



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

IMPLEMENTATION AND EVALUATION OF STATIC CONTEXT HEADER
COMPRESSION FOR IPV6 PACKETS WITHIN A LORAWAN NETWORK

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

NICOLÁS ANDRÉS MATURANA ARANEDA

PROFESOR GUÍA:
SANDRA CÉSPEDES UMAÑA

MIEMBROS DE LA COMISIÓN:
CLAUDIO ESTÉVEZ MONTERO
CÉSAR AZURDIA MEZA

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: NICOLÁS ANDRÉS MATURANA ARANEDA
FECHA: 2019
PROF. GUÍA: SANDRA CÉSPEDES UMAÑA

IMPLEMENTATION AND EVALUATION OF STATIC CONTEXT HEADER COMPRESSION FOR IPV6 PACKETS WITHIN A LORAWAN NETWORK

The Internet of Things (IoT) is a new communication paradigm, currently blooming and spreading, that proposes the interconnection of common everyday objects and all kinds of conventional devices with the Internet. At the same time, the enormous amount of nodes that is expected to get connected to the Internet demands a massive implementation of Internet Protocol version 6 (IPv6). IoT aims for the development of new applications, and thus has promoted the creation of new device classes and new network architectures.

Low Power Wide Area Networks (LPWAN) have recently arisen as a natural evolution of the concept of Wireless Sensor Networks (WSN). In the light of IoT, LPWAN networks open a new field of development, which is mainly focused on monitoring-like services that are carried out over wide areas and which do not require big data transfer rates. LPWAN devices are characterized by their low power consumption and low cost, thus allowing their massive deployment over long periods of time without the need for battery replacement or recharge.

Long Range Wide Area Network (LoRaWAN) is one of the first and principal LPWAN technologies, whose great flexibility makes it ideal for self-designed networks. In the continent of America, it utilizes the Industrial, Scientific and Medical (ISM) frequency band around 915 MHz. However, there are several other LPWAN technologies with distinct proprietary architectures and communication protocols, which hinders the interoperability desired within the IoT environment.

The Internet Engineering Task Force (IETF) working group for the implementation of IPv6 over LPWAN networks (lpwan WG) is currently developing a compression and fragmentation mechanism for IPv6 packet transmission over LPWAN networks called Static Context Header Compression (SCHC). The compression scheme for SCHC is already complete, but has not yet been officially implemented nor evaluated.

In this work, the author presents an experimental platform for the implementation and evaluation of the SCHC mechanism over a LoRaWAN network composed of a Microchip terminal node and an Everynet Radio Gateway (RG). The development process has involved the integration of multiple and diverse resources from the Telecommunications and Information and Communication Technologies (ICT) fields.

The created platform achieves a basic but successful implementation of SCHC's compression scheme. By means of this platform a preliminary evaluation of the functioning of SCHC was carried out, analyzing the level of compression attained by the mechanism for three communication contexts that are representative of LPWAN networks. The obtained results are positive.

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: NICOLÁS ANDRÉS MATURANA ARANEDA
FECHA: 2019
PROF. GUÍA: SANDRA CÉSPEDES UMAÑA

IMPLEMENTATION AND EVALUATION OF STATIC CONTEXT HEADER COMPRESSION FOR IPV6 PACKETS WITHIN A LORAWAN NETWORK

El paradigma de comunicación Internet of Things (IoT), el cual plantea la posibilidad de interconectar objetos cotidianos y toda clase de dispositivos convencionales a Internet, está actualmente en pleno desarrollo. El gran número de nodos que se espera conectar a Internet exige a su vez la implementación a gran escala de Internet Protocol version 6 (IPv6). IoT busca el desarrollo de nuevas aplicaciones y ha impulsado la creación de nuevas arquitecturas de red y nuevas clases de dispositivos.

Las redes Low Power Wide Area Networks (LPWAN) han surgido recientemente como una evolución natural del concepto Wireless Sensor Networks (WSN), redes de sensores interconectadas. A la luz del IoT, las nuevas redes LPWAN abren un nuevo campo de desarrollo, principalmente enfocado en servicios de monitoreo y afines que se desarrollen en áreas amplias y no requieran grandes tasas de transferencia. Los dispositivos LPWAN se caracterizan por ser de bajo consumo energético y de bajo costo, facilitando su despliegue masivo por largos períodos sin necesidad de recargar sus baterías.

Long Range Wide Area Network (LoRaWAN) es una de las primeras y principales tecnologías LPWAN, y presenta una gran flexibilidad que la hace ideal para redes de diseño propio. En América funciona en la banda industrial, científica y médica (ISM) alrededor de los 915 MHz. Sin embargo, también existen muchas otras tecnologías LPWAN con arquitecturas y protocolos propietarios, lo que dificulta alcanzar la interoperabilidad que se desea en el entorno IoT.

El grupo de trabajo para la implementación de IPv6 sobre redes LPWAN (lpwan WG) perteneciente al Internet Engineering Task Force (IETF) se encuentra actualmente desarrollando un mecanismo de compresión y fragmentación de paquetes IPv6 para redes LPWAN denominado Static Context Header Compression (SCHC). El esquema de compresión se encuentra terminado, pero aún no ha sido implementado ni evaluado de manera oficial.

En este trabajo se presenta una plataforma experimental para la implementación y evaluación del mecanismo SCHC sobre una red LoRaWAN consistente en un nodo terminal Microchip y un Radio Gateway (RG) de Everynet. En su desarrollo se han integrado múltiples y diversas herramientas del campo de las Telecomunicaciones y las Tecnologías de Información y Comunicación (ICT).

La plataforma creada logra una implementación básica pero exitosa del esquema de compresión de SCHC. Por medio de ella se ha llevado a cabo una evaluación preliminar del funcionamiento de SCHC, analizando el nivel de compresión logrado por el mecanismo para tres contextos de comunicación característicos de una red LPWAN. Los resultados obtenidos son positivos.

A Marcelo, Angélica, Carmengló, Mario, Cata, Mati, Pablo, los Papis, el Tote, la Ceci, la Jesu y a la Maru.

A la profe Sandra, por su paciencia y su confianza en mí.

A mi papá, que me impulsó constantemente a terminar este camino.

A mi mamá, que siempre ha estado ahí para escucharme cuando la necesito.

A mi hermana, que tantas veces me apoyó y me ayudó a levantarme cuando mi ánimo decayó.

*A todos los champis, por su cariño y apañe incondicional durante todos estos años.
Habría sido imposible sin ustedes.*

A los que formaron parte de CAS, con quienes hicimos algunas cosas geniales que siempre recordaré con alegría.

A Matus, Jaime Aranda y Mati Macaya por toda su ayuda, las veces que me acompañaron y las aventuras que compartimos.

Al Isma, el Simón y a la Dani por la amistad inmortal, a pesar del tiempo y la distancia.

To my dear friends Dani, Martín and Fran, with whom I have shared many laughs and conversations.

A la gente de la casita, por sus comidas, sus risas, su cariño y sus palabras de ánimo en los momentos difíciles.

*A la Pame, que ha visto de cerca todo este proceso y nunca me ha dejado rendirme.
Gracias por ser mi principal apoyo estos últimos años.*

Agradecimientos

Agradezco a José Ignacio Guerra Gómez de Telefónica I+D por proporcionarme los equipos necesarios para desarrollar este trabajo, y por su disposición para facilitarme la información necesaria. También quiero agradecer a los miembros del WiNet Group por las sesiones de feed-back y el conocimiento que compartieron conmigo, sus aportes fueron muy útiles para resolver los problemas que surgieron en el proceso.

Contents

Introduction	1
Motivation and Context	1
Problem Statement	3
Scope	3
Objectives	4
Overall Objective	4
Specific Objectives	4
Methodology	4
Thesis Organization	5
1 Background and State of the Art	6
1.1 Internet of Things	6
1.2 LPWAN Technologies	8
1.3 LoRaWAN	9
1.3.1 LoRa	9
1.3.2 LoRaWAN protocol	11
1.3.3 Message Format	12
1.3.4 Datarate	14
1.4 IPv6	14
1.5 Adaptations of IPv6 over Constrained Networks	16
1.5.1 6LoWPAN	16
1.5.2 6LoRaWAN	16
2 Static Context Header Compression	18
2.1 SCHC Overview	18
2.2 Rules and Context	19
2.3 Rule Format	21
2.4 Packet Processing	21
2.4.1 Matching Operators (MOs)	23
2.4.2 Compression/Decompression Actions (CDAs)	23
2.4.3 Rule Selection	24
2.4.4 Padding	25
3 System Implementation	26
3.1 Equipment and Setup	26
3.1.1 Microchip RN2903 LoRa Technology Mote	26

3.1.2	Everynet LoRaWAN Gateway	29
3.1.3	Everynet Network Management Platform	30
3.1.4	Host Computer and Development Tools	30
3.1.5	Required Configuration	30
3.2	Functional Blocks	32
3.2.1	System Overview	33
3.2.2	IPv6 Packet Generator	34
3.2.3	SCHC Compressor	35
3.2.4	Sender Device Input	36
3.2.5	LoRaWAN Device	37
3.2.6	LoRaWAN Gateway	37
3.2.7	Packet Sniffer	38
3.2.8	LoRaWAN Payload Retriever	39
3.2.9	SCHC Decompressor	39
3.2.10	Message Verifier	40
4	Analysis	41
4.1	Recommendations of Usage Scenario and Comparison	41
4.2	Performance Evaluation	43
4.2.1	Efficiency of Packet Compression	44
4.2.2	Future Evaluation Plan	45
4.3	Summary of Achievements	46
4.4	Future Work	46
4.4.1	Packet Size and Datarate	47
4.4.2	Rules	47
4.4.3	Address Management	47
4.4.4	Downlink Absence	48
	Conclusion	48
	Bibliography	52
	Appendix: Source Code	53

List of Tables

1.1	LPWAN technologies comparative table	8
1.2	Frame payload datarate dependence table	14
1.3	6LoRaWAN vs 6LoWPAN protocol stacks comparison	17
3.1	Field Descriptions in a Rule	35
4.1	SCHC-enhanced vs conventional LPWAN scenarios comparison	43
4.2	Context rules for performance evaluation	44
4.3	Compression percentage by context	45

List of Figures

1.1	IoT communication scheme	7
1.2	LoRa messages transmission and reception timing	10
1.3	LoRa stack	11
1.4	LoRaWAN network architecture	12
1.5	LoRaWAN network protocol stack	12
1.6	LoRaWAN uplink message PHY format	13
1.7	LoRaWAN downlink message PHY format	13
1.8	LoRaWAN message MAC format	13
1.9	LoRaWAN message MAC payload format	13
1.10	LoRaWAN message frame header format	14
1.11	IPv6 header format	15
1.12	IPHC packet format	16
1.13	6LoRaWAN packet format	17
2.1	SCHC protocol stack	18
2.2	SCHC Compression and Fragmentation overview	19
2.3	SCHC Packet format	20
2.4	SCHC Compression Rule format	22
3.1	Microchip RN2903 LoRa Technology Mote	27
3.2	RN2903 command interface	28
3.3	Everynet LoRaWAN Gateway	29
3.4	Everynet Gateway's ports	30
3.5	Everynet Platform	31
3.6	Everynet Platform's message panel	31
3.7	System's block diagram	34
3.8	Packet capture using <code>tcpdump</code> command	38
4.1	SCHC-enhanced LPWAN scenario	42
4.2	Conventional LPWAN scenario	43

Introduction

This thesis is contextualized on the current development of the Internet of Things (IoT), a recent communication paradigm which proposes the interconnection of several diverse devices, such as sensors, actuators, and all sorts of objects, through the Internet, thus allowing the emergence of a whole new diversity of services built upon these new interactions. Within this context, and as a natural evolution of the former concept of Wireless Sensors Networks (WSN), a new category of networks is created: Low Power Wide Area Networks (LPWAN), which have a special set of features that make them particularly interesting for the development of IoT. At the same time, they bring up brand-new challenges concerning the implementation of the new IoT paradigm, especially in regard to interoperability and integration of Internet Protocol version 6 (IPv6). This work focuses precisely on this last challenge.

Motivation and Context

The IoT paradigm, which is currently blooming and spreading, has opened new possibilities for the use and application of the Internet, and thus generated new working areas within the telecommunications field. One of the typical applications of IoT that has been widely implemented in the recent years is that of WSNs, which seek massive deployment of sensors over a vast geographical area, achieving large scale monitoring services not possible until now. In accord with this objective and the possible derived applications, the concept of LPWAN arises. In these networks, several devices, usually called “sensor nodes”, are distributed within a rather big region, and have to periodically collect data that will be sent further through the Internet to a server, for later processing and use. To make this possible, such devices must comply with a particular series of features, namely [1]:

- Extremely low power consumption, allowing their deployment on the field, ideally for a period of several years, on a single battery charge.
- Long range communication, thus covering wide areas with a reasonable number of devices.
- Low transfer rate, in consistency with low power consumption and range optimization, which is supported by the fact that the applications these networks are designed for do not need such a high data transfer rate.
- Low cost, promoting their massive production and deployment in big numbers.

These features, however, place certain constraints over the way nodes communicate and connect to the Internet. In particular, these technologies' natural low transfer rate and small message size make the use of standard Internet protocols, such as Internet Protocol (IP), Transmission Control Protocol (TCP) and Hypertext Transfer Protocol (HTTP), unpractical and generally non-viable. Due to these constraints, nodes do not have individual IP addresses and are unable to connect to the Internet directly, currently having to do so through a Radio Gateway (RG), which acts as an intermediary between the nodes and the Network Gateway (NG) that connects to the rest of the Internet. This network architecture is called a "star-of-stars" topology and represents the traditional LPWAN architecture [1]. Moreover, there are various LPWAN technology developers with distinct communication mechanisms and service models; all of this threatens the interoperability that is desired in IoT and makes its architecture rather rigid and lacking flexibility.

In the last years, new alternative protocols have been developed, especially designed for enabling Internet connection using IPv6 in constrained networks, and particularly in LPWAN technology devices. These protocols include Constrained Application Protocol (CoAP), an Application layer protocol based on User Datagram Protocol (UDP) in the Transport layer, as an alternative to HTTP over TCP. In fact, CoAP has been adopted as the official Application layer protocol for LPWAN networks by the Internet Engineering Task Force (IETF) [2], the organism in charge of proposing and establishing standards for the use of the Internet. Another Application layer protocol that has been recently released is Hypertext Transfer Protocol version 2 (HTTP/2) [3], the latest version of HTTP, which is much more compact and efficient than its predecessor, making it more suitable for the IoT [4]. HTTP/2 is being tested for a configuration tailored for the IoT, so as to take advantage from the already existing know-how on HTTP [4], thus favoring IoT's rapid expansion.

Along with CoAP and HTTP/2, other techniques for adapting traditional protocols, especially IP, have been developed. In particular, the IETF is currently working on a compression and fragmentation scheme called Static Context Header Compression (SCHC) whose objective is to enable IPv6 packet transmission over LPWAN networks [2]. SCHC addresses two main issues: i) compression of headers using physical addresses; and ii) taking advantage of the rather static topology of LPWAN networks, and fragmentation of IPv6 packets when necessary, to make them fit within the Maximum Transfer Unit (MTU) determined by the corresponding technology. Although the fragmentation mechanism is still under development, the compression mechanism is already complete and proposed, but has not yet been officially implemented.

SCHC development is carried on by the Working Group for IPv6 over Low Power Wide-Area Networks (`lpan` WG), and is partially based on former works, also by IETF WGs, addressing devices with limited resources, like `6lo`, `6tisch` and `6lowpan`. The main purpose of these developments is the adaptation of the IPv6 protocol for use in constrained networks, some of them covering the general case and others regarding a specific application. Particularly, the `6lowpan` WG has created an adaptation of IPv6 for networks compliant with the IEEE 802.15.4 standard and Low-power Wireless Personal Area Networks (LoWPAN) in general, constituting what is denominated 6LoWPAN [5].

Some recent works have managed to implement IPv6 over LPWAN networks using an

adaptation based on 6LoWPAN [6]. This constitutes a concrete base that suggests the viability of implementing SCHC and also allows for a potential comparison once this is accomplished. The work hereby presented intends to explore the possibility of a functional integration of SCHC compression in real hardware, carrying out the implementation and evaluation of the resulting system, and analyzing its performance and suitability for the IoT environment.

Problem Statement

Currently, there is no IPv6 support for LPWAN terminal nodes, making them invisible to the network and unable to directly connect to the Internet, having to do so through a Radio Gateway which encapsulates and translates Data Link layer messages of each particular technology into IPv6 packets and relays them to a Network Gateway. This greatly affects the interoperability sought in the Internet of Things ecosystem, adding undesirable complexity and costly equipment to achieve communication between different technologies. There is an Internet Engineering Task Force proposed standard that intends to address this matter: Static Context Header Compression, but to the best of our knowledge, it has not yet been physically implemented nor evaluated to date.

Scope

The present work covers the adaptation and usage of IPv6 for a Long Range Wide Area Network (LoRaWAN), one of the first and principal LPWAN technologies [1], following the IETF's SCHC specifications.

The process considers the development of an integrated hardware and software platform that implements the essential features of SCHC to make the transmission of IPv6 packets over a LoRaWAN network possible. This involves studying and understanding the LoRaWAN protocol and the SCHC specification, so as to properly fit SCHC packets and fragments into the LoRaWAN message format to be transmitted.

The viability and proper operation of the generated platform is to be verified according to certain performance metrics that will be specified throughout the work, leaving out any evaluation method not explicitly mentioned. Where evaluation cannot be performed, whatever the reason, a potential development direction through which evaluation could be achieved will be pointed out.

A qualitative and quantitative analysis of the obtained results will be carried out, regarding the overall system performance and general recommendations on the applicability of SCHC to LPWAN networks and LoRaWAN networks in particular.

This is essentially an experimental work. The system evaluation and required adaptations will be made fundamentally based on tests. No specific application development is intended,

but only understanding what is required for a proper implementation of SCHC over a LoRaWAN network and potential advantages over alternative implementations of IPv6.

Objectives

General Objective

- Evaluate the applicability and performance of Static Context Header Compression for the implementation of IPv6 over a representative LPWAN technology.

Specific Objectives

- Establish the state of the art regarding the implementation and evaluation of IPv6 in LPWAN networks.
- Implement SCHC over a LoRaWAN network concerning one terminal node and a LoRaWAN RG.
- Build an integrated test-bed platform for IPv6 packet transmission and verification over a LoRaWAN network.
- Make a comparative evaluation of the system regarding the additional capabilities granted by IPv6 in contrast with a traditional LoRaWAN network.

Methodology

This work will begin with a revision of the state of the art for IPv6 implementations over LPWAN networks including the basic concepts and fundamentals that allow the understanding of the following development. Firstly, the main obstacles and difficulties for the implementation of IPv6, given the characteristic constraints of LPWAN technologies, will be established. LoRa and LoRaWAN technologies will be described, along with their particular features and architecture. Next, 6LoWPAN will be covered, explaining how it solves the IPv6 adaptation for some constrained networks [7]. 6LoRaWAN adaptation will also be covered, including its relation to 6LoWPAN and the way it solves IPv6 application over a LoRaWAN network. Finally, SCHC will be presented, along with the features it shares with the former adaptations.

The platform to be developed will be described in a general fashion, enumerating its components and explaining each block. The adaptation programming will be based mainly on the IETF work and the equipment's existing documentation. The required code is written fundamentally in Python. The use of other related programming languages, whenever necessary, will be explicitly stated and clarified.

The system hardware consists of an Everynet LoRaWAN Gateway integrating the SX1301 and SX1276 chips and a Microchip LoRaWAN node populated with the RN2903 module. Each device carries a corresponding antenna for the 915 MHz frequency band. The setup for the whole system and for each test will be explained step by step.

The implementation process for each block will be described in detail, addressing its function within the global system. Modifications with respect to the original code, when present, will be precised and argued, along with the aspects of traditional IPv6 they solve and the advantages they generate, if any.

Evaluation criteria for the implemented system's performance will be established, both numeric and purely qualitative. Qualitative criteria will be clearly defined, as well as the circumstances in which they are fulfilled.

Thesis Organization

This work is composed of five chapters. The first chapter presents the fundamental knowledge that is necessary for understanding the development hereby described, as well as other works that pursue a similar goal and serve as precedents. The following chapter describes the hardware used throughout the process, detailing the connections of the system's setup and the methodology applied for the tests run. In chapter three, a full explanation of the SCHC scheme is given, along with the manner in which it was implemented by the author. Chapter four addresses the results obtained in the tests and a general evaluation of the implemented system's performance, regarding the metrics previously established. At last, chapter five gives the main conclusions derived from the evaluation, including advice concerning the use of SCHC, particularly in LoRaWAN networks, and what future improvements or tests might be relevant for its full applicability.

Chapter 1

Background and State of the Art

This chapter presents the fundamental concepts that are necessary for the study and comprehension of LPWAN technologies, including a review of the most relevant protocols and platforms, with an emphasis in LoRaWAN. Latest advancements towards IPv6 adaptations for LPWAN networks are also covered.

1.1 Internet of Things

The IoT, in a general sense, can be considered as a broad vision of communications that has both technological and societal implications. From a technical perspective, it consists of a global infrastructure capable of interconnecting physical and virtual objects through information and communication technologies (ICT), new and legacy, with the purpose of providing advanced services and applications [8].

The referred objects are commonly called “things”, and they must be able to be identified and integrated into communication networks. Physical things correspond to things existing in the physical world, which can be sensed, acted upon, and interconnected with a communications network through devices especially designed for this purpose. They include all sorts of goods and regular objects, machines, equipment, robots, and the environment itself. Virtual things, on the other hand, belong to the information world and can be stored, processed or accessed. They correspond to multimedia content, application software and other pieces of functional software [8].

To allow for communication and information exchange between the different things, each physical thing must be associated with a device at least capable of establishing a communication link, and optionally capable of data sensing, capture, storage or processing, or any form of external actuation [8]. Communication between devices can be direct, through the network by using a gateway, or through the network without the need of a gateway. A general scheme for IoT communication can be seen in Figure 1.1.

IoT applications are virtually unlimited, covering a wide range of purposes, such as in-

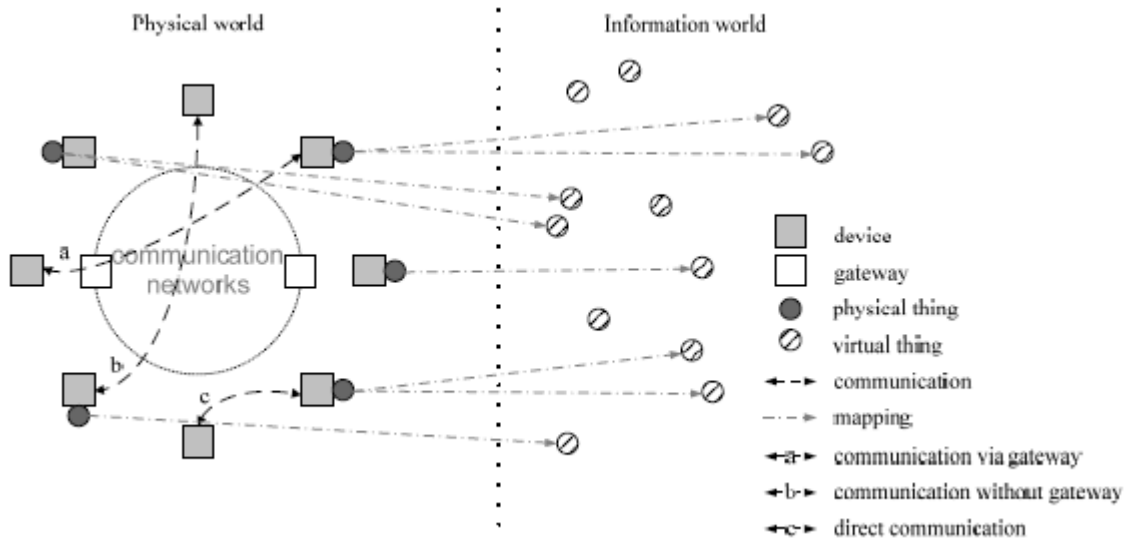


Figure 1.1: General scheme for IoT communications. Physical things are objects in the real world, to which devices are connected in order to make them communicate with one another. Devices are pieces of electronic equipment with communication capabilities, sometimes including sensors, actuators and others. Virtual things are information structures related to physical objects or representing them, such as databases, virtual memories or scripts. Source: [8].

telligent transport systems, health monitoring, environmental monitoring, measurements in distributed generation systems, and many others. Each application requires a different type of device, depending on the abilities needed to achieve its purpose. This implies that devices are frequently designed and configured for a specific application (or range of applications), but also that adding new abilities to devices makes them suitable for new applications, expanding potential solutions.

A fundamental aspect of IoT with respect to its global implementability concerns the heterogeneity of the communication networks that compose it. Before this new vision was born, the Internet had been designed and built based on the interconnection of equipment with relatively big storage capacities, processing power, and data transfer rates. In fact, its development has always been focused on finding methods to improve these attributes as much as possible. According to this new paradigm, however, and given the nature of the desired applications, the capabilities of the devices composing the network are constrained by their size, power consumption, price and the need for portability in many cases. As a consequence, the new kind of devices that are intended to get connected to the Internet do not fit the usual requirements. Besides, the diversity of technologies that have emerged by virtue of the wide range of possible applications, constitutes a challenge regarding interoperability, as all the different devices must comply with the Internet’s basic communication structure. Thus, the Internet is currently faced with a variety of devices with very dissimilar characteristics that need to connect through a common infrastructure and language.

1.2 LPWAN Technologies

LPWAN technologies seek to interconnect a variety of devices generically denominated “sensor nodes” through the Internet, with the purpose of delivering services centered around the collection of data within a wide geographical area, by creating applications that require relatively low data transfer rates and long periods of permanence for the devices on site [1]. Such technologies have been under development by different entities, which results in the coexistence of very diverse systems, with their own proprietary hardware, communication mechanisms, infrastructure and service model. Some of the principal LPWAN technologies, which are considered by the IETF in their adaptation developments, are the following [1]:

- LoRaWAN
- SigFox
- Narrowband IoT (NB-IoT)
- Wi-SUN Alliance Field Area Network (Wi-SUN)

A general comparison of some of these is represented by Table 1.1.



	 LoRa / LoRaWAN	 Sigfox	NB-IoT
Origin	France	France	USA (Global)
Proprietary or open	LoRa – proprietary LoRaWAN - open	Net – proprietary Devices – open	Open
Cellular	No	No	Yes
Spectrum	Unlicensed	Unlicensed	Licensed
Range, km	urban: 2-5 rural: 15	urban: 3-10 rural: 30-50	urban: 1-5 rural: 10-15
Speed, uplink / downlink	50 kbps / 50 kbps	300 bps / –	250 kbps / 250 kbps
Power consumption	●●●	●	●
Security	●●	●●	●●●
Availability of devices	●●	●●●	●●
Price*	●●	●	●●

Table 1.1: Comparative table for some of the principal LPWAN technologies. Extracted from: [9]

All LPWAN technologies considered by the IETF share the following characteristics [1]:

- Low power consumption sensor node hardware
- Low data transfer rate
- Wide range communication link
- Relatively low sensor node cost
- Star-of-stars architecture, where the NG represents the center of the main star, connecting to one or more RGs, and each RG constitutes a smaller star connecting with several sensor nodes

The first four characteristics are related to the type of application for which the devices are designed for. Low power consumption is desired so that the sensor nodes can remain functioning in place for long periods of time, ideally up to five or ten years [1], powered only by a battery and without needing to replace it during this period. This gives the nodes autonomy and reduces maintenance costs. The low data transfer rate typical of this sort of technology derives from the devices sending a very limited amount of data in each working cycle, staying in “sleep” state during most of it; this accords with the low power demand. As the technology’s name suggests, the nodes’ wide communication range allows for a new kind of service based on their deployment over vast areas. Generally, there is a trade-off between the communication range and the transfer rate. The modules’ low cost makes it possible to simultaneously use several of them for the same application, which is also one of this technology’s objective. Lastly, the nodes’ natural architecture, with terminal nodes organized around a RG, is a consequence of them being unable to connect to the Internet directly, as they do not support IPv6.

1.3 LoRaWAN

1.3.1 LoRa

Long Range (LoRa) is the name given to the Physical (PHY) layer technology developed by Semtech, which allows for a long range communication using a spread spectrum frequency modulation technique called Chirp Spread Spectrum (CSS). LoRa is the first low cost commercial implementation of this modulation technique, which has been used in the military field for decades, because of its long range communication and robustness against interference [10].

LoRa technology is designed to operate over some Industrial, Scientific and Medical (ISM) bands of the electromagnetic spectrum, which are unlicensed and free to use, under certain constraints. Available ISM bands vary depending on the region of the world, so there is an according LoRa hardware fabricated especially for each one of them: EU868 (European Union, 868MHz), EU433 (European Union, 8433MHz), US915 (United States, 915MHz) and AS430 (Asia, 430MHz). All of them utilize the CSS technique, which privileges communication reach over transfer rate [11]. LoRa devices have different modes of operation that regulate the trade-off between data transfer rate and communication range, by modifying the Spreading Factor (SF) and transmission power [12].

LoRa technology contemplates three classes of devices, depending on the available power sourcing and message reception availability needed:

- **Class A (all):** This class represents the basic functionality, and all LoRa devices are implemented with it. Class A terminal nodes have the lowest duty cycle, and therefore the lowest power consumption. They are designed to be powered using a battery. They have predefined and fixed transmission and reception windows. Their duty cycle is less than 1%.
- **Class B (beacon):** Also designed to be powered with a battery, although in addition to the elemental working cycle, they include configurable scheduled reception windows. These windows are initiated through a synchronization beacon sent by the RGs, giving the terminal nodes a time reference to open periodical receive windows, called “ping slots”.
- **Class C (continuous):** This device class can receive downlink messages at anytime, whenever it is not transmitting. As such devices are constantly working and they have the largest power consumption, they are supposed to be mains powered.

All LoRa modules must implement at least class A, and optionally add class B or class C. Classes correspond to operation options of the LoRaWAN protocol, the Medium Access Control (MAC) layer protocol designed for LoRa technology (see Section 1.4).

Figure 1.2 shows the basic message transmission and reception time schedule, implemented by class A. Additional reception times, either continuous or beacon-based, corresponding to classes C and B respectively, are added to this structure.

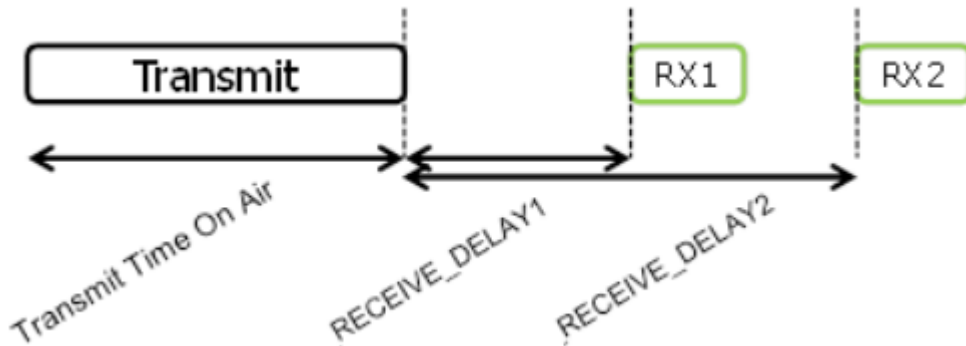


Figure 1.2: Basic timing structure for message transmission and reception in LoRa devices (Class A). The device wakes up and opens a ‘transmit window’, after which there is a delay for the RG and NS to forward and process the message, and then there is a first receive window (RX1) for the device to listen to the NS reply. If no reply is received in RX1, there is a second delay and then a second receive window (RX2) opens, for a potential NS reply. Source: [12].

A general overview of the layers involved in LoRa communication can be seen in Figure 1.3:

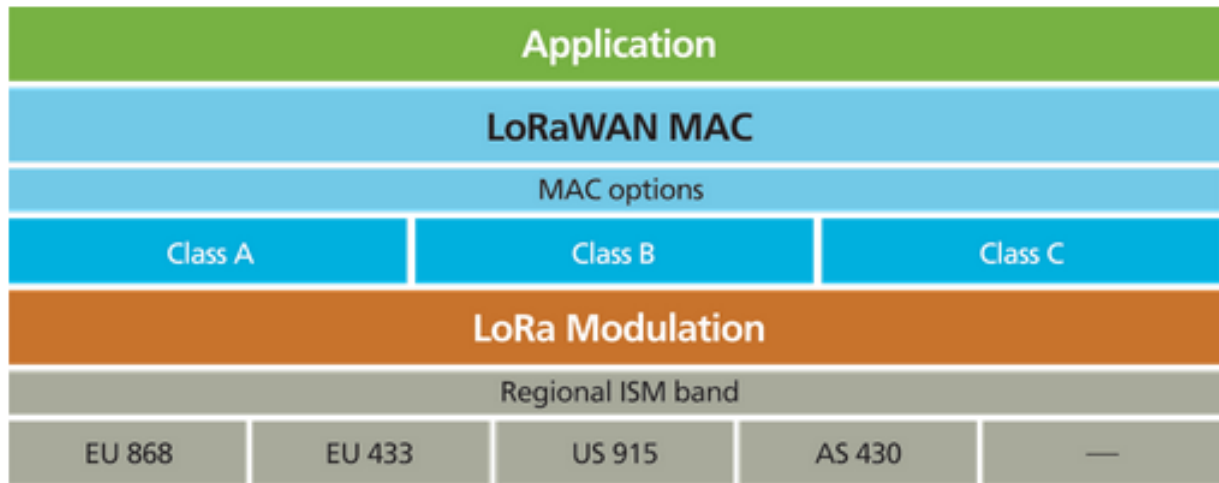


Figure 1.3: General scheme for LoRa communication stack. Source: [13].

1.3.2 LoRaWAN protocol

LoRaWAN is the communication protocol used by LoRa devices, and it has been developed by the LoRa Alliance. LoRaWAN defines the MAC layer protocol and the system’s architecture [10].

In the architecture defined by LoRaWAN, terminal nodes¹ organize in a star fashion around a RG² (or many) with which they communicate using the LoRaWAN protocol, while the RG establishes a link with a Network Server (NS) through a standard IP connection [12]. Although communication within these systems is bidirectional, uplink strongly predominates. In Figure 1.4 one can observe a scheme for a LoRaWAN system. A more detailed description of the node-RG-NS interaction is shown in Figure 1.5.

Communication between nodes and RGs is distributed in different frequency channels and data rates, which represent the different levels of trade-off between the message duration and the communication range [12]. By virtue of CSS modulation, communications using different data rates do not interfere each other. LoRa data rates range from 0.3 kbps to 50 kbps [12]. LoRa infrastructure is capable of applying a mechanism called Adaptive Data Rate (ADR) which optimizes the transfer rate and communication range of all devices within the network. This mechanism follows these rules [12]:

- Each node changes channels in a pseudorandom manner in each transmission; this frequency diversity makes the system more robust.
- Nodes respect the duty cycles established depending on the sub-band used and local spectrum regulations.
- Nodes respect the the maximum transmission duration depending on the sub-band used and local spectrum regulations.

¹The term used by LoRa Alliance is “End-device”.

²The term used by LoRa Alliance is “Gateway”.

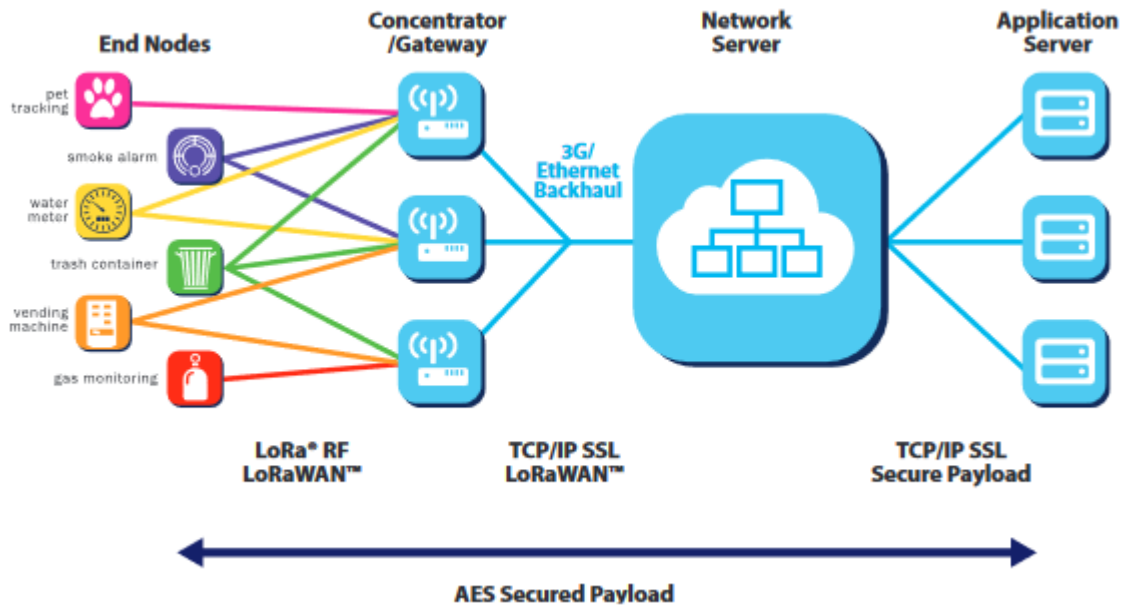


Figure 1.4: LoRaWAN communication network and platform architecture. Source [10].

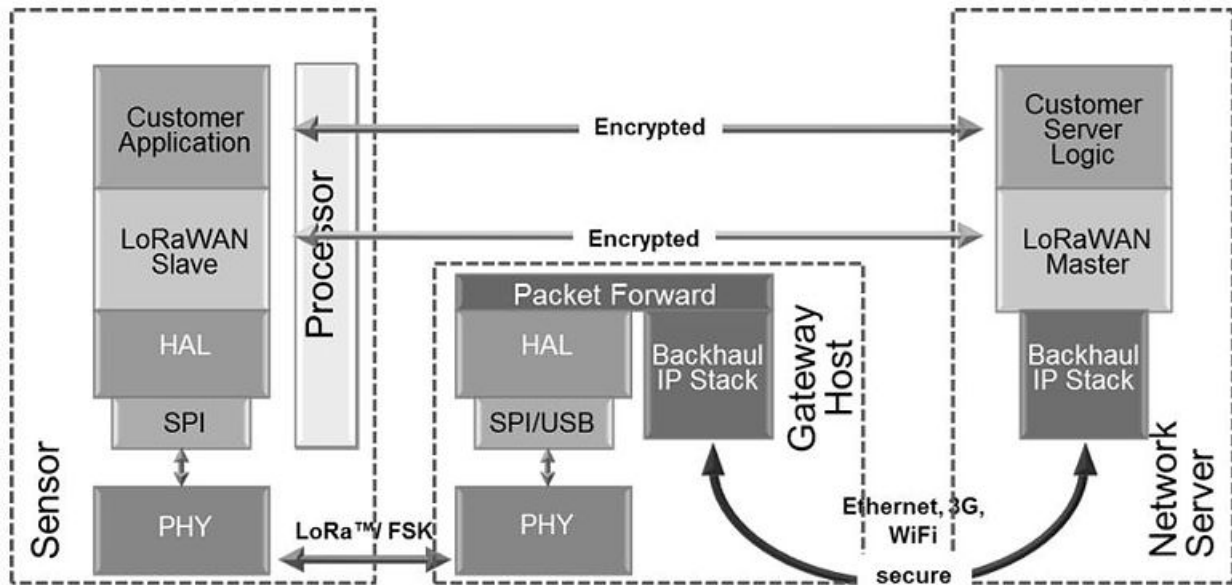


Figure 1.5: Protocol stack structure for communication between a LoRaWAN end-node and the NS through a Gateway. Source: [14].

1.3.3 Message Format

Here, the general format of LoRaWAN messages is described and explained. As the present work focuses on the Network layer, the description's emphasis is placed on the format fields general organization and fields related to terminal nodes identifiers, i.e., their address within the LoRaWAN network. The rest of the fields will not be discussed in detail.

Messages are classified according to their direction within the system in *uplink* (from terminal node to server) and *downlink* (from server to terminal node).

At the PHY layer, uplink and downlink message format is observed in Figures 1.6 and 1.7, respectively. The fields constituting them are [12]:

- **Preamble:** Synchronization preamble
- **PHDR:** PHY layer header
- **PHDR_CRC:** PHY layer header's Cyclic Redundancy Check
- **PHYPayload:** PHY layer payload
- **CRC:** Message's Cyclic Redundancy Check

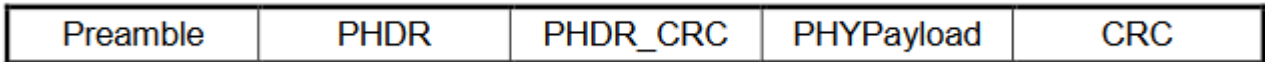


Figure 1.6: LoRaWAN uplink message PHY format. Source: [12].

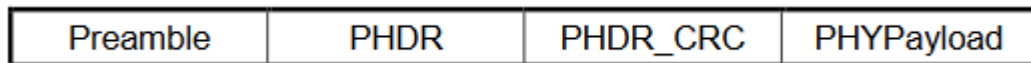


Figure 1.7: LoRaWAN downlink message PHY format. Source: [12].

The PHY layer payload (PHYPayload) contains the MAC layer message. Its format can be seen in Figure 1.8. The corresponding fields are the following [12]:

- **MHDR:** MAC layer header
- **MACPayload:** MAC layer payload
- **MIC:** Message Integrity Check

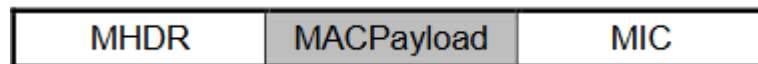


Figure 1.8: LoRaWAN uplink and downlink message MAC format. Source: [12].

MACPayload field's inner format is visible in Figure 1.9. Its composing fields are [12]:

- **FHDR:** Frame header
- **FPort:** Frame port
- **FRMPayload:** Frame payload

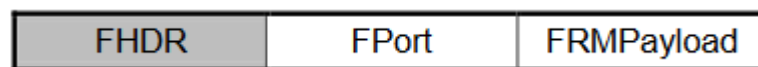


Figure 1.9: LoRaWAN message MAC payload format. Source: [12].

Finally, inside the FHDR field, one can find the DevAddr field, which refers to the terminal node address within the LoRaWAN network. This 4-byte long field is of interest regarding

the Network layer routing over a LoRaWAN network. The complete FHDR format can be seen in Figure 1.10.

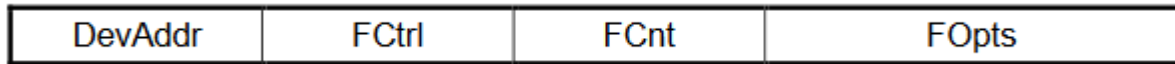


Figure 1.10: LoRaWAN message frame header format. Source: [12].

1.3.4 Datarate

LoRaWAN’s maximum available message payload is dependent on a variable denominated datarate (DR), which represents the link’s data transfer rate and is tightly related to allowed time on air, transmission power and modulation settings. For the US region, the relationship between DR and the maximum frame payload is summarized in Table 1.2.

DataRate	<i>M</i>	<i>N</i>
0	19	11
1	61	53
2	137	129
3	250	242
4	250	242
5:7	Not defined	
8	41	33
9	117	109
10	230	222
11	230	222
12	230	222
13	230	222
14:15	Not defined	

Table 1.2: Maximum available frame payload (in bytes) depending on current datarate. M and N represent the available payload size when the FOpts field is absent or present, respectively. Source: [12]

1.4 IPv6

It is the most recent version of the Network layer standard protocol, which allows the interconnection between devices with distinct characteristics through the Internet. Its basic data unit is the packet, which is composed of a header and a payload. The header consists of a minimum fixed structure of 40 bytes in size that contains the indispensable information about the packet, like the protocol version, payload and additional headers lengths (in bytes), hop limit (time to live), origin and destination addresses, and others [15]. A graphic description of an IPv6 packet’s fixed header structure can be observed in Figure 1.11.

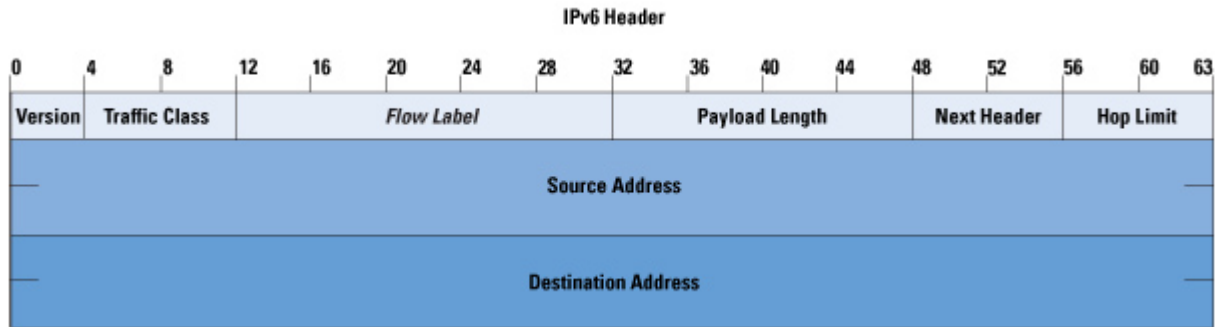


Figure 1.11: IPv6 packet main header format. Source: [16].

After the fixed header come the extension headers, which are optional. They provide additional functionalities for security, routing, fragmentation and other features. At the end of the packet is the payload, which contains the information coming from higher layers protocols.

IPv6 introduces some substantial changes with respect to Internet Protocol version 4 (IPv4), its former version, especially regarding IP addresses, going from 32 to 128 bits in size [15]. This extended address aims to definitively solve the IPv4 address exhaustion matter, a consequence of the massive expansion of devices currently able to connect to the Internet. The huge amount of addresses supported by IPv6 is in tune with the foreseen IoT scenario for the upcoming years, with several devices deployed in vast areas. Because of this, IPv6 is considered adequate and necessary for IoT and the future of the whole Internet. However, IP was originally designed for networks built upon the high transfer rates, processing resources and power connection typical of traditional Internet scenarios, and is not natively viable in constrained networks and particularly in LPWAN networks. Therefore, IPv6 adaptations that are especially conceived to operate over constrained networks are required. The main issues that need to be addressed regarding the use of IPv6 in LPWAN networks are the following [5, 7]:

- Its header is at least 40 bytes long, which is excessive for the typical frame size in LPWAN networks.
- It requires a Maximum Transfer Unit (MTU) of at least 1280 bytes over the MAC layer [15], which is disproportionate and natively non-viable within the LPWAN environment [5].

To put this into context, the maximum frame size at the PHY layer for a LoWPAN network as established by standard IEEE 802.15.4 is 127 bytes [17]. If we take out the PHY and MAC headers plus the security options of devices complying with the standard, the worst-case scenario leaves only 41 bytes for the information concerning the Transport layer and higher layers. This is rather small and far away from the 1280 imposed by the MTU requirement, so an adaptation layer that provides a compression and fragmentation mechanisms is needed [7].

1.5 Adaptations of IPv6 over Constrained Networks

1.5.1 6LoWPAN

It consists in an adaptation layer for IPv6 aimed for use in LoWPAN networks, compliant with the standard IEEE 802.15.4. 6LoWPAN integrates header compression and packet fragmentation mechanisms to achieve the necessary size reduction from the standard MTU of 1280 bytes to the 127 bytes of the MAC layer frame [7]. This adaptation layer makes use of the communication context to compress the IPv6 packet header, omitting a significant fraction of the addresses, the longest fields. In order to do it, it takes advantage of the 16-bit and 64-bit addresses defined in the IEEE 802.15.4 standard [17].

6LoWPAN defines a header compression format denominated IP Header Compression (IPHC) which essentially compresses the header of IPv6 packets according to the link and network context. In the best case, total header length is reduced to 2 bytes, while in the worst case, it reduces to 3 bytes [18].

The IPHC format can be seen in Figure 1.12.

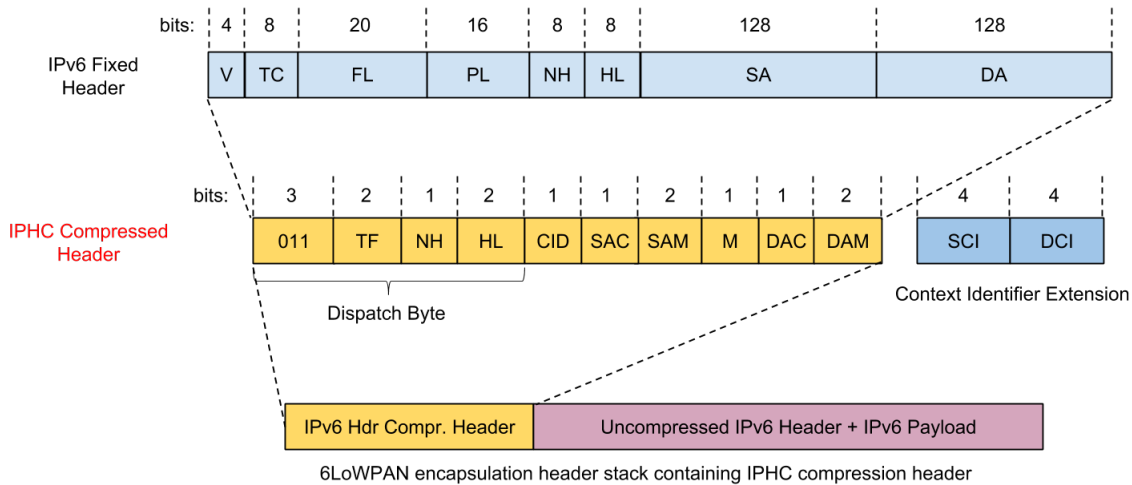


Figure 1.12: 6LoWPAN's IPHC packet format. Adapted from: [19].

1.5.2 6LoRaWAN

6LoRaWAN is the name given to the adaptation of IPv6 for LoRaWAN networks developed in [6]. The adaptation is completely analogous to 6LoWPAN, adopting UDP and CoAP as the higher layer protocols. In Table 1.3, a protocol stack comparison between 6LoWPAN and 6LoRaWAN is displayed.

	IEEE 802.15.4	6LoWPAN	LoRaWAN	6LoRaWAN
Application layer	e.g. ZigBee	e.g. CoAP	custom	e.g. CoAP
Transport layer	-	e.g. UDP	-	e.g. UDP
Network layer	e.g. ZigBee	IPv6	-	IPv6
		6LoWPAN adaptation		6LoRaWAN adaptation
Data Link layer	802.15.4 MAC	802.15.4 MAC	LoRaMAC	LoRaMAC
Physical layer	802.15.4 PHY	802.15.4 PHY	LoRaPHY	LoRaPHY

Table 1.3: 6LoRaWAN vs 6LoWPAN protocol stacks compared. Standard IEEE 802.15.4 and LoRaWAN native protocol stacks included as reference. Adapted from: [6].

6LoRaWAN allows for the transmission of IPv6 packets within a LoRaWAN network in a way that is compatible with native LoRaWAN packets. In other words, by using this adaptation it is possible to integrate native LoRaWAN nodes, 6LoRaWAN nodes, and nodes operating on traditional IPv6 into the same network.

A 6LoRaWAN packet's format is visible in Figure 1.13. Compression introduced by 6LoRaWAN corresponds to the Comp. IPv6 Header field, where the fundamental advantage resides in the compression of origin and destination addresses through the usage of the terminal nodes' DevAddr, which is stored by RGs present in LoRaWAN networks [6].

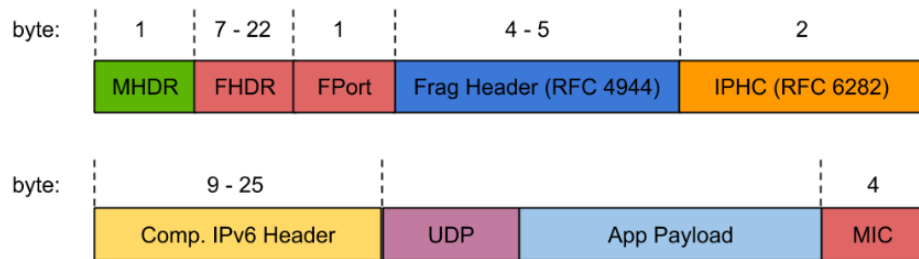


Figure 1.13: 6LoRaWAN packet format. Adapted from: [6].

Chapter 2

Static Context Header Compression

This chapter describes the SCHC proposal in a generic fashion, details its compression mechanism and introduces the required associated concepts. The fragmentation scheme is out of the scope of this work, and is therefore not included in the detailed description. All the information hereby presented is the result of the IETF lpwan WG work [2, 20], combined with the author's interpretation and understanding where the IETF document is not explicit or sufficiently clear.

2.1 SCHC Overview

SCHC is generally considered as an adaptation layer between IPv6 in the Network layer and the underlying LPWAN protocol from the Data Link layer. It is composed of two sublayers: one for compression and one for fragmentation. This is graphically shown in Figure 2.1.

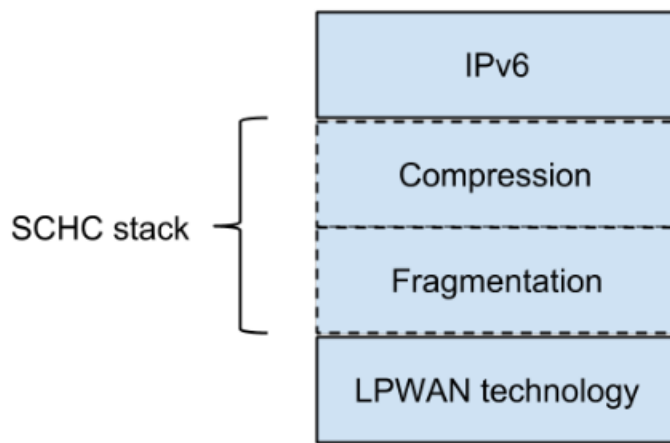


Figure 2.1: SCHC protocol stack. Based on [20].

When an IPv6 packet is ready for transmission in the sender side, header compression is first applied to it, resulting in what is called a SCHC Packet. If the SCHC Packet is greater than the LPWAN link's MTU, it should be fragmented, producing two or more SCHC Fragments. The SCHC Packet or each Fragment is then sent in an LPWAN frame to the receiver. If the SCHC Packet was fragmented, the SCHC Fragments are reassembled on the receiver side to recover the SCHC Packet. Finally, the SCHC Packet, either native or reassembled, is then decompressed following the corresponding actions. The diagram in Figure 2.2 illustrates this behavior. Note that despite SCHC is designed for all LPWAN technologies, the actual decision to use SCHC Fragmentation is not compulsory and is therefore left to the specific technology.

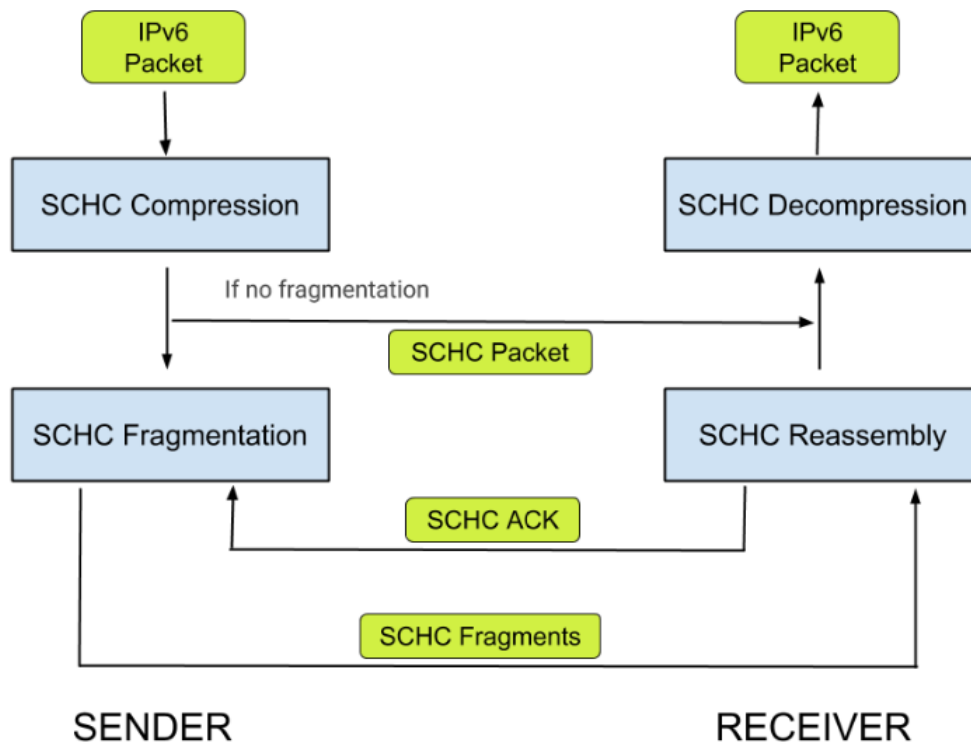


Figure 2.2: SCHC Compression and Fragmentation overview. Based on [20].

The compression and decompression mechanism used on each side is denominated SCHC Compressor/Decompressor (SCHC C/D). The functional units implementing such mechanism are also referred to with this name. Thus, a SCHC C/D exists both inside the device and on the network side. They achieve compression and decompression based on certain Rules.

2.2 Rules and Context

The SCHC mechanism's essential purpose is to take advantage of the rather static context of LPWAN communications, where devices are intended for a particular set of applications, and the network's architecture, besides some potential degree of mobility, generally does not change over time. For any given LPWAN network the set of devices is usually known

(i.e., their addresses/identifiers), and as each one of them communicates with one or very few specific built-in applications, communication occurs in much the same fashion most of the time, and traffic is highly predictable [20]. Thus, instead of sending full IPv6 packets whose information is mostly already known, SCHC proposes to establish a set of shared Rules between sender and receiver that determines in which cases and to what extent information is known by the other side and can therefore be omitted, and when (and how) it must be sent or compressed. These Rules are based on the communication context, regarding the network’s architecture and the characteristics of data exchanged between devices and servers. This allows SCHC to avoid context synchronization, which is the most bandwidth-consuming operation in other header compression mechanisms [20].

Rules define which header fields can be omitted or compressed and what actions should be performed during the compression and decompression stages to respectively compress the packet header on the sender side and recover it on the receiver side. In SCHC, compression is achieved by sending the Rule IDs in place of the header fields, plus a Compression Residue bearing some parameters used for compression, as well as the header fields that could not be compressed. Together they compose SCHC’s Compressed Header. The Compression Residue is always followed by the packet’s original payload. A SCHC Packet’s format can be seen in Figure 2.3. It should be noted that the Compression Residue might be absent, and that the compressed header may not necessarily be smaller than the original header, although of course this is generally not the case. Another detail to take into account is that Rule IDs do not have a fixed length; this is left to the implementation.

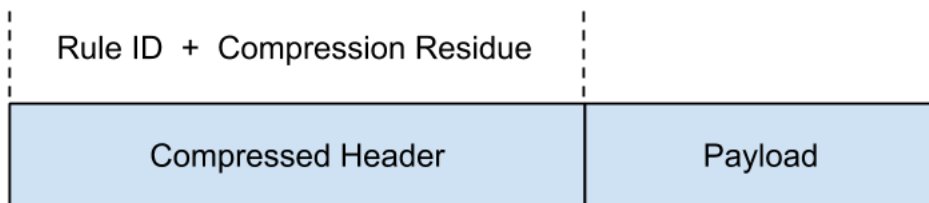


Figure 2.3: SCHC Packet format. Based on [20].

Regarding Figure 2.3, the Compressed header is formed by concatenating two blocks: the Rule ID block and the Compression Residue block. The Rule ID block contain the chosen Rule’s Rule ID, while the Compression Residue block contains the Compression Residues for each of the fields in the same order that they were produced. This is detailed in section 2.4. Although the use of more than one compression Rule at the same time is not explicitly forbidden in [20], the analysis in the document consistently seems to assume the use a unique Rule, which is the most simple and effective case. While the author strongly believes that using more than one Rule is actually feasible and some parts of the developed system were in fact designed with this in mind, for the sake of simplicity and programming ease a unique Rule will be assumed from here onwards, both within the theory and the implementation.

2.3 Rule Format

A Rule corresponds to a set of Field Descriptions (FD), each referring to a particular IPv6 header field. FDs comprise the following items:

- Field ID (FID): Uniquely identifies the header field to analyze.
- Field Position (FP): Refers to the occurrence of the header field within the header. Most header fields only appear once, so FP defaults to 1.
- Field Length (FL): The length of the field in bits. If its length is variable, FL represents the type of value, which defines its length unit (bit, byte, or other) and how to compute it.
- Direction Indicator (DI): Describes the direction for communication (uplink, downlink or bidirectional) to which the FD applies. This allows different management for information going up or down.
- Target Value (TV): The value in an FD with which the actual header Field Value (FV) will be compared, using the Matching Operator. It can be a single value or some sort of array with multiple entries.
- Matching Operator (MO): Strategy used when comparing a TV to an FV. Its result is either True or False, and determines whether the Rule is applicable or not to a given packet. It is used only during compression.
- Compression/Decompression Action (CDA): Defines the action taken when compressing and decompressing the field, depending on the context. It is only applied when a Rule is actually selected.

A Rule might include any set of available header fields, not necessarily all of them and potentially some of them more than once, making use of the DI or FP items. However, for each header field included in the Rule, i.e., for each FD, all of the items must be present, with the only exception of TV when the corresponding MO is set to `ignore`. The FD's in a Rule must be presented in the order in which the fields appear in a given header[20]. Rules look as depicted in Figure 2.4.

2.4 Packet Processing

In order to correctly transmit SCHC Packets over an LPWAN link and recover the original IPv6 packets, a few things should be ensured. First, both sides of communication must share the same set of Rules. Rules are used by the SCHC C/D on each side to perform compression and decompression, each of the Rules being used for both processes. Rules have to be provisioned before SCHC communication takes place. Second, the network side must be able to uniquely identify the devices before choosing the proper Rule, as the same Rule ID might be used in different devices to represent distinct Rules. Devices, on the other hand, do not face this issue since they only hold Rules that apply to themselves [20]. Finally, any LPWAN technology-specific parameters, such as the Data Link layer Word, i.e., the link's minimum data unit (usually a byte), need to be configured.

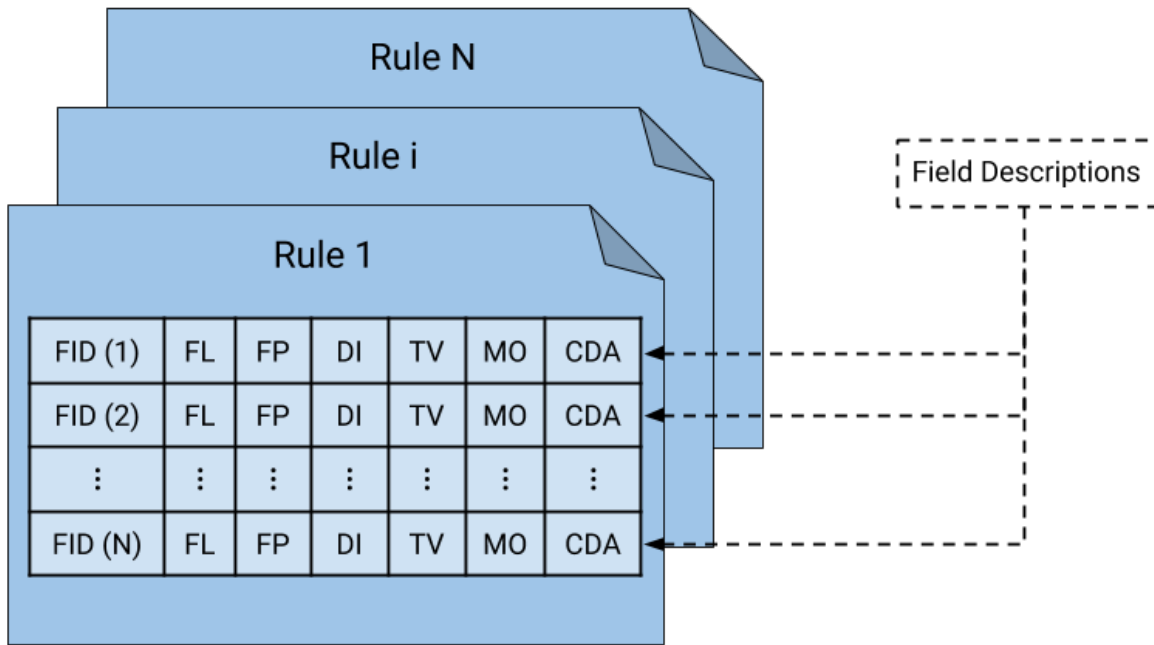


Figure 2.4: SCHC Compression Rule format. Rules are composed of Field Descriptions, each referring to a particular header field from the IPv6 packet. Based on [20].

Once both ends have been properly configured and are prepared for communication, the actual processing and transmission can start. The process's outline is as follows:

1. An IPv6 packet is ready for transmission.
2. Rules are searched for an applicable compression Rule.
3. A compression Rule is selected. Its Rule ID becomes the Rule ID block.
4. The corresponding fields are compressed according to the Rule, producing a (potentially empty) Compression Residue block consisting of the concatenation of individual Compression Residues for each field, in the order they were generated.
5. If no matching Rule was found, the packet header fields remain unmodified. A special Rule may be selected to indicate that no compression was possible, whose Rule ID becomes the Rule ID block.
6. The SCHC Packet is created by joining together the Rule ID block, the Compression Residue block and the IPv6 packet's original payload, in that order. In case no compression was done, the SCHC Packet consists of the no-compression Rule ID followed by the original packet.
7. If the resulting SCHC Packet does not exceed the LPWAN technology's maximum available payload size, it is sent. Otherwise, SCHC Fragmentation should be applied, unless the LPWAN technology specifies its own fragmentation mechanism.
8. If SCHC Fragmentation is used, it produces SCHC Fragments no larger than the LPWAN technology's maximum available payload size, thus ensuring that they can be transmitted through the link.

9. SCHC Fragments (if present) are reassembled on the receiver side, reconstituting the SCHC Packet.
10. The appropriate decompression Rule is identified using the received Rule ID.
11. The corresponding fields are decompressed following the CDAs specified by the Rule, whose application order can differ from the FD order. Computations are done after all other actions. Any decompression parameters needed by a field are taken from the corresponding Compression Residue.
12. Once this process is finished, the output is the original IPv6 packet.

2.4.1 Matching Operators (MOs)

When compression takes place on either side of the link, there are two important factors to take into account: how to choose Rules and how to execute the compression. MOs serve the first purpose, by defining what kind of comparison is carried out between an FV in the packet being compressed and the TV in a Rule. The comparison's result, which can be either True or False, is what determines whether the associated strategy, i.e., the corresponding CDA, is appropriate for a given packet. When all MOs in a Rule give a True result, the Rule is selected. However, some fields can be ignored depending on the context, in which case the comparison is omitted. This will be explained in section 2.4.3.

The values that an MO can take are the following:

- **equal**: The result is True if the FV is exactly the same as the Rule's TV for that field.
- **ignore**: No comparison is carried out, the result is True by default.
- **MSB(x)**: The comparison is made considering only the 'x' most significant bits of FV, and TV. 'x' is of course an integer number smaller than FL. If FL is variable, 'x' must be a multiple of the FL's unit, be it a single bit, a byte, or any other given number of bits.
- **match-mapping**: Here, TV is a list or array of values. Each value has a unique index. The result is True if FV is equal to one of the elements of TV.

2.4.2 Compression/Decompression Actions (CDAs)

CDAs are the actions actually taken in order to effectively compress or decompress a SCHC Packet once a Rule has been chosen or identified, respectively. They result in the omission of fields or their replacement for smaller pieces of data when compressing, and the reconstruction of the fields using TVs, Compression Residues or performing computations when decompressing.

They correspond to the following:

- **not-sent**

- Compression: The header field is completely omitted. No Compression Residue is generated.
- Decompression: The header field is reconstructed using the FD's TV.
- **value-sent**
 - Compression: The header field is sent in full, that is, it is added to the Compression Residue.
 - Decompression: The header field is read from the received SCHC Packet. No additional parameters are required.
- **mapping-sent**
 - Compression: The header field is omitted. Its corresponding index from the TV's matching element is added to the Compression residue.
 - Decompression: Reconstruction of the header field is achieved using the TV's element whose index comes in the Compression Residue.
- **LSB**
 - Compression: Only the 'y' least significant bits from the field are sent in the Compression Residue, the rest are omitted.
 - Decompression: For reconstruction, TV is joined with the bits from the Compression Residue. TV length in bits plus 'y' must equal FL.
- **compute-length**
 - Compression: The header field is omitted.
 - Decompression: The decompressor computes the length of the corresponding field. It can also be used for UDP fields, which is out of the scope of this work.
- **compute-checksum**
 - Compression: The header field is omitted.
 - Decompression: The decompressor computes the corresponding checksum from the data already received. This is intended for computing the UDP checksum field, which is out of the scope of this work.
- **DevIID**
 - Compression: The header field is omitted.
 - Decompression: The Device Interface ID (DevIID) is recovered from the LPWAN Data Link layer frame.
- **AppIID**
 - Compression: The header field is omitted.
 - Decompression: The Application Interface ID (AppIID) is recovered from the LPWAN Data Link layer frame.

2.4.3 Rule Selection

At this point, all the necessary elements for understanding Rule selection have been described. The Rule selection algorithm is presented next.

1. First, the direction of communication must be identified. The SCHC C/D should know where it lies (either device or NS), so based on this information and whether it is performing compression or decompression, it can identify the direction. Uplink corresponds to device compression and NS decompression, and downlink is the other way around.
2. Rules are then scanned line by line for their applicable direction, looking into the DI item of each field. If an FD does not correspond to the current direction, it is ignored. If no FD in the Rule matches the direction, that Rule is discarded and the next one is scanned.
3. When a DI matches the direction, the FP item is then checked against the actual occurrences of the field represented by FID within the packet to be sent. If an FP is not valid, the Rule is discarded.
4. Once DI and FP in a given FD are valid, the MO is calculated. If all MO results are True, the Rule is selected. Otherwise, the following Rule is considered.
5. Finally, the selected Rule's CDAs are executed, resulting in the SCHC Packet.

2.4.4 Padding

Some LPWAN technologies use a minimal data unit, called the 'LPWAN Word', that is greater than a single bit, and normally a byte. Because the SCHC scheme operates on bits, some extra bits might need to be added in order for the LPWAN technology to correctly process and deliver the corresponding frame. When sent, the SCHC Packet will become an LPWAN frame's payload, which must comply with the LPWAN Word length. The extra bits are called 'padding bits', and the process of adding them is called 'padding'. After the compression process involving Rules is finished, the resulting SCHC Packet is padded as needed to an integer multiple of the Word. The number of padding bits is strictly less than the number of bits in a Word. Padding bits have the same value, which can be either 1 or 0 and needs to be specified, since padding bits are considered in MIC calculations. When decompression takes place, after completely processing the SCHC Packet and once the original IPv6 packet has been recovered, only the padding bits (right after the end of the payload) remain unprocessed, and are then simply dropped.

Chapter 3

System Implementation

This chapter covers the implemented functional SCHC scheme in detail. First, the hardware and working setup is described. Then, the complete system design is presented, including its block diagram, inputs and outputs for each block, and the connections between them. Finally, a description is given on the way each block was implemented, explaining the functions that each piece of software accomplishes.

3.1 Equipment and Setup

The whole purpose of the developed system consists in achieving the transmission of an IPv6 packet through a LoRaWAN link, from device to NS. The LoRa link (that is, the physical radiofrequency modulation link) involves a gateway and a device. The LoRaWAN link, i.e, the communication established through the use of the LoRaWAN protocol, involves the device and the NS as the two ends of the link. A network management Application Programming Interface (API) is needed to effectively connect the device to the NS. All these elements are described below.

3.1.1 Microchip RN2903 LoRa Technology Mote

As the LoRaWAN device, a Microchip LoRa Technology Mote is used, which incorporates a Microchip RN2903 LoRa Technology Transceiver Module chip. The RN2903 implements the LoRaWAN Specification version 1.0.2, which enables the use of the LoRaWAN protocol to communicate with an NS. The device is a class A LoRaWAN end-device. This Mote has two possible versions, depending on the region it is intended to operate in: the 868 MHz and the 915 MHz versions. The device used throughout this project is the latter, corresponding to the ISM band available in North America and some countries in South America, including Chile [21]. The Microchip Mote is shown in Figure 3.1.

The device has two basic operation modes: USB and Battery. In USB mode, it is connected



Figure 3.1: Microchip RN2903 LoRa Technology Mote, a device compliant with the LoRaWAN Specification version 1.0.2. It uses the [902 - 928] MHz ISM band.

through a USB cable to a host, which can then issue serial commands to the device. In Battery mode the device is powered on using a built-in switch, and controlled using two push buttons [22]. Hereafter USB mode is always assumed; Battery mode is irrelevant to this work.

Serial Commands

While in USB mode, the device can receive ASCII serial commands coming from a terminal or appropriate application within a computer. There are many commands, allowing different configurations and operations regarding the module's radio, the LoRaWAN protocol, and some memory reading functions from the internal microcontroller. These are named `radio`, `sys` and `mac` commands, respectively. Their scopes and interactions are depicted in Figure 3.2.

The Mote's `mac` commands have to be encoded in ASCII, and trailed with the special characters 'Carriage Return' (<CR>) and 'Line Feed' (<LF>) when sent through a serial interface. They are written in lowercase and each of their arguments is represented as <argument>. Only the most relevant commands for this implementation will be explained below, which are mostly the LoRaWAN-related `mac` commands.

- `sys get hweui`: Outputs the device's EUJ-64, a 64-bit unique identifier. It can be used to build network identifiers, if needed.
- `mac join <mode>`: Joins the device to an available network. Requires providing valid keys for the network, namely NwksKey and AppSKey, as well as a recognized DevAddr.
 - <mode>: Possible values are either 'abp' or 'otaa', respectively representing the

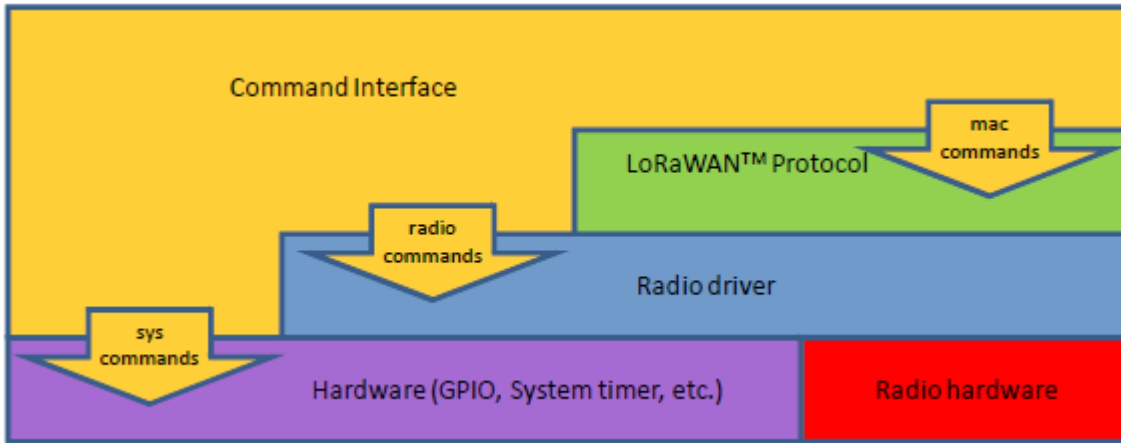


Figure 3.2: RN2903 command interface (yellow) and its relationship to the module’s internal components. Source [23].

ABP and OTAA network join methods.

- `mac tx <txtype> <portno> <data>`: Issues a LoRaWAN message to be sent by the device to the NS. Requires the device to be joined to a network.
 - `<txtype>`: Either ‘`cnf`’ or ‘`uncnf`’, for a ‘confirmed’ message (expecting an ACK from NS) or an ‘unconfirmed’ one (no ACK expected).
 - `<portno>`: A decimal number in the range [1 - 223], representing the device’s port number through which the message will be sent.
 - `<data>`: The LoRaWAN frame payload to be sent, a sequence of bits represented in uppercase hexadecimal format. The number of characters must be even, for completing an integer number of bytes.
- `mac set devaddr <address>`: Sets the device’s network address (DevAddr) to the value input in `<address>`, in uppercase hexadecimal format. DevAddr is provided by the network and required for ABP joining method.
- `mac set nwkskey <nwksesskey>`: Sets the device’s Network Session Key (NwkSKey) to the value input in `<nwksesskey>`, in uppercase hexadecimal format. NwkSKey is provided by the network and required for ABP joining method.
- `mac set appskey <appsesskey>`: Sets the device’s Application Session Key (AppSKey) to the value input in `<appsesskey>`, in uppercase hexadecimal format. AppSKey is provided by the network and required for ABP joining method.
- `mac get devaddr`: Gets the DevAddr current value, in uppercase hexadecimal format.
- `mac get nwkskey`: Gets the NwkSKey current value, in uppercase hexadecimal format.
- `mac get appskey`: Gets the AppSKey current value, in uppercase hexadecimal format.
- `mac set dr <dataRate>`: Sets DR for the next transmission to `<dataRate>`, a decimal value in the range [0 -4]. DR influences the maximum allowed length for the `<data>` argument in `mac tx` command.
- `mac save`: Saves the currently set setting values (since last reset or power-on), such as the DevAddr, NwkSKey, AppSKey and DR, to the device’s EEPROM, so that they survive power-cycling.

3.1.2 Everynet LoRaWAN Gateway

This piece of equipment corresponds to the LPWAN RG, which will forward message's received from the device to the NS. The Everynet LoRaWAN is compliant with the LoRaWAN Specification version 1.1, which supersedes the previous versions by introducing new options and better security management, but is however backwards compatible. It is essentially a GNU/Linux machine with relatively limited resources, which incorporates a built-in LoRa message forwarding application. Busybox, an executable file that provides many standard GNU/Linux tools as compact versions, is also included in the gateway's machine. The `tcpdump` command, which is essential for the Packet Sniffer presented in section 3.2.7, comes among these tools. The Everynet LoRaWAN Gateway can be seen in Figure 3.3.



Figure 3.3: Everynet LoRaWAN Gateway, compliant with the LoRaWAN Specification version 1.1.

Two connectors are available: an Ethernet port and a custom serial port. A Power-over-Ethernet (PoE) adapter is connected to the Ethernet port in order to provide both power and Internet connection to the gateway. A custom cable connects the serial port to a host computer (via USB) to grant Secure Shell (SSH) connection to the gateway's terminal, for debugging and configuration purposes. The SSH connection will be used to issue some commands relevant to the system's functionality. Both the PoE adapter and the custom cable are included with the gateway. A 915 MHz antenna has also been connected for proper radiofrequency communication. Figure 3.4 shows the gateway's connectors.

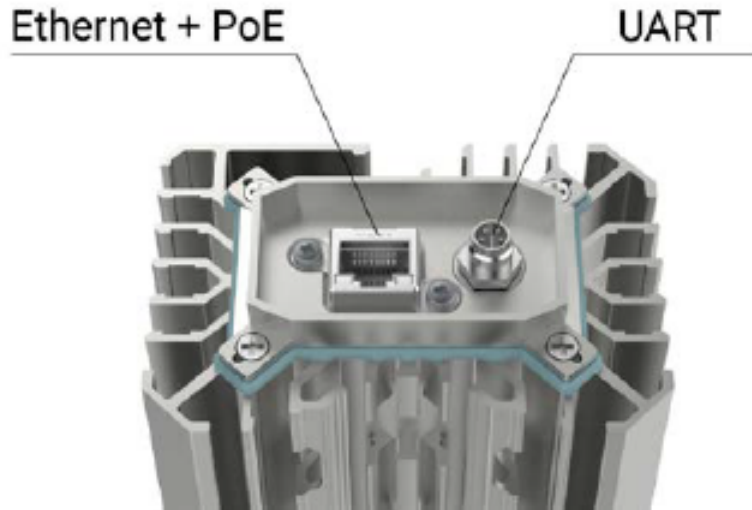


Figure 3.4: Everynet Gateway’s Ethernet and custom serial ports. Source: [24]

3.1.3 Everynet Network Management Platform

Everynet provides a Network Management Platform (hereafter the ‘Everynet Platform’) for device administration and NS settings. It lets the manager configure the join method, the device address and session keys, and see the data exchanged between the NS and a given device, for all devices connected to the network. Session keys and addresses can be randomly updated for better security and network flexibility, and new devices can be created and tagged for easy search and data filtering. The uplink and downlink messages, as well as useful information regarding the communication link are visible as JSON (JavaScript Object Notation) objects, when opened on a message panel located below the settings. Figure 3.5 shows the main configuration interface, while Figure 3.6 shows an opened message on the same platform, at the lower part within the webpage.

3.1.4 Host Computer and Development Tools

A GNU/Linux computer with Ubuntu 16.04 has been used as the host computer connected to the devices for SSH and serial communication, and also as the main programming tool. Python 2.7 and Node.js 10 were installed into the computer for script development, with IDLE and Gedit being used the respective script editors. The `minicom` terminal utility was used to connect with the gateway’s internal machine via SSH.

3.1.5 Required Configuration

In order to establish a LoRaWAN link capable of transporting SCHC Packets, the equipment must be configured in some particular ways. This is done through some serial commands

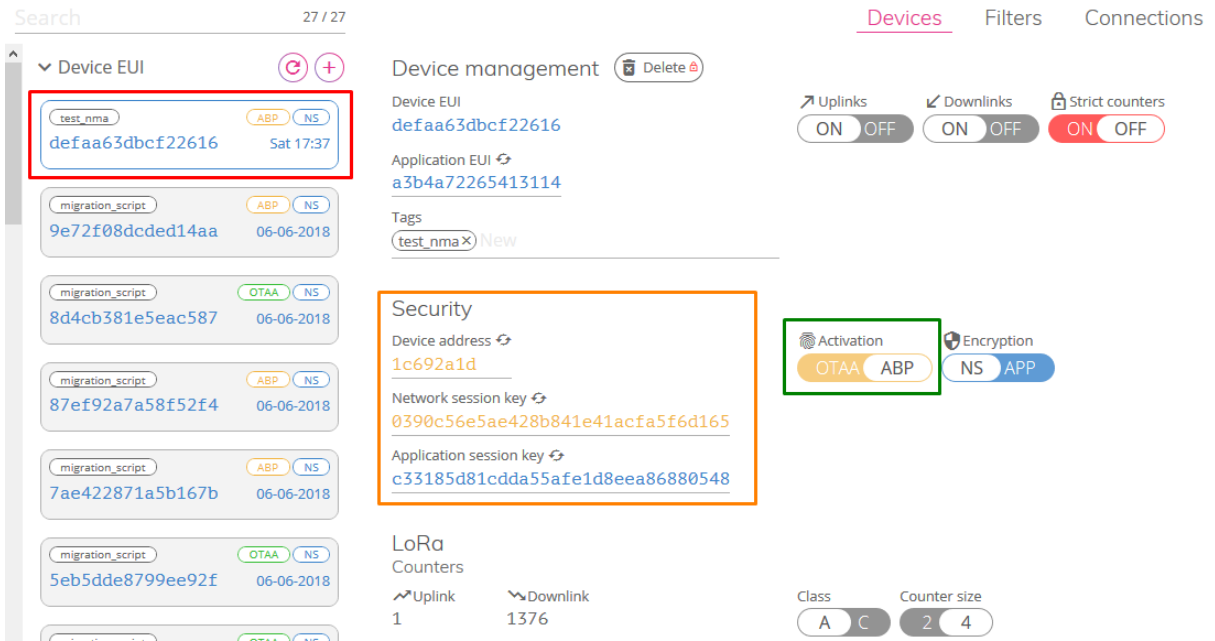


Figure 3.5: Everynet Network Management Platform for device administration. In the red rectangle (left), the current device. Credentials needed for ABP appear in the yellow rectangle (center). On the right, marked in green, the joining method selection switch.

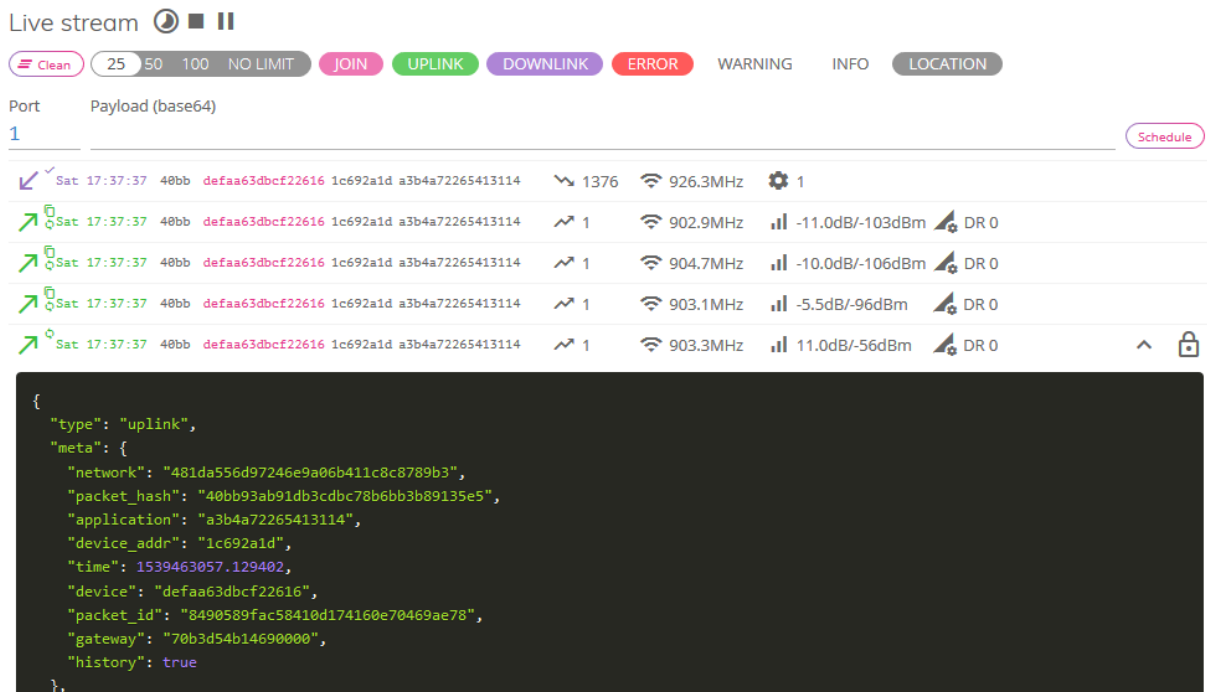


Figure 3.6: Everynet Platform's message panel. It shows the uplink and downlink messages recently exchanged, as well as some other information. Messages can be opened and visualized as JSON objects.

for the Mote in conjunction with Everynet Platform's settings. After entering the Mote's correct settings, the `mac save` command is given for it to remember these settings when reset or power-cycled.

Network Joining

For simplicity, and as the developed test-bed platform does not require much flexibility, the ABP joining method was preferred. ABP requires three pieces of information to join a device with a network: a DevAddr identifier, and the NwkSKey and AppSKey security session keys. All three codes are randomly generated by the Everynet Platform, and are then configured into the device with the `'mac set devaddr/nwkskey/appskey'` commands, correspondingly. Once the variables have been configured, the `'mac join abp'` command is issued to execute the joining procedure. When the device replies with `'accepted'` upon receiving the command, it has effectively been connected to the network.

LoRaWAN Datarate

Tests for verifying the system's functioning assume a 100-byte long IPv6 packet, whose 40-byte long header will be compressed using SCHC. This might actually result in making the header one byte longer in some cases, but reduces it to only one byte in the best-case scenario, and somewhere inbetween in most cases. Thus, the length range for the actual SCHC Packet is [61 - 101] bytes. As explained in chapter 1, LoRaWAN maximum available application payload (where the SCHC Packet will be put) is dependent on the device's current DR. The [61 - 101] bytes range requires DR 3 or DR 4, both allowing at least 222 bytes for the application payload. Between the two, DR 3 implicates less power consumption, so it is preferred. Therefore, the LoRaWAN Mote will be configured with DR 3, using the `'mac set dr 3'` command.

Network Message Confirmation

As a way to verify and debug the device-NS connection, the `'cnf'` transmission type is used for all `'mac tx'` commands, so the device will wait for an ACK message from the NS after every transmission, and reply with `'mac_tx_ok'` if received or `'mac_error'` otherwise.

3.2 Functional Blocks

In this section, the system design is described. It comprises several functional blocks with well defined inputs and outputs. The way they interact and connect is explained in the following subsections. The system's block diagram is also given. As this work has been done as a proof of concept, not all the system's blocks are fully integrated, but each of them is self-contained.

3.2.1 System Overview

Simply put, the system creates an IPv6 standard packet, compresses it using the SCHC Compression scheme, sends it through a LoRaWAN uplink, recovers it on the network end (actually, before reaching it), applies SCHC Decompression to the received SCHC Packet inside the LoRaWAN payload, and verifies that the recovered IPv6 packet matches the original one. The steps are the following:

1. The Packet Generator creates a standard IPv6 packet (no extension headers).
2. The SCHC Compressor compresses the packet, producing a SCHC Packet.
3. The SCHC Packet is padded to a multiple of bytes and passed onto the LoRaWAN device for uplink transmission.
4. The LoRaWAN device sends the packet to the gateway (which forwards it to the NS) and expects an ACK from the NS (only to verify that the link is working and that the packet was properly received).
5. The Packet Sniffer (in the local network) catches the packet forwarded by the gateway, whose IP connection is already identified. The packet's payload is extracted.
6. A LoRaWAN Payload Retriever takes the sniffed packet's payload, corresponding to the complete LoRaWAN frame, and uses a LoRaWAN decoder to obtain the LoRaWAN decrypted payload, which is in fact the SCHC Packet.
7. The SCHC Packet undergoes SCHC Decompression, thus recovering the original IPv6 packet.
8. A Message Verification block receives both the original IPv6 packet from the Packet Generator and the packet recovered through decompression, and compares them, validating the result.

The functional block diagram is illustrated in Figure 3.7 below.

Most blocks were implemented in Python, although JavaScript modules and GNU/Linux Command Line Interface utilities were also used. As a notable resource, the `bitstring` Python module [25] was utilized for ease and practicality in handling bits. All packets are ultimately represented as `bitstring` objects, and computations involving them take advantage of the `bitstring` module's built-in functions and methods. Another bit-handling module, the `bitarray` module, was also used, mainly for easy creation of random `bitstrings` in combination with the module just mentioned. As in Python's natural representation, the prefixes '0b' and '0x' are used to denote binary and hexadecimal strings or data, respectively.

The most relevant `bitstring` methods used in this implementation are:

- `bitstring.uint`: Returns a decimal number equal to the `bitstring` object interpreted as an unsigned integer.
- `bitstring.bin`: Returns a string with '1's and '0's representing the `bitstring` object in binary format.
- `bitstring.hex`: Returns a string representing the `bitstring` object in lowercase hexadecimal format.

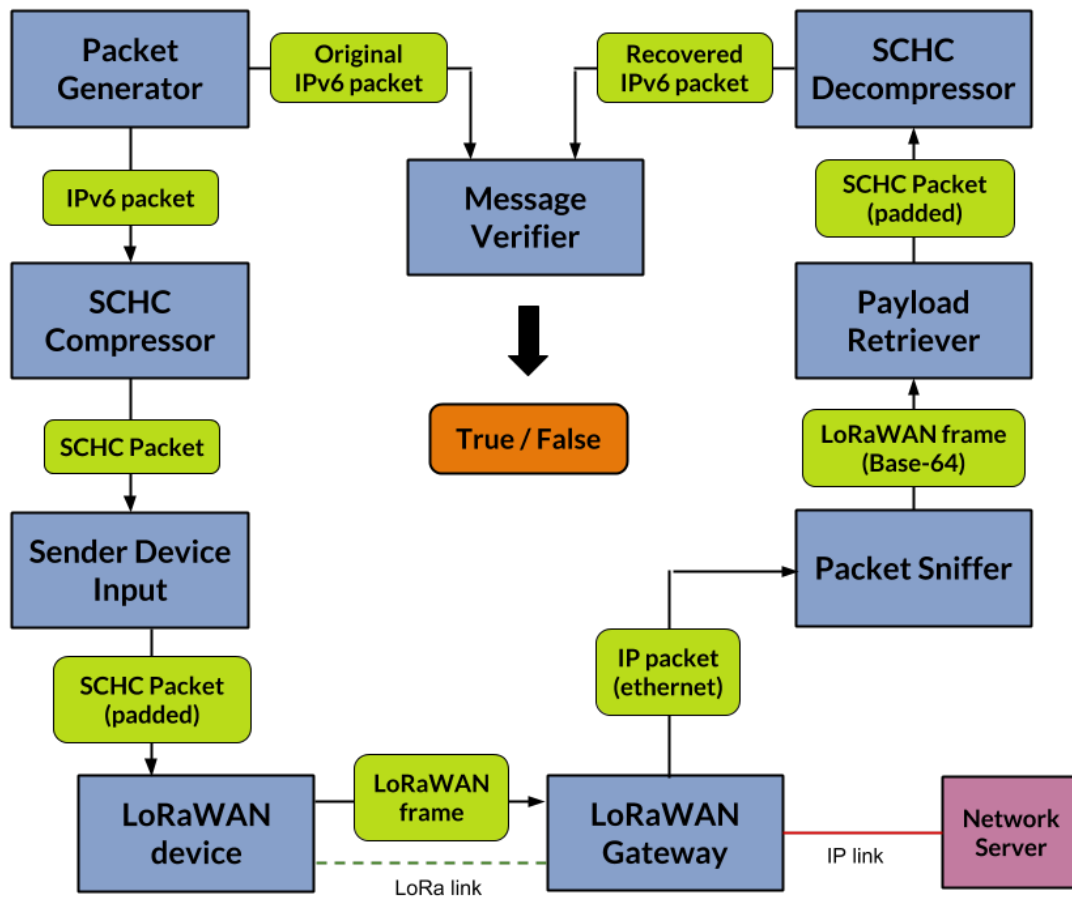


Figure 3.7: The complete system’s functional block diagram, including inputs, outputs and connections between blocks.

- `bitstring.len`: Returns a decimal number equal to the `bitstring` object interpreted as an unsigned integer.

3.2.2 IPv6 Packet Generator

This block generates a standard IPv6 packet with a valid fixed header and no extension headers. The header’s length is 40 bytes, and the payload length has been fixed to 60 bytes, so that the total packet size is 100 bytes. For completeness, the header’s fields are: Version, Traffic Class, Flow Label, Payload Length, Next Header, Hop Limit, Source Address, and Destination Address. For details, refer to [15]. Payloads smaller than 60 bytes can also be used, but were not tested. The payload’s content is a dummy; it does not represent any particular information, but only random bits as a generic application payload.

This block uses the `bitstring` module’s `Bits` object class to represent the IPv6 packet. It essentially creates the fixed header fields, a random payload, and concatenates all the elements in the proper order to produce the packet. It also creates a friendly visual representation of

the packet as a block of 4-byte lines for debugging purposes. The header fields' hexadecimal representations and values (always interpreted as unsigned integers) used in the system testing are:

- Version: 0x6 - (6)
- Traffic Class: 0x00 - (0)
- Flow Label: 0x00000 - (0)
- Payload Length: 0x003c - (60 [bytes])
- Next Header: 0x11 - (17 [UDP])
- Hop Limit: Random value in the range [1 - 256]. Ex.: 0xb5 - (181)
- Source Address: 0x5555 5555 5555 5555 5555 5555 5555 5555 - (irrelevant)
- Destination Address: 0xaaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa - (irrelevant)

Block summary:

- **Input:** None.
- **Output:** A standard IPv6 packet consisting of the fixed 40-byte header followed by a 60-byte random payload.
- **Programming Language:** Python.

3.2.3 SCHC Compressor

The IPv6 packet received by this block is compressed using the SCHC mechanism as described in chapter 2. The basic mechanism was implemented in a Python script using 4 Rules and all 8 IPv6 header fields per each. In order to process the Rules for selection, they have been put together in a list (array-like) object. The Rules themselves are as well lists that contain 8 elements, the FDs, each of which corresponds to an `OrderedDict` (OD) object. An OD is just a regular dictionary object, only it preserves the order in which the entries were first entered when the object was created, for ease of use and friendly visualization. A dictionary is a collection of (key, value) pairs, where key is normally a string and value can be any object. The values are accessed using their corresponding keys as the indexes, as in `dict[key1] = value1`, and so on. An OD works just the same.

Thus, an FD within a rule has the following structure, given in Table 3.1:

"FID"	"FL"	"FP"	"DI"	"TV"	"MO"	"CDA"
fid	fl	fp	di	tv	mo	cda

Table 3.1: Field Description, a row within a Rule. The double-quoted fields are the keys, used to refer to their respective values, in lowercase below.

A for loop goes through the Rules collection, where a nested for loop goes through FDs in each Rule. From there onwards, the Rule selection algorithm is applied using the usual comparison and conditional execution strategies. In the first three Rules, DI was defined as 'Up' for all FDs, while in the last one all DIs were defined as 'Dw' with the intention of

testing Rule failure and the no-compression case. As only the standard header is used in this project, all FPs are equal to '1', because no field is optional nor repeated.

Rules were designed using three groups where the header fields are organized, and which define the MO-CDA strategy for each of them. The groups are: 'known', 'from set' and 'unknown'; their strategies are respectively 'equal' - 'not-sent', 'match-mapping' - 'mapping-sent' and 'ignore' - 'value-sent'. The 'MSB(x)' MO and the 'LSB' CDA were included as potential cases within the script, but the time needed to create their related functions for bitwise calculations exceeded the time available for this work. The same applies for the rest of the CDAs that were not implemented, except for the `compute-checksum` CDA, which is targeted at UDP and is therefore out of scope. The `AppIID` and `DevIID` CDAs represent a way of avoiding address transmission, the longest fields in the IPv6 header, and are therefore relevant to SCHC's performance. Their implementation however required a little more research and programming, in order to correctly simulate different application and device addresses within the context.

Block summary:

- **Input:** An IPv6 standard packet (from Packet Generator).
- **Output:** A SCHC Packet, corresponding to the concatenation of a Rule ID, a (possibly empty) Compression Residue, and the IPv6 packet's original payload. Its size ranges from 482 bits (60.25 bytes) to 802 bits (100.25 bytes).
- **Programming Language:** Python.

3.2.4 Sender Device Input

The block takes the SCHC Packet coming from the SCHC Compressor, adds padding bits as needed and hands it onto the LoRaWAN Mote through the `'mac tx'` command. Padding is managed here and not in the SCHC Compressor block because it is LPWAN technology-specific, so this leaves the SCHC Compressor block compatible with potential implementations using technologies different from LoRaWAN.

The first thing this block does is selecting the communication port that corresponds to the Mote, and opening a serial port using the `serial` module. It then creates a string representing the command that will be issued, namely `'mac tx cnf 1 <loradata>'`, where `<loradata>` is the padded SCHC Packet in uppercase hexadecimal format. The message is a confirmed one in order to verify that the NS has received it (or not). The port number is irrelevant to this implementation, 1 is used by default. As discussed in section 3.1.5, DR 3 (or DR 4) is needed to transmit the 100 bytes packet. Therefore, it must be configured into the Mote before sending the message. However, the device's DR is adjusted by the NS with a command coming in each ACK it sends back to the device. This behavior could not be avoided, not even by disabling the NS ADR function in the platform and the Mote's ADR with the corresponding command.

The solution found was to check for the device's DR value before sending each message, and changing it to DR 3 whenever it was neither 3 nor 4. Thus, the operating loop of the

block does the following:

- Checks the DR value with the ‘`mac get dr`’ command.
- If DR is not 3 nor 4, the device adjusts it through the ‘`mac set dr 3`’ command.
- After ensuring that DR is appropriate, the device sends the message to the NS using the ‘`mac tx`’ as described.
- The device then waits for the NS’s ACK and prints the reply.
- After this, the loop is restarted.

The loop can be interrupted at any moment by the user, by pressing `ctrl-C`. This allows re-running the former blocks to produce a different SCHC Packet, although only the random values are reset, and the Rule selection conditions will be the same. This is meant to be enhanced into further packet randomization, so that Rule selection is not predictable *a priori*.

Block summary:

- **Input:** A SCHC Packet, not necessarily an integer number of bytes long.
- **Output:** A padded SCHC Packet with an integer number of bytes, represented in ASCII hexadecimal format. Its maximum possible length is 101 bytes.
- **Programming Language:** Python.

3.2.5 LoRaWAN Device

The LoRaWAN device is also part of the previous block’s loop (as it processes and answers the `mac` commands), but is conceptually separated here for block-wise clarity. In a later version of this implementation, the loop behaviour should be modified to involve all the blocks in the system, that is, a full compression-transmission-decompression-verification cycle should be executed per each newly generated packet. The LoRaWAN device block starts once the ‘`mac tx`’ command has been issued, after which the command is received and processed by the device, and the corresponding LoRaWAN frame containing the SCHC Packet is generated. The message is then translated into a physical LoRa modulation signal and transmitted through the antenna to reach the gateway.

- **Input:** An ASCII hexadecimal representation of a SCHC Packet, padded to an even number of characters (integer number of bytes). It is issued with the ‘`mac tx`’ command and therefore interpreted as a LoRaWAN frame payload.
- **Output:** A LoRaWAN frame containing a padded SCHC Packet as the payload. Sent towards the LoRaWAN gateway using LoRa radio modulation.
- **Hardware:** Microchip LoRaWAN device.

3.2.6 LoRaWAN Gateway

The gateway receives the LoRaWAN frame sent by the device and encapsulates it inside an IP packet, which will be sent through the Internet to the NS. The NS will reply if it

corresponds, but this is not part of the block's function. The custom debug cable connects the gateway with the host computer, which allows access to the gateway via an SSH session in the computer's terminal. This will allow the forwarded packet (gateway to NS) to be caught by the next block.

- **Input:** An ASCII hexadecimal representation of a SCHC Packet, padded to an even number of characters (integer number of bytes). It is issued with the 'mac tx' command and therefore interpreted as a LoRaWAN frame payload.
- **Output:** An IP packet forwarding a LoRaWAN frame to the NS for processing. Sent through Ethernet interface.
- **Hardware:** Everynet Gateway.

3.2.7 Packet Sniffer

The Packet Sniffer's function is to identify and access the packet sent from gateway to NS, and then finding the LoRaWAN frame. It has been implemented by using the tcpdump command from inside the gateway's terminal, through the SSH session. The interface tun0 has been identified as the network tunnel between gateway and NS. Packets going through this connection are therefore printed in ASCII and observed, until finding the plaintext word 'data', which indicates that the subsequent portion of the message corresponds to the LoRaWAN frame, which is coded in Base-64 format. It is then copied to be introduced into the next block. A captured (partial) packet can be appreciated in Figure 3.8. The LoRaWAN data section is marked in red.

```

19:52:05.186104 IP (tos 0x0, ttl 64, id 32042, offset 0, flags [DF], proto UDP
(17), length 1444)
 192.168.64.12.42666 > 10.0.2.142.1680: [udp sum ok] UDP, length 1416
 0x0000: 4500 05a4 7d2a 4000 4011 aadc c0a8 400c  E...}*@. @.....@.
 0x0010: 0a00 028e a6aa 0690 0590 1698 0207 b900  .....
 0x0020: 70b3 d54b 1469 0000 7b22 7278 706b 223a  p..K.i..{"rxpk":
 0x0030: 5b7b 2274 6d73 7422 3a33 3137 3037 3435  [{"tmst":3170745
 0x0040: 3036 382c 2274 696d 6522 3a22 3230 3138  068,"time":"2018
 0x0050: 2d31 302d 3135 5431 393a 3532 3a30 352e  -10-15T19:52:05.
 0x0060: 3137 3239 3732 5a22 2c22 7473 7263 223a  172972Z","tsrc":
 0x0070: 2273 7973 7469 6d65 222c 2263 6861 6e22  "systemtime","chan
 0x0080: 3a36 2c22 7266 6368 223a 312c 2266 7265  :6,"rfch":1,"fre
 0x0090: 7122 3a39 3034 2e37 3030 3030 302c 2273  q":904.700000,"s
 0x00a0: 7461 7422 3a31 2c22 6d6f 6475 223a 224c  tat":1,"modu":"L
 0x00b0: 4f52 4122 2c22 6461 7472 223a 2253 4631  ORA","datr":"SF1
 0x00c0: 3042 5731 3235 222c 2263 6f64 7222 3a22  0BW125","codr":
 0x00d0: 342f 3522 2c22 6c73 6e72 223a 2d31 322e  4/5","lsnr":-12.
 0x00e0: 302c 2272 7373 6922 3a2d 3130 322c 2273  0,"rssi":-102,"s
 0x00f0: 697a 6522 3a31 362c 2264 6174 6122 3a22  ize":16,"data":
 0x0100: 6742 3071 6152 7943 6a67 4146 4177 4732   "gB0qaRyCjgAFawG2
 0x0110: 754a 5a65 3251 3d3d 227d 2c7b 2274 6d73   uJZe2Q=="}, {"tms
 0x0120: 7422 3a33 3137 3037 3435 3036 382c 2274  t":3170745068,"t
 0x0130: 696d 6522 3a22 3230 3138 2d31 302d 3135  ime":"2018-10-15
 0x0140: 5431 393a 3532 3a30 352e 3137 3332 3235  T19:52:05.173225
 0x0150: 5a22 2c22 7473 7263 223a 2273 7973 7469  Z","tsrc":"systi
 0x0160: 6d65 222c 2263 6861 6e22 3a33 2c22 7266  me","chan":3,"rf
 0x0170: 6368 223a 302c 2266 7265 7122 3a39 3033  ch":0,"freq":903
 0x0180: 2e33 3030 3030 302c 2273 7461 7422 3a31  .300000,"stat":1

```

Figure 3.8: Packet captured using tcpdump command. LoRaWAN data, in Base-64 format, is shown in the red rectangle.

Block Summary:

- **Input:** An IP packet traveling from gateway to NS.
- **Output:** The complete LoRaWAN physical data encoded as a Base-64 string.
- **Utility:** GNU/Linux terminal `tcpdump` command.

3.2.8 LoRaWAN Payload Retriever

LoRaWAN Payload Retriever block does the following: it takes the copied LoRaWAN frame from the Packet Sniffer and gives it to a command line utility named `devlora-packet-decode`, which decodes the frame and obtains the decrypted payload, printing it in the terminal. The `devlora-packet-decode` command is based on the `lora-packet` project by Anthony Kirby [26], which is written in JavaScript. The utility has been internally provided with the `NwkSKey` and `AppSKey` used in the Mote and platform, so it can perform the decrypting actions that use these keys. The retrieved LoRaWAN payload corresponds to the SCHC Packet, adequately padded.

Block summary:

- **Input:** A Base-64 encoded full LoRaWAN physical frame.
- **Output:** The raw LoRaWAN decrypted payload in hexadecimal format, which in this case represents a SCHC Packet.
- **Programming Language:** Node.js JavaScript.
- **Utility:** GNU/Linux terminal `devlora-packet-decode` command.

3.2.9 SCHC Decompressor

The SCHC Decompressor receives the retrieved SCHC Packet coming from the LoRaWAN link, and performs decompression as per chapter 2. It is implemented in a way similar to the SCHC Compressor, as it performs the reciprocal actions. Both blocks share the same Rules and the same information about them. The SCHC Decompressor starts off by reading the first `rb` bits, where `rb` is the number of bits needed to encode the Rule ID. Rule IDs are encoded as unsigned integers, using the least number of bits possible, i.e., the minimum number of bits required for encoding the highest Rule number, in this case 3. Upon reading the first `rb` bits, the Rule is identified. Its CDAs are then applied, in the same order that the FDs are read, that is, the order of the IPv6 header fields.

A ‘reader’, i.e., a counter for already read bits, is used in order to keep track of the fields that have been reconstructed so far during the process, and to correctly get and use the Compression Residue elements. After this process, the IPv6 packet has been recovered. The only unprocessed bits are the padding bits (if any), at the end of the payload. They are conceptually ‘dropped’, which in practice means that nothing else is done with them.

Block summary:

- **Input:** SCHC Packet represented in hexadecimal format.
- **Output:** A standard IPv6 packet recovered by means of SCHC Decompression.
- **Programming Language:** Python.

3.2.10 Message Verifier

Finally, the recovered packet, coming from the SCHC Decompressor, and the original packet, coming from the Packet Generator, are compared for equality, as `bitstring` objects. It prints the question “Decompressed packet = original packet?” If they in fact are equal, the result is ‘True’, otherwise, it is ‘False’. The result is then printed below the question. The Message Verifier has been implemented within the same script as the SCHC Decompressor, given that its function is quite simple.

In a more realistic scenario where a comparison between the generated packet and the recovered one is wanted, there would be essentially two options: i) Provisioning both ends with an identical copy of a set of previously generated packets; or ii) Providing the generated packets to the Decompressor end by reliable out-of-band means. In any case, the integrity and identity of the packets being compared must be ensured. Studying the actual feasibility of either of these methods is however out of the scope of this work.

Block summary:

- **Inputs:** An originally generated IPv6 packet (from Packet Generator) and a recovered IPv6 packet from SCHC Packet decompression.
- **Output:** The result of an equivalence comparison between the two inputs, either ‘True’ or ‘False’.
- **Programming Language:** Python.

Chapter 4

Analysis

In this chapter, the implemented system's features will be analyzed, emphasizing its scopes and limitations. In section 4.1 a qualitative evaluation of the application scenarios for SCHC over an LPWAN network is carried out. Section 4.2 gives a quantitative compression efficiency evaluation for representative LPWAN scenarios, and a direction for a future complete evaluation. The most significant achievements accomplished by this work are summarized in section 4.3. Finally, in section 4.4, the main limitations of the system are discussed and proposed as future enhancements.

4.1 Recommendations of Usage Scenario and Comparison

So far, the SCHC mechanism's theory and the implementation hereby presented have been explained in detail. The reader should have a relatively clear idea of the requirements that an implementation of SCHC imposes, regarding both hardware and software. At this point, it is relevant to think more deeply of these requirements and how they relate to the more conventional LPWAN scenarios, what the differences between them are and which are the implications. Therefore, it is of key importance to recall what the main features of LPWAN technologies are:

- Low power consumption
- Wide communication coverage
- Low transfer rate
- Constrained low cost devices

As discussed, these features are designed to fit into a particular type of applications, where devices are expected to stay on-site for several years on a single battery charge, while conveying small periodical amounts of data and staying in sleep-mode most of the time. They are also intended to be deployed in big numbers for several monitoring-like services. The implementation of SCHC in an LPWAN network does not exactly fit in this conventional

scenario. While twisting the requirements in some ways, it provides new possibilities for more network-sensible and device-centered services, all at the same time. It also opens a new direction leading to more complex services by overcoming the interoperability issue between different LPWAN technologies.

With these factors in mind, a new scenario for SCHC-enhanced LPWAN networks is created, and the need for a comparison of advantages and requirements arises. Let us state what the additional requirements for a SCHC-enhanced LPWAN network are:

- Embedded computer for SCHC C/D and TCP/IP stack on device side
- IPv6 address allocation for devices
- Implementation of SCHC C/D on network side

In this scenario, depicted in Figure 4.1, the SCHC mechanism is implemented in an embedded computer due to the higher memory and processing capabilities demanded, too high for a device of constrained nature such as an LPWAN mote. It is the embedded computer that will process actual data packages and hold the IPv6 address. Meanwhile, the mote will act as the Network Interface Controller, being directly connected to the embedded computer for SCHC Packet input and output, while transmitting and receiving the corresponding LPWAN frames. All elements traditionally connected to the LPWAN device, such as sensors, actuators and other functional units, are now connected to the embedded computer. As a comparison, Figure 4.2 shows what an equivalent conventional scenario would look like.

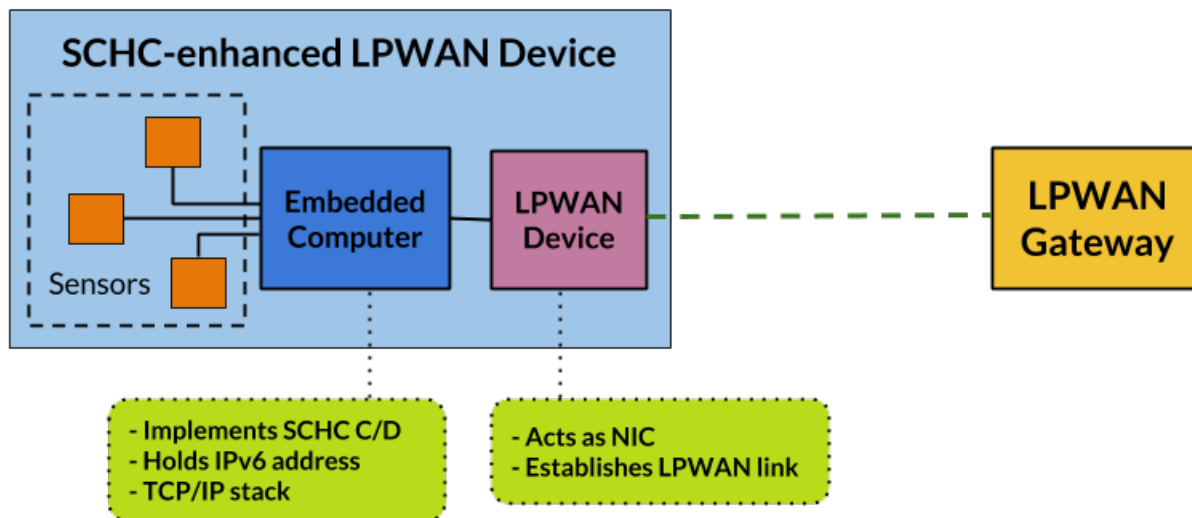


Figure 4.1: SCHC-enhanced LPWAN scenario. An embedded computer is directly connected to a plain LPWAN device, both acting together as a SCHC-enhanced LPWAN device. The embedded computer implements the SCHC C/D and holds the IPv6 address, while the conventional LPWAN device acts as the node’s NIC, establishing the link with the RG. Sensors and actuators are controlled by the embedded computer.

Each of these scenarios holds distinct advantages and disadvantages, which are listed in Table 4.1. The two scenarios should be seen as complementary, and not as one being better than the other. Which scenario fits a desired application best is something that needs to be

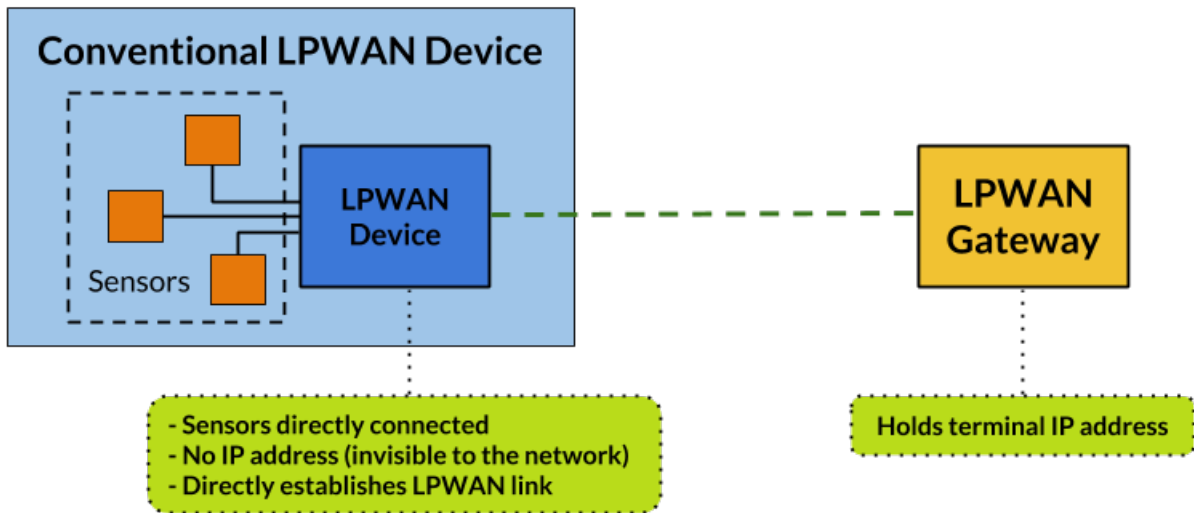


Figure 4.2: Conventional LPWAN scenario. A stand-alone LPWAN device directly communicates with an RG, which holds an IP address. The device is invisible to the network.

looked into and is not trivial. This ultimately expands the possibilities that the IoT universe can offer by providing a new dimension for development.

Scenario	Advantages	Disadvantages
SCHC-enhanced	Header compression Network visibility Interoperability Transparent system Architecture flexibility Custom management Single device control UDP/CoAP and TCP/HTTP	Potential higher power consumption Higher cost per node
Conventional	System simplicity Known low power consumption Plug and play	Invisible to the network Proprietary system UDP/CoAP constrained

Table 4.1: SCHC-enhanced and conventional LPWAN scenarios compared. The advantages and disadvantages for each scenario are listed.

4.2 Performance Evaluation

In this work, a functional proof of concept for a SCHC implementation over a LoRaWAN network has been achieved, and the system's performance has been preliminarily studied. Two typical packet sizes in IoT have been used: 100 bytes for basic working tests and 60 bytes for compression efficiency under three representative IoT scenarios. Afterwards, a robust set of tests and a direction for future performance evaluation is hereby given.

4.2.1 Efficiency of Packet Compression

The system’s performance has been studied by applying the following methodology:

- The most frequent IPv6 packet sizes have been researched. Sizes 60 and 100 bytes have been selected as the target packet sizes, due to high frequency and compatibility with the system’s scenario.
- DR 3 has been fixed through the whole test.
- Three different contexts that are reasonable for a generic LPWAN network have been defined. The appropriate Rules for each of them were designed and implemented.
- For each context, 100 random packets have been generated and run through the system. The mean percentage of compression achieved has been calculated for each case.

A 2007 technical report by the University of Southern California shows that between 40% and 80% of total packets transmitted over the Internet are less than 100 bytes long, with a mode at 40 bytes [27]. This size represents the whole of a packet (header and payload) and packets involved were mostly IPv4. As IPv4 header is 20 bytes long, this means 20 bytes for the payload. If we make this equivalent to the IPv6 case, a total packet length of 60 bytes is obtained. Therefore, 60 bytes was selected as one of the target packet sizes for the test. In order to cover the upper limit of the same packet size range, 100 bytes has also been studied. Although in the case of 60-byte packets DR 2 would also be viable (see section 3.1.5), DR 3 has been maintained with both tests, for the sake of simplicity and fair comparison.

Three contexts were chosen considering what some common LPWAN network communication scenarios might be. For each of the contexts, the IPv6 header fields were separated in three categories by level of knowledge: 1) already known value; 2) value belongs in a well-known set of values; and 3) value is completely unknown. A context always corresponds to a Rule being implemented in the system. Table 4.2 describes the three contexts used.

Context	Fields		
Rule number	Known	From set	Unknown
Rule 0	V - TC - FL - PL - NH - HL - SA - DA	-	-
Rule 1	V - TC - FL - NH - HL	SA - DA	PL
Rule 2	V - TC - FL	NH	PL - HL - SA - DA

Table 4.2: Context rules for performance evaluation test.

The first context (Rule 0) corresponds to an ideal context where all information regarding the communication taking place is already expected: mainly the length of packets exchanged (which is fixed), the transport protocol (none in this case), the hop limit established for the link, and the source and destination nodes. This context would represent a perfectly predictable traffic for a static application that sends the same kind of information all the time. The second context (Rule 1) fits in a more common LPWAN scenario where there are two well-known sets of nodes exchanging information, and any node in a group can communicate with any other node in the second group. Both sets were defined as composed by 1000 nodes each. The protocol and general communication link information is known, but the packet size is variable. Finally, the third context (Rule 2) would represent a much

more dynamic environment, where a node within the network can receive messages from a node which is not previously known or predictable (a new user establishing a session in an application, for example). The transport protocol belongs to the set {None, UDP, TCP} but is not predictable, and the packet size and established hop limit are variable. In all Rules, the version field (V) is always known (IPv6 is constantly used), and the traffic class and flow label are considered known (see [15] for details).

As there are three possible Rules, the Rule ID is composed of 2 bits. These will add to the compressed header, thus distorting the ‘base’ compression percentage a little. This gives place to what is denominated here as the ‘real’ compression percentage, which will depend on the amount of Rules used. It also depends on the amount of indexes in a given well-known set.

Table 4.3 summarizes the compression percentage for the whole 60-byte packet in each of the three given contexts.

Context Rule	Real packet compression percentage
Rule 0	66.25%
Rule 1	58.75%
Rule 2	7.5%

Table 4.3: Real packet compression percentage for each proposed context.

The compression rates achieved are considerable, especially in the first two cases. The big difference observed for case three arises from the fact that in this scenario, the source and destination addresses are completely unknown, and as they are 16 bytes long each, together they account for 80% of the whole header. Thus, addresses are the most important fields impacting the compression rate, although case three is extremely uncommon by virtue of how LPWAN networks are designed and how they make use of IPv6 addresses. As discussed in section 4.4, IPv6 addresses within the LPWAN network in this work are not properly used, and are proposed as a future work improvement.

4.2.2 Future Evaluation Plan

Performance should be first evaluated by establishing a fixed amount of packets, ‘N’, and a fixed DR. N copies of the same packet, which will become a SCHC Packet of length ‘L’, should be transmitted through the link using the same DR. Then, N random valid LoRaWAN frames with payload length L should be transmitted. Then, the rate of successfully received messages in each case should be compared. This will result in a measure of how SCHC Compression affects the LoRaWAN link, which is expected to be negligible. This way, subsequent evaluations involving the SCHC Packets’ intrinsic characteristics will have a base success rate to compare with. This step has been omitted.

Next, performance should be evaluated applying the following general study schemes:

- For a fixed IPv6 packet size, study the mean compression achieved by SCHC, given a well defined header field randomization that represents a common case for an LPWAN

network.

- For a given DR, and a given packet size range, study the minimum compression needed in order for the packet to be transmittable after SCHC Compression. Then study how often that level of compression is actually reached, and what are the main factors involved.
- For a given packet profile, as in which header fields are usually known, unknown or somehow compressible, study which SCHC strategies are more effective. Also, which strategies are more generic and which ones are more specific, in a common case scenario.
- Study how Rules can be ordered in the most effective way, depending on the known network/traffic context and the strategies each of the Rules applies.

4.3 Summary of Achievements

A few important milestones are achieved through this work. First, although the implemented system has some limitations (they will be discussed in the next section), the system's purpose, namely an experimental functional implementation of SCHC over a well established LPWAN technology, has been successfully fulfilled. Despite this being a basic SCHC version, some of its characteristic features have been tested and showed the expected behavior. SCHC's Rule concept was correctly understood and applied, and some functional 'details' such as padding and datarate management performed as intended within the major system.

This work also represents a multi-factored development involving many different resources and tools, which have been properly connected in a way that operates as expected. Telecommunication concepts, such as IoT and the layered communication model have been put together with network sniffing techniques and GNU/Linux systems utilities, giving a strong example of the wide range of resources that exist within the Information and Communication Technologies field and what kind of complex systems can be created when they are combined. In fact, all designed blocks within the system worked as expected even if they were not fully integrated, despite some of them being implemented using different programming languages, and relying on particular hardware-software interactions.

Finally, this project managed to create a working experimental test-bed platform for the IETF's SCHC proposal, starting only from the theory and isolated resources. This platform can be extensively enhanced, allowing a fully-featured SCHC implementation with a rather small amount of relatively simple modifications. Once that has been accomplished, SCHC could be properly studied and experimented with.

4.4 Future Work

The developed system lacks some features that are ultimately necessary to carry out a proper complete evaluation. The implemented system presents many limitations, most of which are a consequence of the system's scope, while some have arisen as a result of time

constraints. At its most essential level, the system is conceived as a test-bed platform for a SCHC basic implementation, a first proof of concept with much space for improvement and functionality expansion. The most notable limitations are related to the size of packets tested, the Rules not testing all possible CDAs, the way addresses are managed, and the system's setup being uplink-only. All of these features are hereby proposed as future work for improving the system.

4.4.1 Packet Size and Datarate

Only one fixed IPv6 packet size was used, so the interaction between SCHC's performance cases and the constraints placed by the link has not been studied to its ideal extent. Varying the packet size, while exploring all datarate combinations, would be an interesting and relevant way to study in which cases SCHC allows a successful transmission under the current link constraints, and to what extent it is compatible with a technology's characteristic maximum message sizes. A mechanism for intelligent and dynamic datarate selection based on the packet size and the compression achieved could be developed, even in the presence of power or datarate adjustments enforced by the NS. This last factor should also be explored further, as it has not yet been fully understood by the author.

4.4.2 Rules

The Rules used in the implementation lack the use of some relevant constituents of a SCHC full version, namely the `MSB(x)` MO and the `LSB`, `compute-length`, `DevIID` and `AppIID` CDAs. Functions for including these elements were intended to be created, but the time to do so exceeded the available resources. The way a packet is generated could also be modified in a way that might result in different Rules being selected depending on the packet. Thus, Rules have not been fully explored and their actual performance regarding different contexts cannot yet be evaluated.

4.4.3 Address Management

Addresses are managed in a way that does not fully represent the way networks and devices organize in IPv6, but rather reflects a very basic and generic network behavior. This, which is tightly related to the lack of `DevIID` and `AppIID` CDAs, strongly affects the performance that SCHC Compression can achieve, as addresses are the most massive fields in the IPv6 header, and their transmission could be completely avoided in many cases. This is considered the most critical feature to be included in a future version.

4.4.4 Downlink Absence

Probably the most fixed limitation of the whole system is its intrinsic uplink-only condition, as it is essentially hardware and setup dependent. Despite there is a functional LoRaWAN communication downlink present, if SCHC Compression was to be implemented on the NS side, some kind of external processing would have to be linked to the Everynet Platform, in order to send the SCHC Packet as the message's payload. This is not impossible, since the platform offers the possibility to issue custom downlink messages upon uplink receipt. However, for an NS to initiate downlink communication, a class B LoRaWAN device is needed, rather than the current class A Mote; this requires studying and understanding class B functionality.

Conclusion

In this work, an experimental test-bed for the IETF's SCHC proposal has been presented. A minimal working SCHC-compliant system has been successfully designed and implemented. The system has been proven to work satisfactorily; its performance has been studied through some preliminary tests which, although limited, have shown positive results. Three possible LPWAN network contexts were evaluated, obtaining packet compression rates of 66.25%, 58.75% and 7.5%.

The implemented test-bed offers a potential development tool for a new dimension within the IoT scenario, where interoperability can be experimented and put to work. This could greatly expand the existing possibilities for interconnection of LPWAN networks, as it allows for architectural flexibility and customized management. By overcoming the interoperability issue and thus breaking the proprietary technology barrier, more complex services become possible, for which new potential applications are yet to be explored.

During the development process, a number of diverse tools from the ICT field were combined into a fully functional system. Despite the system not being fully integrated, the whole system worked correctly, which is a good example of the great potential of telecommunication technologies and resources, and the wide range of applications that can be developed.

Successful transmission of IPv6 packets through an LPWAN network was done and message integrity was verified in order to study relevant contexts for LPWAN networks where SCHC can effectively compress the header and potentially compensate power consumption due to larger data processing while granting network visibility and flexibility. A promising usage scenario for SCHC-enhanced LPWAN networks has been presented and supported.

Bibliography

- [1] S. Farrell, “LPWAN Overview, draft-ietf-lpwan-overview-07,” *IETF, Internet draft (work in progress)*, Oct. 2017. [Online]. Available: <https://datatracker.ietf.org/doc/html/draft-ietf-lpwan-overview-07>
- [2] S. Krishnan, “IPv6 over Low Power Wide-Area Networks WG,” RFC Editor, Fremont, CA, USA, Oct. 2016. [Online]. Available: <https://datatracker.ietf.org/doc/charter-ietf-lpwan/>
- [3] M. Belshe, R. Peon, and M. Thomson (Ed.), “Hypertext Transfer Protocol Version 2 (HTTP/2),” RFC 7540 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–96, May 2015. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc7540.txt>
- [4] G. Montenegro, S. Cespedes, S. Loreto, and R. Simpson., “HTTP/2 Configuration Profile for the Internet of Things,” Mar. 2017. [Online]. Available: <https://github.com/h2ot-wg/h2ot-profile/blob/master/draft-montenegro-httpbis-h2ot-profile-00.txt>
- [5] N. Kushalnagar, G. Montenegro, and C. Schumacher, “IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals,” RFC 4919 (Informational), RFC Editor, Fremont, CA, USA, pp. 1–12, Aug. 2007. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc4919.txt>
- [6] P. Weber, D. Jackle, D. Rahusen, and A. Sikora, “IPv6 over LoRaWAN™,” in *2016 3rd International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*. IEEE, Sept. 2016. [Online]. Available: <https://doi.org/10.1109/idaacs-sws.2016.7805790>
- [7] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” RFC 4944 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–30, Sept. 2007, updated by RFCs 6282, 6775, 8025, 8066. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc4944.txt>
- [8] *Overview of the Internet of things*, ITU-T Recommendation Y.2060, June 2012.
- [9] (2017, June) Top 7 technologies for IoT connectivity 2017. Flespi. [Online]. Available: <https://flespi.com/blog/top-7-technologies-for-iot-connectivity-2017>
- [10] “LoRaWAN™ What is it? A technical overview of LoRa® and LoRaWAN™,”

- LoRa[®] Alliance Technical Marketing Workgroup, Nov. 2015. [Online]. Available: https://docs.wixstatic.com/ugd/eccc1a_ed71ea1cd969417493c74e4a13c55685.pdf
- [11] “LoRaWAN[™] 101 A Technical Introduction,” LoRa[®] Alliance. [Online]. Available: https://docs.wixstatic.com/ugd/eccc1a_20fe760334f84a9788c5b11820281bd0.pdf
- [12] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, *LoRaWAN[™] Specification V1.0*, LoRa[®] Alliance LoRa Specification, Jan. 2015.
- [13] What is LoRa? Semtech. [Online]. Available: <https://www.lora-alliance.org/technology>
- [14] LoRa Alliance[™] Technology. LoRa Alliance[™]. [Online]. Available: <https://www.semtech.com/technology/lora/what-is-lora>
- [15] S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” RFC 8200 (Internet Standard), RFC Editor, Fremont, CA, USA, pp. 1–42, July 2017. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc8200.txt>
- [16] IPv6 Header Format. Redes locales y globales. [Online]. Available: <https://sites.google.com/site/redeslocalesyglobales/6-arquitecturas-de-redes/6-arquitectura-tcp-ip/7-nivel-de-red/8-direccionamiento-ipv6/2-formato-de-la-cabecera-ipv6>
- [17] *Standard for Information technology— Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE Std. 802.15.4[™]-2006, Sept. 2015, revision of IEEE Std 802.15.4-2003.
- [18] J. Hui (Ed.) and P. Thubert, “Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks,” RFC 6282 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–24, Sep. 2011, updated by RFC 8066. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc6282.txt>
- [19] I. Ishaq, D. Carels, G. Teklemariam, J. Hoebeke, F. Abeele, E. Poorter, I. Moerman, and P. Demeester, “IETF standardization in the field of the internet of things (IoT): A survey,” *Journal of Sensor and Actuator Networks*, vol. 2, no. 2, pp. 235–287, apr 2013. [Online]. Available: <https://doi.org/10.3390/jsan2020235>
- [20] A. Minaburo, L. Toutain, C. Gomez, and D. Barthel, “LPWAN Static Context Header Compression (SCHC) and fragmentation for IPv6 and UDP,” Internet Engineering Task Force, Internet-Draft draft-ietf-lpwan-ipv6-static-context-hc-16, Jun. 2018, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/html/draft-ietf-lpwan-ipv6-static-context-hc-16>
- [21] *Aprueba Plan General de Uso del Espectro Radioeléctrico*, Subsecretaría de Telecomunicaciones del Ministerio de Transporte y Telecomunicaciones de Chile Decreto No. 127, Mar 2006, accessed: Nov 2018. [Online]. Available: http://oraias.subtel.cl/sgr_reclamos/bdc_subtel.PKG_SGD_BUSCADOR_NORMAS.prc_despliegue

- [22] “LoRa® Mote User’s Guide,” Microchip Technology Inc., Datasheet DS40001808B, 2015-2016. [Online]. Available: <http://ww1.microchip.com/downloads/en/DeviceDoc/LoRa%20Mote%20Users%20Guide.pdf>
- [23] “RN2903 LoRa™ Technology Module Command Reference User’s Guide,” Microchip Technology Inc., Datasheet DS40001811A, 2015. [Online]. Available: <http://ww1.microchip.com/downloads/en/DeviceDoc/40001811A.pdf>
- [24] “Everynet LoRaWAN Gateway v.2 User Manual,” Everynet, 2017. [Online]. Available: <http://everynet.com/wp-content/uploads/2017/06/Everynet-GW-User-Manual-ENG-V1.3.pdf>
- [25] S. Griffiths. bitstring module. Python Software Foundation. [Online]. Available: <https://pypi.org/project/bitstring/>
- [26] A. Kirby. LoRa radio packet decoder. GitHub, Inc. (US). [Online]. Available: <https://github.com/anthonykirby/lora-packet>
- [27] R. Sinha, C. Papadopoulos, and J. Heidemann, “Internet packet size distributions: Some observations,” USC/Information Sciences Institute, Tech. Rep. ISI-TR-2007-643, May 2007, originally released October 2005 as web page <http://netweb.usc.edu/%7ersinha/pkt-sizes/>. [Online]. Available: <http://www.isi.edu/%7ejohnh/PAPERS/Sinha07a.html>

Appendix: Source code

The code used in this project was uploaded to a *GitHub* repository. The latest versions are located at:

<https://github.com/nicomatu/schc-lorawan>