{act}\ (mA) = \\ = \frac{713.143 * (Frag - Win) + 2331.193 * (Win - 1) + 2038.691 + 0.04 * (ceil(Frag/6) - (Frag - Win) * 9.24 + (Win - 1) * 47.746 + 40.045)}{ceil(Frag/6) * 3600} \quad (4.2)$$

Frag and *Win* depend on the SHCH Packet size category. More specifically, for each SCHC Packet category they will be:

a) Packets of size smaller than 300 bytes

For this category, the number of fragments and windows is:
Frag = ceil(Bytes/11)
Win = ceil(Frag/7)=ceil(ceil(Bytes/11)/7)

b) Packets of size larger or equal to 300

For this case, the number of fragments and windows is:
Frag = ceil(Bytes/10)
Win = ceil(Frag/31)=ceil(ceil(Bytes/10)/31)

Figure 4.1 presents the average current consumption of all different SCHC Packet sizes from 11 to 2250 bytes:

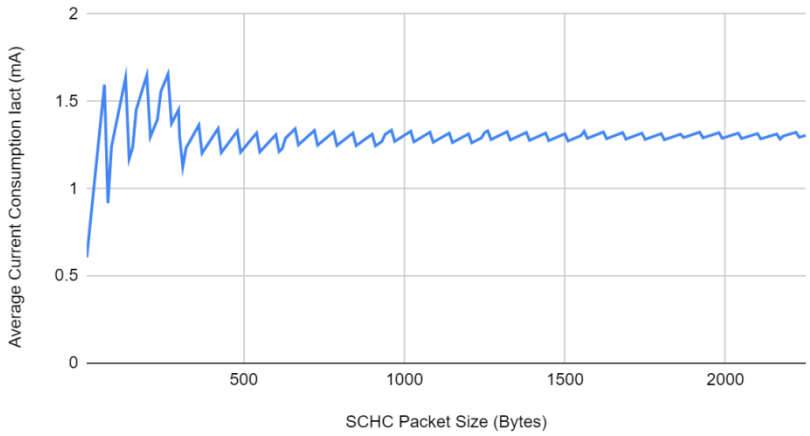

Figure 4.1: Average current consumption I_{act} of different SCHC Packet sizes

The average current consumption I_{Act} follows Equation (4.2). As seen, the average current consumption oscillates around 1.25 mA. The minimum value is 0.606 mA and happens for the smallest SCHC Packet size of 11 bytes, while the maximum value is 1.657 and happens for a SCHC Packet size of 264 bytes.

Figure 4.2 shows the energy consumption for the different SCHC Packet sizes. Equation (4.3) describes the energy consumption:

$$E = I_{act} * V * T_{Total} = I_{act} * T_{Total} * 3.3 \quad (4.3)$$

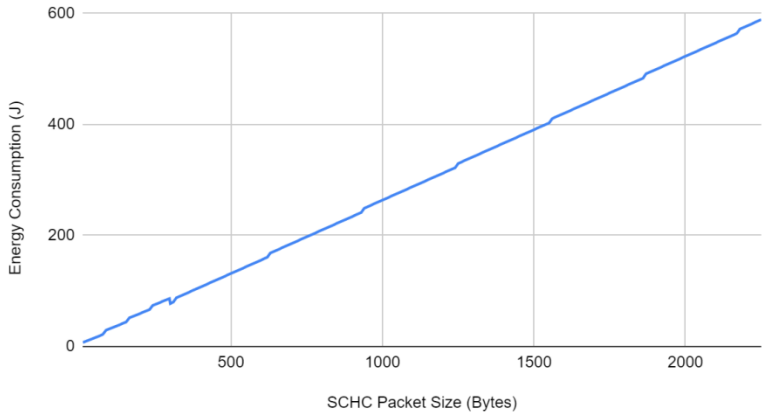

Figure 4.2: Energy consumption of different SCHC Packet sizes

The energy consumption is quite linear, as expected. The energy consumption increases as the packet size increases. That is normal, as the total time needed to send the SCHC Packet increases accordingly. The energy consumption for the smallest SCHC Packet of 11 bytes is 7.198 J, while the consumption of the biggest packet of 2250 bytes is 589.003 J.

4.3 Energy Performance metrics

In this final subsection, three energy performance metrics will be analysed. Before explaining in detail the metrics, the term of period T will be introduced. Period T represents the period of transmitting the SCHC Packet. In other words,

a period T means that the device should start the transmission of a SCHC Packet (by sending the first fragment) every time T. As expected, time T should be greater or equal to the time T_{Total} needed to send the SCHC Packet. T_{Total} includes as well the sleep time needed in order to respect the duty cycle. So, T would be the sum of two terms, $T_{Total,}$ and the waiting time until the SCHC Packet is sent again, denoted as T_{Wait}. During T_{Wait}, the device will be sleeping. Summing up the above:

$$T = T_{Total} + T_{Wait} \tag{4.4}$$

The three metrics analysed are the average current consumption I_{Avg}, the Energy Cost and the Device Lifetime for a period T. The device lifetime would depend on the battery used to power the device. For the sake of the analysis, the battery chosen was a 3.7V battery having a capacity of 2000 mAh. It is important not to confuse the 3.7V of the battery with the operating voltage of 3.3 V of the LoPy. The device has an internal voltage regulator that regulates the input voltage, that should be in the range of 3.5-5.5 V, to 3.3V. All metrics were analysed for 5 different SCHC Packet sizes for the sake of the comparison: 77 bytes, 154 bytes, 275 bytes, 510 bytes and 2250 bytes. Finally, the analysis was done for a period T ranging from the minimum T_{Total} needed to send the SCHC Packet, up to 7200 minutes (5 days). Figure 4.3 - 4.5 below present the results for the metrics mentioned above:

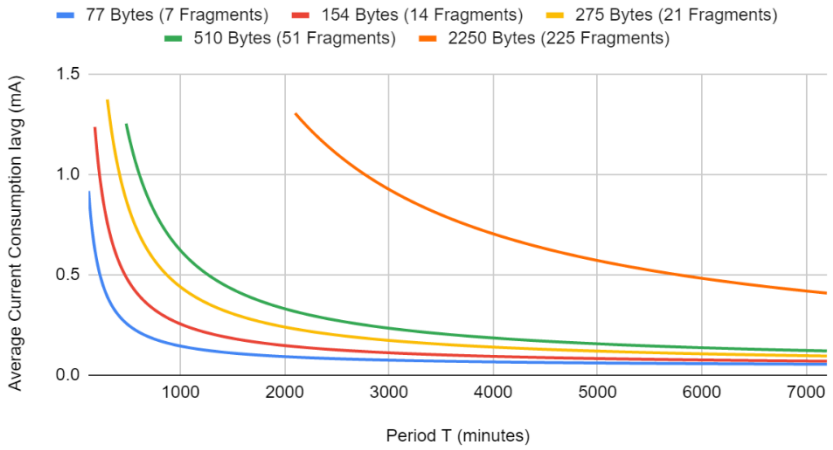

Figure 4.3: Average current consumption I_{avg} vs T

Looking at Figure 4.3, the first important observation is that not all curves are starting from the same point. That happens because period T should be at least the time T_{Total} needed to send the SCHC Packet. Of course, for each packet size, this time is different, and that is why the starting point is different at each curve. In addition, Figure 4.3 shows that as period T increases, the average consumption drops for all SCHC Packet sizes. That is explained by the fact that as period T increases, the sleep time also increases, and during sleep time the current consumption is only 40 uA, making the average consumption lower. Finally, it can be observed that for the small SCHC Packet sizes the average current consumption ends up pretty much at the same point, which is around 0.1 mA. The bigger SCHC Packet is still in a higher value, around 0.4 mA. Eventually, if the Period T is enlarged a lot more, they will all end up in the same value, when the sleep current would dominate the average current.

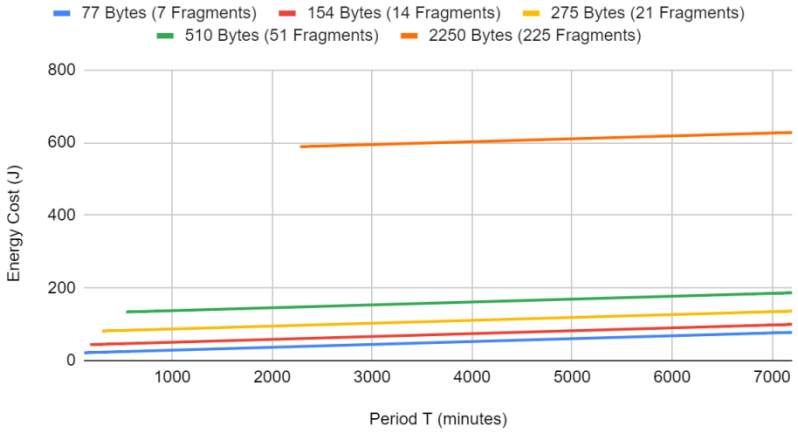

Figure 4.4: Energy cost vs T

Figure 4.4 shows how the energy cost evolves as the period T increases. In that figure, the curves are parallel. As Period T increases, the energy cost of course increases, but with a very small inclination, as the device is in sleep mode and consumes only a small amount of energy.

In Figure 4.5 the LoPy4 device lifetime is shown, when transmitting SCHC Packet of various sizes. The battery used for the analysis has a capacity of 2000 mAh. For a small Period T, the device lifetime is also small, ranging from 63 days for the SCHC Packet size of 2250 bytes, up to 90 days for the smallest packet of 77 bytes. It is normal that, as the SCHC Packet size increases, the device needs to send more SCHC Fragments, consumes more energy and therefore has a shorter lifetime for the same battery capacity. Then, as the period T increases, the device lifetime also increases accordingly. More specifically, the maximum lifetime, which corresponds to a period T of 5 days, goes up to 1525 days (more than 4 years) for the smallest packet of 77 bytes and up to 189 days for the largest packet of 2250 bytes. Finally, for the theoretical case of a packet size of 1 SCHC Fragment, having a period T tending to infinity, device lifetime tends to 5.7 years.

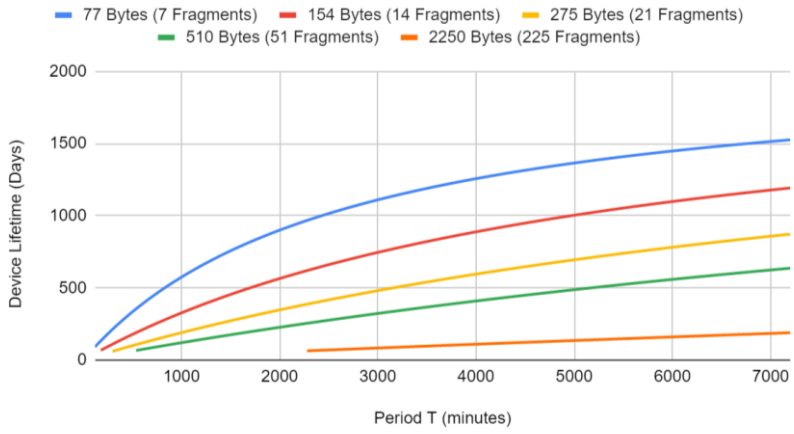

Figure 4.5: Device lifetime vs T

5. Conclusions and future work

In this chapter, the conlusions of the results presented in the previous chapters will be demonstrated. Finally, some proposals will be shown for future work that can be done in the same topic

5.1 Conclusions

The purpose of this Master Thesis was to evaluate the performance of the SCHC adaptation layer over Sigfox network. SCHC is used to enable transmitting larger packets, e.g. IPv6 packets, through a LPWAN network, that typically supports smaller packet sizes. More specifically, Sigfox network supports up to 12 bytes of payload, and in the Master Thesis packets up to 2250 bytes of payload were used for the evaluation.

The evaluation was focused on evaluating different metrics, as the transmission duration, the UL and DL messages sent and the energy consumption. The transmission duration evaluation showed that the time needed to errorlessly send a SCHC packet in the range of 0-2250 bytes, starts from 37 seconds and goes up to 38 minutes. In addition, taking into consideration the fact that in the RCZ1 zone there is a duty cycle restriction of 1%, where the device should sleep for the rest of the hour after sending 6 fragments, that duration would increase and range from 1 hour to 38 hours.

Adding a random error probability could significantly increase the transmission time needed. More specifically, an error probability of 20% could increase the transmission duration by approximately 70% more. That would happen as, apart from the fragment retransmissions, there are more All-1 messages generated that need a significant amount of time. Regarding UL and DL messages sent, an error probability of 20% would increase the UL messages by 40-60%, depending on the SCHC Packet size. The DL messages, which would be only 1 for an errorless transmission, could increase to up to 6 for a SCHC Packet size of 512 bytes.

Regarding the energy evaluation, the average current consumption in RCZ1 is approximately 1.25 mA, ranging from 0.606 for small packets to 1.657 mA for large packets. Finally, if a device is powered by a battery of a 2000 mAh capacity, it would have a lifetime in a range between 63 to 90 days, depending on the SCHC Packet size, if it sends the packet continuously, without any additional sleep time. On the other hand, if the SCHC Packet is sent every 5 days, the device lifetime would range from 1565 days for the smallest SCHC Packet of 11 bytes, to 189 days for the largest SCHC Packet size of 2250 bytes, which means that the SCHC Packet size would play a significant role to the device lifetime.

5.2 Future work

SCHC is a newly defined layer. That means that there is still a lot of work and testing that could be done in order to evaluate its performance. In this Master Thesis, only a specific part of the SCHC specification was analysed. The work was focused only on ACK-on-Error mode. However, there are two more modes, ACK-Always and No-ACK, to be evaluated.

In addition, as the work was done in RCZ1, the results are limited to the transmission durations of that zone. However, Sigfox has 7 diferent RCZ where SCHC can be tested and compared to the results in RCZ1.

Finally, the energy evaluation was limited to the case of errorless transmission. A model could be defined for cases where errors happen and parameters like energy cost or device lifetime can be foreseen.

REFERENCES

[1] Salim Chikhi, B. M. (2018). *Survey of Internet of Things Applications in Smart Agriculture_A typical architecture.Pdf*.

[2] Mekki, K., Bajic, E., Chaxel, F., & Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express*, *5*(1), 1–7. https://doi.org/10.1016/j.icte.2017.12.005

[3] Technical Marketing Workgroup. (2015). Overview LoRa and LoRaWAN. *LoRa Alliance*, *November*.

[4] Shuda, J. E., Rix, A. J., & Booysen, M. J. (2018). Towards Module-Level Performance and Health Monitoring of Solar PV Plants Using LoRa Wireless Sensor Networks. *2018 IEEE PES/IAS PowerAfrica, PowerAfrica 2018*, 172–177. https://doi.org/10.1109/PowerAfrica.2018.8521179

[5] Sinha, R. S., Wei, Y., & Hwang, S. H. (2017). A survey on LPWA technology: LoRa and NB-IoT. *ICT Express*, *3*(1), 14–21. https://doi.org/10.1016/j.icte.2017.03.004

[6] Liaut, A. (2020). *Sigfox connected objects: Radio specifications*. *February*. https://storage.googleapis.com/public-assets-xd-sigfox-production-338901379285/b2be6c79-4841-4811-b9ee-61060512ecf8.pdf

[7] Gomez, C., Veras, J. C., Vidal, R., Casals, L., & Paradells, J. (2019). A Sigfox energy consumption model. *Sensors (Switzerland)*, *19*(3). https://doi.org/10.3390/s19030681

[8] https://support.sigfox.com/docs/callbacks-documentation

[9] Minaburo, A., Toutain, L., Gomez, C., & Barthel, D. (2020). *SCHC: Generic Framework for Static Context Header Compression and Fragmentation*. *8724*. https://rfc-editor.org/rfc/rfc8724.txt%0Ahttps://www.rfc-editor.org/info/rfc8724

[10] Zúñiga, J.-C., Gomez, C., & Toutain, L. (2020). *SCHC over Sigfox LPWAN*. *Draft-ietf-lpwan-schc-over-sigfox-03*, 1–13. https://datatracker.ietf.org/doc/html/draft-ietf-lpwan-schc-over-sigfox-03

[11] https://docs.pycom.io/datasheets/development/lopy4/

[12] https://code.visualstudio.com/docs

[13] https://pycom.io/products/supported-networks/pymakr/

[14] Ergen, S. C. (2004). ZigBee/IEEE 802.15. 4 Summary. *UC Berkeley, September*, *10*, 17.

[15] Souza Oliveira, A., & Franco Pereira, M. (2015). Near Field Communication (NFC) Technology and Measurements. *Gestao e Producao*, *21*(3), 133–144.

[16] Gomez, C., Oller, J., & Paradells, J. (2012). Overview and evaluation of Bluetooth low energy: An emerging low-power wireless technology. *Sensors (Switzerland)*, *12*(9), 11734–11753. https://doi.org/10.3390/s120911734

[17] Oliveira, L. M. L., De Sousa, A. F., & Rodrigues, J. J. P. C. (2011). Routing and mobility approaches in IPv6 over LoWPAN mesh networks. *International Journal of Communication Systems*, *24*(11), 1445–1466. https://doi.org/10.1002/dac.1228

[18] Garg, R., & Sharma, S. (2017). A study on Need of Adaptation Layer in 6LoWPAN Protocol Stack. *International Journal of Wireless and Microwave Technologies*, *7*(3), 49–57. https://doi.org/10.5815/ijwmt.2017.03.05

[19] Gomez, C., Paradells, J., Bormann, C., & Crowcroft, J. (2017). From 6LoWPAN to 6Lo: Expanding the Universe of IPv6-Supported Technologies for the Internet of Things. *IEEE Communications Magazine*, *55*(12), 148–155. https://doi.org/10.1109/MCOM.2017.1600534

ACRONYMS

AODV	Ad-Hoc On Demand Vector Routing
AppIID	Application Interface Identifier
BPSK	Binary Phase Shift Keying
CSS	Chirp Spread Spectrum
D-BPSK	Differential Binary Phase Shift Keying
DevIID	Device Interface Identifier
DL	Downlink
DSR	Dynamic Source Routing
DSSS	Direct-Sequence Spread Spectrum
EPS	Evolved Packet System
FDD	Full Function Device
FH	Frequency Hopping
GCP	Google Cloud Platform
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol Version 6
L2	Layer 2
LPWAN	Low Power Wide Area Network
M2M	Machine to Machine
MAUTH	Message Authentication
NFC	Near Field Communication
OFDM	Orthogonal Frequency Division Multiplexing
PAN	Personal Area Network
QPSK	Quadrature Phase Shift Keying
RCS	Reassembly Check Sequence
RDD	Reduced Function Device
SF	Spreading Factor
TDMA	Time-Division Multiple Access
UL	Uplink
WPAN	Wireless Personal Area Network

Annex 1: WPAN Technologies

WPANs are networks that, as reflected in their name, are restricted to the personal area of an individual. The first standards for WPAN appeared around the 2000's and have been growing since. Their main characteristics are:

- Low power consumption
- Short range
- Small transmission latency
- Small transmission bandwidth
- No radio subscription costs

There are a lot of technologies implemented in the WPAN environment and below I will explain the main aspects for some of those.

IEEE 802.15.4 / ZigBee

IEEE defined, among others, the 802.15.4 standard for the physical and the MAC layer. It is a wireless standard and was firstly introduced in 2003. Since then, it underwent a lot of changes and improvements in order to adapt to newer requirements. Zigbee alliance made a partnership with IEEE in order to provide the full protocol stack, being responsible for the upper layers, from network to application layer.

As explained in [14], the devices in a IEEE 802.15.4 network are divided into two categories: Reduced function devices (RDDs) and Full Functional Devices (FDDs). The RDDs have reduced functionality and are end-devices that usually have the role of taking a physical measurement once a while and sending it through their wireless interface. That means that the majority of time it is not necessary for them to be functional, so they are sleeping in order to save power. On the other hand, FDDs have more functionalities, can be mains powered, and have the role of either a device, or a coordinator, or a PAN coordinator, which is the device responsible for coordinating the whole PAN and could also be an interface to an external network.

There are mainly two different topologies in IEEE 802.15.4: star topology and cluster-tree topology. The first one can be used for smaller areas, while the second one can be used to extend the coverage area of the network. In Figure A1.1 below we can observe visually those two topologies:

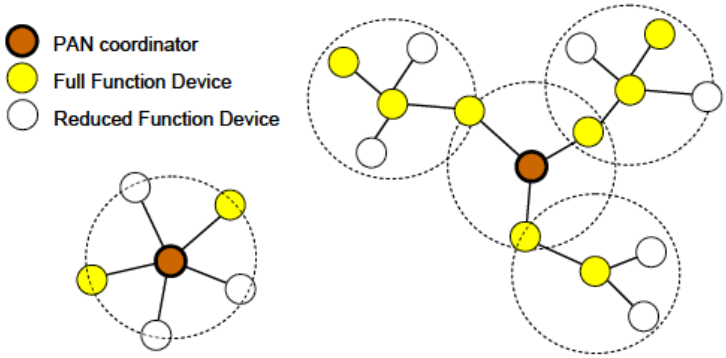

Figure A1.1: Network topologies of 802.15.4 [14]

In the star topology, all devices send their data to the PAN coordinator, while in the cluster-tree we have a peer-to-peer network, where end-devices send their data and the coordinators have the role of routers towards the PAN coordinator.

IEEE 802.15.4 is a wireless communication protocol, so it uses radio frequencies to operate on the physical layer. More specifically, it uses 3 different frequency bands, one for America at 915 MHz, one for Europe at 868 MHz, and one international in the 2,4 GHz ISM band. The data rates provided are of tens of kbps, depending on the frequency band, starting from 20kbps at 868 MHz, 40 kbps at 915 and reaching up to 250 kbps at 2,4 GHz. All the different bands use a modulation based on DSSS, being BPSK for the first two bands and OQPSK for the 2,4 GHz.

Regarding the channels, there is only 1 channel at the 868 band, 10 channels at the 915 band and 16 channels in the 2,4 GHz band. This is also visualized in Figure A1.2:

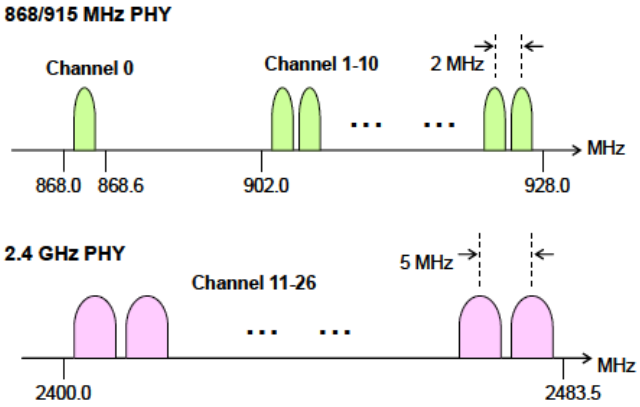

Figure A1.2: Channels of IEEE 802.15.4 [14]

The MAC layer of 802.15.4 is responsible for the channel allocation, the beacons management, association, acknowledgments, synchronization and other mechanisms.

The MAC layer offers an optional superframe structure. This superframe is sent by the coordinator and is used to synchronize the devices, as it is initiated by a beacon frame. After that beacon frame there is a contention access period (CAP) and a contention free period (CFP). During CAP, the network access is performed through the CSMA-CA algorithm, during which devices are competing to access the medium in 10 consecutive slots. During CFP, the slots are guaranteed for some devices, which usually happens for applications where it is more critical to send the data, e.g. alarm. In Figure A1.3 below the structure of the superframe is presented:

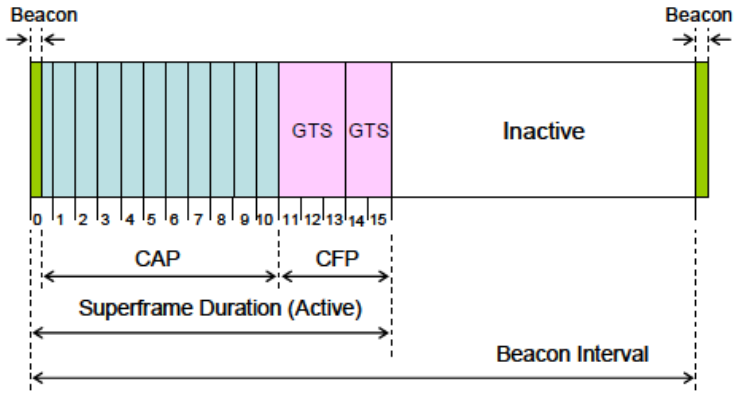

Figure A1.3: IEEE 802.15.4 Superframe structure [14]

Bluetooth Low Energy

The Bluetooth Low Energy protocol, also known as BLE, was part of the specification of Bluetooth 4.0 that was introduced in 2010. It is based on the original Bluetooth protocol, but is more targeted to low power and low range devices. The most usual use cases that BLE targets are health, sports, wellness and home automation. One of the biggest advantages that BLE has compared to similar technologies is that the majority of smartphones support the Bluetooth technology and the phone can be used as the coordinator/ master of the network, while other devices, like health & wellness bands, can be end-devices/ slaves taking measurements and transmitting the data. Initially, the only topology that BLE supported was the star topology, like the one explained in the previous example, but now the mesh topology is also supported.

Regarding the protocol stack, as explained in [15], Bluetooth specifies the full protocol stack, dividing the stack into 3 parts, the physical and link layer which define the controller, while the other layers are defined in the host part. They will not be analysed in details, but an overview of the stack is presented in Figure A1.4:

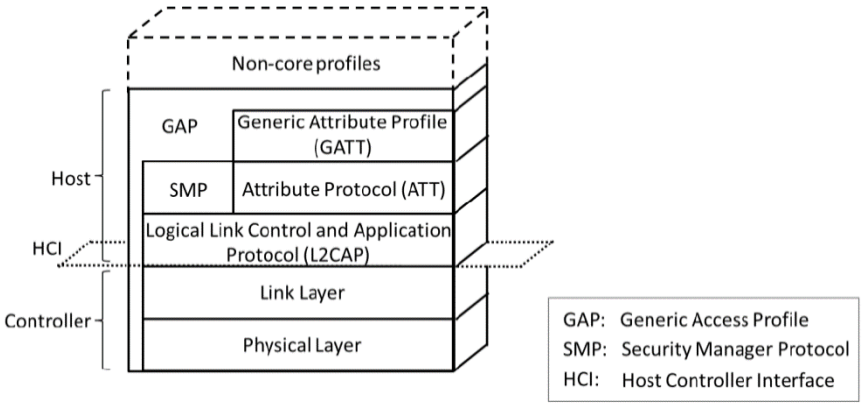

Figure A1.4: Protocol stack of Bluetooth Low Energy [15]

The physical layer functions on the 2,4 GHz ISM band, having 40 channels that the devices can access. Additionally, frequency hopping, which is a technique to change the channel in order to avoid stucking in a noisy channel, is supported. The data rates can reach up to 1 Mbps, a value quite high compared to similar technologies.

The medium access is done by TDMA, which means that the devices are transmitting one-by-one, coordinated by their master. One of the advantages of BLE is that it supports fragmentation and reassembly of packets, which means that it can fragment big packets into smaller fragments so that the physical layer would be able to send them, while at the receiver the inverse mechanism takes place.

Z-Wave

Another wireless technology used mainly for home automation is Z-wave. It was originally created in 1999. Regarding its protocol stack, the first 2 layers, physical and MAC, are defined by ITU-T G.9959 standard. They are operating in the sub-1 GHz band, starting from 865 MHz and going until 926 MHz, depending on the region. The data rates are also ranging from 9,6 kbps up to 100 kbps at maximum.

The topology is mesh, working in an ad-hoc way, so that intermediate devices are forwarding the frames to the next step until it reaches its final destination. That requires a routing protocol, which in the case of Z-wave could be AODV or Dynamic Source Routing DSR.

Regarding the MAC layer, the CSMA/CA algorithm is used as the medium access technique, like the case of IEEE 802.15.4. Finally, Z-wave is another technology that supports fragmentation and reassembly of packets.

Near Field Communication

Near Field Communication (NFC) is another wireless technology that is present nowadays. Its main characteristic is the very short range that supports, which is

less than 10 cm. It is used mostly on secure applications, as the short distance of communication gives the advantage of security towards potential hackers. For example, it is used for card contactless payments of credit cards or a payment through the smartphone, as more and more smartphones support the NFC technology. The wireless transmission is based on the electromagnetic field created by the sending device, that triggers the listening device, as shown in Figure A1.5:

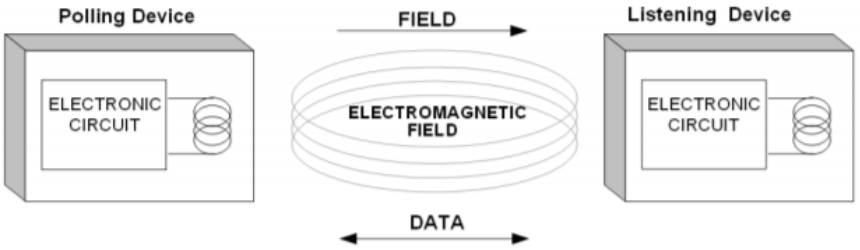

Figure A1.5: Data transmission of NFC [16]

The data rates that NFC supports are from 106 to 424 kbps and operates at frequencies centered at 13.56 MHz. Fragmentation and Reassembly is also supported by NFC technology.

Annex 2: 6LoWPAN/6Lo adaptation Layers

The first adaptation layer presented is the Ipv6 over LoWPAN or commonly known as 6LoWPAN. This adaptation layer was created to be able to match the 802.15.4 MAC and PHY layers to the upper layer protocols defined by IETF. An illustration of the protocol stack can be observed in Figure A2.1:

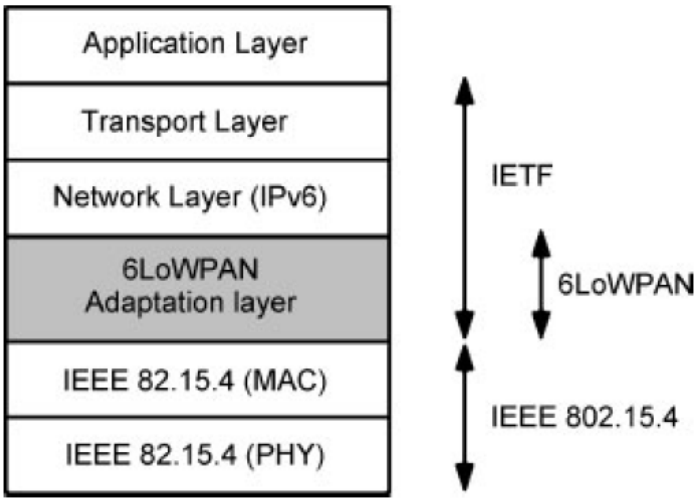

Figure A2.1: 6LoWPAN protocol stack [17]

As explained in [18], one of the mechanisms that 6LoWPAN implements is header compression. An original Ipv6 Header is 40 bytes long, which is unacceptable for the 15.4 packet size. 6LoWPAN uses two header compression types: HC1 and HC2. HC1 is used to reduce the Ipv6 header to 3 bytes and HC2 is used to reduce the size of the TCP layer header.

Another important mechanism is the support for fragmentation and reassembly. This consists of fragmenting an Ipv6 payload to smaller payload so that they will be able to fit in a 15.4 MAC fragment payload. On the other side, after the successful transmission of all payloads, the original Ipv6 payload will be reconstructed.

6LoWPAN is also in charge of handling the routing, when it is needed. There are various protocols for routing, like 6LoWPAN Ad-hoc On-Demand Distance Vector (LOAD), Multipath based 6LoWPAN Ad-hoc On-Demand Distance Vector (MLOAD), Dynamic MANET On-Demand for 6LoWPAN Routing (DYMO-Low), Hierarchical Routing (Hi-Low) and Extended Hi-Low.

Finally, 6LoWPAN handles the addressing and neighbor discovery. The addressing is mainly based on the EUI-64 address of a 15.4 device. In addition, the neighbor discovery is used so that a device can identify its neighbors. It uses prefix discovery and route configuration, while also performing address resolution and duplicate address detection.

Following the same principles, there are several other adaptation layers between Ipv6 and the underlying wireless technologies. The generic format of

6LoWPAN is called 6lo. Some examples of other technologies that are compatible to Ipv6 due to 6lo are Bluetooth Low Energy, PLC, NFC, ITU-T G. 9959, DECT-ULE, MS/TP, IEEE 1901.2 and IEEE 802.11ah. More info can be found on [19].