Contributions to IPv6 support over Low Rate Low Power Wireless Networks

Ph.D. Thesis

Sergio Aguilar Romero

Directors:
Carles Gomez Montenegro
Rafael Vidal Ferré

Department of Network Engineering
Universitat Politècnica de Catalunya

January, 2023

ACKNOWLEDGMENT

Acknowledgment
\ *ak-nol-ij-muhnt* \
noun
1 the fact of accepting that something is true or right
2 recognition of the importance or quality of something
3 a statement printed at the beginning of a book expressing the author's or publisher's gratitude to others
4 [**ACK**] a signal sent by a receiver to indicate that a message or packet has been received and processed successfully.

I would like to thank my wife, my parents, my family and my friends.
Thanks for always being there and helping make this possible.
Thanks to my mentors Carles and Rafael for your guidance, patience, and advice.
Thanks to all the reviewers for your feedback.
Thanks to everyone with whom I traveled.
It was an honor to walk with you.
Thanks to the light for enlightening our way.
Love,
Sergio

CONTENTS

1 Introduction **19**
1.1 Motivation . . . 19
1.2 Results and contributions . . . 20
1.3 Thesis organization . . . 20

2 Technical Background **23**
2.1 Introduction . . . 23
2.2 Low rate low power wireless network technologies . . . 23
2.2.1 LPWAN technologies . . . 23
2.2.1.1 Long Range Wide Area Network (LoRaWAN) . . . 24
Physical Layer . . . 25
Physical Layer message format . . . 26
MAC Layer message format . . . 27
2.2.1.2 Sigfox . . . 28
Physical Layer . . . 28
Radio configuration and features . . . 29
Uplink procedure (U-procedure) . . . 30
Bidirectional procedure (B-procedure) . . . 31
2.2.1.3 NB-IoT . . . 32
Physical Layer . . . 33
Physical Layer message format . . . 34
2.2.1.4 Other LPWAN technologies . . . 35
2.2.2 Wake-up radio systems as low rate data channel . . . 36
2.2.2.1 Wake-up Wi-Fi and secondary radio . . . 36
2.3 IPv6 over low rate low power wireless network technologies . . . 38
2.3.1 Motivation . . . 38
2.3.2 Static Context Header Compression (SCHC) . . . 40
2.3.2.1 SCHC adaptation layer overview . . . 40
2.3.3 SCHC header compression . . . 40
2.3.4 SCHC fragmentation . . . 41
2.3.4.1 SCHC Packet format . . . 41
2.3.4.2 SCHC F/R messages and headers . . . 42
2.3.4.3 Tiles . . . 43
2.3.4.4 Windows . . . 43

2.3.4.5 Bitmap . . . 43
2.3.5 SCHC Fragmentation and Reassembly modes . . . 44
2.3.5.1 No-ACK mode . . . 44
2.3.5.2 ACK-Always mode . . . 44
2.3.5.3 ACK-on-Error mode . . . 44
2.3.6 Counters and Timers . . . 46
2.3.7 SCHC Profiles . . . 46

3 Performance Analysis of SCHC F/R modes 49
3.1 Introduction . . . 49
3.2 Related work . . . 50
3.3 Performance metrics . . . 51
3.3.1 Total channel occupancy . . . 51
3.3.2 Goodput . . . 51
3.3.3 Total delay . . . 51
3.4 Simulations . . . 52
3.4.1 Total channel occupancy . . . 52
3.4.2 Goodput . . . 53
3.4.3 Total delay . . . 54

4 Performance Analysis and Optimal Tuning of SCHC ACK-on-Error Mode 56
4.1 Introduction . . . 56
4.2 Related work . . . 57
4.3 Mathematical model and analysis . . . 58
4.3.1 ACK Message Overhead . . . 58
4.3.2 ACK Bit Overhead . . . 60
4.3.3 ACK Bit Overhead with L2 headers . . . 60
4.3.4 Percentage of used bits per fragment . . . 60
4.3.5 Maximum window and bitmap sizes . . . 61
4.4 Performance evaluation . . . 61
4.4.1 ACK Message Overhead . . . 61
4.4.2 ACK Bit Overhead . . . 61
4.4.2.1 Optimal tile size . . . 61
4.4.2.2 Optimal window size . . . 63
4.4.3 ACK Bit Overhead with L2 headers . . . 64
4.4.4 Percentage of used bits per fragment . . . 65
4.5 Technology-oriented evaluation . . . 67
4.5.1 LoRaWAN and Sigfox relevant parameters . . . 67
4.5.2 Determining SCHC ACK-on-Error mode parameter settings . . . 68
4.5.2.1 Fragment size (F) . . . 68
4.5.2.2 Optimal tile size . . . 68
4.5.2.3 Optimal window size . . . 69
4.5.2.4 FCN field size (N) . . . 70
4.5.2.5 Window field length (M) . . . 70
4.5.2.6 Maximum window and bitmap sizes . . . 71
4.5.3 Optimal window size results: Discussion . . . 71
4.5.4 Impact of optimal ACK-on-Error parameter settings on downlink traffic . . . 72

5 Alternative Receiver-Feedback Techniques for SCHC 75
5.1 Introduction . . . 75
5.2 Related work . . . 76
5.3 Reliable fragmentation over LPWAN . . . 77
5.4 Receiver-Feedback Techniques for LPWAN Fragmentation . . . 77
5.4.1 Uncompressed Bitmap (UB) . . . 77
5.4.2 Compressed Bitmap (CB) . . . 78
5.4.3 List of Lost Fragments (LLF) . . . 78
5.4.4 List of Deltas (LoD) . . . 79
5.5 Simulation environment . . . 81
5.5.1 Main features . . . 81
5.5.2 Settings . . . 82
5.5.3 Error patterns and rates . . . 83
5.6 LoRaWAN operation and settings . . . 84
5.6.1 LoRaWAN physical layer . . . 84
5.6.2 LoRaWAN L2 layer . . . 85
5.6.2.1 LoRaWAN Device Class . . . 86
5.7 Evaluation . . . 86
5.7.1 ACK payload size analysis . . . 86
5.7.1.1 Uniform losses . . . 86
5.7.1.2 Burst losses . . . 89
5.7.2 Average number of L2 frames and ToA gain . . . 90
5.7.2.1 Uniform losses . . . 90
5.7.2.2 Burst losses . . . 92

6 SCHC Packet Fragmentation: Implementation and Performance Evaluation 95
6.1 Introduction . . . 95
6.2 Related work . . . 96
6.3 SCHC over Sigfox Profile . . . 96
6.3.1 SCHC over Sigfox overview . . . 97
6.3.2 Architecture . . . 97
6.3.3 ACK-on-Error mode over Sigfox . . . 97
6.3.3.1 Single-byte-header ACK-on-Error . . . 97
6.3.3.2 Two-byte-header ACK-on-Error . . . 98
6.3.3.3 Operation specifics . . . 98
6.3.3.4 Single- vs two-byte-header trade-off . . . 98
6.4 Theoretical analysis . . . 99
6.4.1 Sigfox message transmission U/B procedures delay . . . 99
6.4.1.1 U-procedure delay . . . 99
6.4.1.2 B-procedure delay . . . 99
6.4.2 SCHC Packet transfer time: Analysis . . . 100
6.4.3 SCHC Packet transfer time: Results . . . 101
6.4.4 Number of messages (UL and DL) . . . 102
6.5 Experimental evaluation . . . 103
6.5.1 Experimental environment . . . 103
6.5.2 Experiment details . . . 104
6.5.3 SCHC Packet transfer time: Analysis validation . . . 105
6.5.4 SCHC Packet transfer time: Experimental results . . . 105
6.5.5 Average number of messages (UL and DL) . . . 106
6.5.6 Results applicability and implementation considerations . . . 107

7 Energy Consumption Model of SCHC Packet Fragmentation 109
7.1 Introduction . . . 109
7.2 Related work . . . 109
7.2.1 Sigfox energy performance . . . 110
7.2.2 SCHC F/R . . . 110
7.3 Modeling SCHC F/R over Sigfox current consumption . . . 111
7.3.1 Experimental setup . . . 112
7.3.2 SCHC Packet transfer states . . . 113
7.3.3 Current consumption profile . . . 114
7.3.3.1 Sleep and Wake-up states current consumption profile . . . 114
7.3.3.2 SCHC Fragmentation states current consumption profile . . . 114
7.3.3.3 U-procedure current consumption profile . . . 115
7.3.3.4 B-procedure current consumption profile . . . 117
7.3.4 SCHC Packet transfer current and energy consumption model . . . 118
7.3.4.1 Number of U-procedure and B-procedure . . . 118
7.3.4.2 Single SCHC Packet transfer model . . . 119
7.3.5 Periodic SCHC Packet transfer energy performance metrics . . . 120
7.4 Evaluation . . . 121
7.4.1 SCHC Packet current and energy consumption . . . 121
7.4.2 Periodic SCHC Packet transfer energy performance . . . 122

8 IPv6 over Cross-Technology Communications with Wake-up Radio using SCHC 125
8.1 Introduction . . . 125
8.2 Related work . . . 127
8.2.1 Cross-Technology Communication (CTC) . . . 127
8.2.2 SCHC . . . 128
8.3 Cross Technology Communication with WuR . . . 128
8.3.1 WuR-CTC Physical Layer overview . . . 128
8.3.2 WuR-CTC Link Layer overview . . . 129
8.3.2.1 WuR-CTC frame types . . . 130
8.3.3 WuR-CTC Message Exchange Overview . . . 130
8.4 SCHC over WuR-CTC Profile design . . . 131
8.4.1 SCHC over WuR-CTC header compression . . . 131
8.4.2 SCHC over WuR-CTC Fragmentation and Reassembly . . . 131
8.4.3 SCHC message adaptation over WuR-CTC . . . 132
8.4.4 SCHC over WuR-CTC message exchange . . . 132
8.5 Evaluation of SCHC over WuR-CTC Profile . . . 133
8.5.1 SCHC over WuR-CTC testbed implementation . . . 133
8.5.2 Experimental methodology . . . 135
8.5.3 Performance metrics . . . 135
8.5.4 Evaluation results . . . 136

9 Compound ACK Message and LPWAN Convergence with SCHC 140
9.1 Compound ACK message: Reducing downlink traffic . . . 140
9.1.1 Introduction . . . 140
9.1.2 Description . . . 141
9.1.3 ACK reduction examples . . . 142
9.2 The SCHC Convergence Profile . . . 144
9.2.1 Introduction . . . 144
9.2.2 Motivation . . . 144
9.2.3 Use cases . . . 144

9.2.4 Description 145
9.2.4.1 Protocol stack 145
9.2.4.2 Architecture 146
9.2.5 Single SCHC ID 146
9.2.6 Uplink fragmentation 147
9.2.7 SCHC Convergence Profile Uplink ACK-on-Error mode 148
9.2.8 SCHC Convergence Profile message formats 148
9.2.8.1 Regular SCHC fragment 148
9.2.8.2 All-1 SCHC fragment 149
9.2.8.3 SCHC ACK message 149
9.2.8.4 SCHC Receiver-Abort message 149
9.2.8.5 SCHC Sender-Abort messages 150
9.2.8.6 SCHC ACK Request message 150

10 Conclusions and Future Work 151
10.1 Conclusions 151
10.2 Future work 153
10.2.1 A new era of interoperability 154
10.2.2 SCHC: LPWAN and beyond 155

11 Contributions and Recognitions 157

Bibliography 161

LIST OF FIGURES

1.1 Thesis structure. 21

2.1 LoRaWAN architecture [14]. 24
2.2 LoRaWAN protocol architecture [15]. 25
2.3 LoRaWAN physical layer message format [15]. The length of each field is denoted below. 27
2.4 LoRaWAN MAC message format [15]. The length of each field is denoted below. . 27
2.5 Sigfox architecture [1]. 28
2.6 Illustration of a Sigfox U-procedure example. State numbers and their corresponding durations are indicated at the lower part of the figure. In the U-procedure, states 1 and 2 are present three and two times, respectively. 30
2.7 Sigfox uplink frame format. 31
2.8 Sigfox B-procedure example illustration. State numbers and their corresponding durations are indicated at the lower part of the figure. In this example, a downlink frame is sent by the Sigfox Cloud and received by the device. The latter sends a confirmation frame. In a B-procedure, states 1 and 2 are present three and two times, respectively. T_{Rx} can take values between T_{RxMIN}, when the downlink frame is received at the start of the reception window, and T_{RxMAX}, when no downlink frame is received. 32
2.9 Sigfox downlink frame format. 33
2.10 NB-IoT features, in comparison with other technologies [23]. 33
2.11 NB-IoT architecture [1]. 33
2.12 NB-IoT operation modes [25]. 34
2.13 NB-IoT downlink and uplink frame structure [28] 35
2.14 Diagram illustrating the operation of WuR system. [5]. 37
2.15 WuR system using the secondary radio as a low rate data channel. Each device main radio communicates with the other device WuR 38
2.16 Example of a low rate data channel using the secondary radio for data exchange and the WuS to begin communication. An ACK is sent by the Target Device to notify the Initiator Device that the WuRx is ready for data reception. 38
2.17 Adaptation layer functionality example [37]. 39
2.18 MTU and capacity comparison between 6Lo and LPWAN technologies [37]. 39
2.19 Protocol stack illustrating the location of the SCHC sublayers between the IPv6 layer and the underlying LPWAN technology [6]. 40

2.20 SCHC compression and fragmentation process [6]. 41
2.21 The IPv6 header is compressed using SCHC. The output of SCHC compression is called a SCHC Packet. 41
2.22 Example of a SCHC Packet fragmented into 20 tiles, with 5 tiles per window. The tile index in the window, called the Fragment Compressed Number (FCN), is indicated for each tile. 42
2.23 Example of a SCHC Packet. 42
2.24 Illustration of a regular SCHC Fragment, an All-1 SCHC Fragment with the RCS field and an All-0 SCHC Fragment. The length in bits of each header field is indicated below each field. 43
2.25 Illustration of a SCHC ACK message. The top SCHC ACK message notifies successful reassembly of a SCHC Packet by carrying a $C = 1$. The bottom one indicates a failed SCHC Packet reassembly ($C = 0$) and carries a bitmap. The length of each header field (in bits) is indicated below each field. 43
2.26 Example of a SCHC ACK, where a window of 7 tiles ($w = 7$) was sent and tile FCN # 6 was not received successfully. 44
2.27 Example of a SCHC Packet transfer that requires fragmentation in the No-ACK mode. The SCHC Packet needs 3 fragments to be carried. 44
2.28 Example of a SCHC Packet transfer that requires fragmentation in the ACK-Always mode. In (a), the SCHC Packet transfer requires 2 windows of 3 tiles, for a total of 6 tiles. In (b) the SCHC Packet transfer requires 1 window of 3 tiles, as the SCHC Packet needs only 3 fragments to be carried. Losses are present in the first (and only) window in (b). 45
2.29 ACK-on-Error mode. 47

3.1 Channel Occupancy vs packet size. Darker colors correspond to $\mathrm{CO}_{\mathrm{tx}}$, and lighter colors correspond to $\mathrm{CO}_{\mathrm{rx}}$. 53
3.2 Goodput vs packet size. 54
3.3 $T_{\mathrm{d\ SCHC}}$ vs packet size. 54

4.1 Example of a SCHC Packet and SCHC Fragments. The SCHC Packet is fragmented in 5 tiles per window ($w = 5$) and the SCHC Fragments can transport 4 tiles per fragment ($f = 4$). For a successful transmission of the packet, W_0 needs $k_2 = 2$ successful SCHC Fragment transmissions, then W_1, W_2, and W_3 need each $k_1 = 1$ more successful SCHC Fragment transmission, in compliance with (4.5): a proportion $\frac{w}{f} - \lfloor \frac{w}{f} \rfloor = 1/4$ of windows need $k_2 = \lceil \frac{w}{f} \rceil$ new SCHC Fragment transmissions, whereas the other windows need only $k_1 = \lfloor \frac{w}{f} \rfloor = 1$ new SCHC Fragment transmission. 59
4.2 E_k for $F = 11$, $t = 9$ and $L_{\mathrm{SH}} = 2$ for different window sizes as a function of P_{s} (a) and k (b). 62
4.3 O_{ACK} vs window size. Impact of the tile size t on the O_{ACK}. $L_{\mathrm{SH}} = 2$. 62
4.4 Impact of the window size w (in tiles) or wt (in bytes) on the ACK overhead O_{ACK}. The optimal window size (w_{opt}) is indicated in (c) and (d) with a mark. $L_{\mathrm{SH}} = 2$. 64
4.5 Impact of the window size w (in tiles) or wt (in bytes) on the ACK overhead O_{ACK}. The optimal window size (w_{opt}) is indicated in (c) and (d) with a mark. $L_{\mathrm{SH}} = 2$. 65
4.6 Optimal window size vs P_{s} for different settings. 66
4.7 Impact of the window size w (in tiles) or wt (in bytes) on the ACK overhead $O_{\mathrm{ACK_{L2}}}$. The optimal window size (w_{opt}) is indicated in (c) and (d) with a mark. $L_{\mathrm{SH}} = 2$ and $L_{\mathrm{L2H}} = 13$. 66
4.8 Percentage of usage P_U vs t. $L_{\mathrm{SH}} = 2$. 67
4.9 Optimal window size (w_{opt}) vs P_{f}. 70

4.10 Impact of w_{nopt} on the AEF (a) and on the BEF (b) for different R_{opt} of devices using w_{opt}. Results for LoRaWAN US915, SF10/DR0, $F = 11$, $L_{\text{SCHC}} = 1280$, and $t_{\text{opt}} = 9$. 73
4.11 Impact of t_{nopt} on the AEF (a) and on the BEF (b) for different R_{opt} of devices using t_{opt}. Results for LoRaWAN US915, $F = 11$, $L_{\text{SCHC}} = 1280$, $N_{\text{SCHC}} = 100$, and w_{opt}. 73

5.1 Example of the transmission of a 10-fragment packet with errors. 77
5.2 Example of ACK payload for UB, and ACK format, for the transmission of a 10-fragment packet where two fragments (with FNs 1 and 6) are lost. 78
5.3 Example of a bitmap before compression and the corresponding compressed bitmap. The rightmost sequence of consecutive bitmap bits set to 1 is removed to obtain the compressed bitmap. The example assumes a byte-oriented underlying L2 LPWAN technology. In this example, padding is not needed. 79
5.4 With LLF, the FNs of lost fragments are converted to binary and appended to the SCHC ACK Header. Finally, padding is added as needed. 79
5.5 Example of different SDNV-encoded numbers. Data bits are represented in bold font. 80
5.6 Examples of ACKs, using LoD, and for the fragment losses shown in Fig. 5.1 (*i.e.*, fragments with FN=1 and FN=6), for different SDNV bases. The decimal numbers to be encoded are 1 (first FN) and 5 (first delta). All considered SDNV base sizes require one base to encode the FN of the first lost fragment. To encode the delta, SDNV-4 and SDNV-5 only require 1 base, whereas SDNV-2 and SDNV-3 need 3 and 2 bases, respectively. 81
5.7 Probability Density Function (PDF) of the theoretical vs simulated burst length ($\lambda = 10$). 84
5.8 ToA_{frame} for different LoRaWAN L2 payload sizes obtained by using (5.1) and the parameters shown in Table 5.3. 85
5.9 Delta Probability Density Function (PDF), for $FER = 1\%$ and uniform losses. . . 87
5.10 Delta Probability Density Function (PDF), for $FER = 10\%$ and uniform losses. . 88
5.11 Delta Probability Density Function (PDF), for $FER = 20\%$ and uniform losses. . 89
5.12 Delta Probability Density Function (PDF), for $BOP = 1\%$. 90
5.13 Average number of L2 frames (a) and ToA_{Gain} (b,c,d) vs packet size for $FER = 1\%$ and uniform losses. In (b) results for LLF, and all LoD variants considered are overlapped. 91
5.14 Average number of L2 frames (a) and ToA Gain (b,c,d) vs packet size for $FER = 10\%$ and uniform losses. 92
5.15 Average number of L2 frames (a) and ToA_{Gain} (b,c,d) vs packet size for $FER = 20\%$ and uniform losses. 93
5.16 Average number of L2 frames (a) and ToA Gain (b,c,d) vs packet size for $BOP = 1\%$ and $\lambda = 10$. 94

6.1 SCHC over Sigfox architecture. 97
6.2 Experimental setup for SCHC Packet transfer time measurements. 101
6.3 Theoretical SCHC Packet transfer time for (a) RC1 and (b) RC4. In (a), the *duty-cycle* is of 1% and the total time off (T_{OFF}) required to comply with *duty-cycle* regulations is depicted in shaded blue. 102
6.4 SCHC F/R mechanism diagram using `Fragmenter` and `Reassembler` objects. . . . 104
6.5 Theoretical and experimental SCHC Packet transfer time with 1% *duty-cycle*. Sigfox zone RC1. 106

6.6 Experimental SCHC Packet transfer time: results obtained in RC1 (1% *duty-cycle*). The solid colored part of the bars represents T_{SCHC}, and the dashed part illustrates the time off (T_{OFF}) required to comply with *duty-cycle* regulations. 106
6.7 Experimental SCHC Packet transfer time: results obtained in RC4. 107
6.8 Number of UL messages: experimental results, Sigfox zones RC1 and RC4. The solid colored part, the darker colored part, and the lighter colored part represent the Regular SCHC Fragments, All-0 SCHC Fragments, and All-1 SCHC Fragments, respectively. 107
6.9 Number of DL messages, experimental results: Sigfox zones RC1 and RC4. 108

7.1 Experimental setup with the Sigfox device and the power analyzer used. 112
7.2 Fragmented SCHC Packet transfer state diagram. Sleep and Wake-up states, SCHC Fragmentation-related states, and Sigfox transmission states are depicted in blue, green, and purple, respectively. X and Y variables correspond to the number of cycles (N_C) and to the number of SCHC Fragments per cycle (N_{pC}), respectively. 113
7.3 Average energy consumption of Wake-up and Sleep states for different T_{Sleep} values, and for light and deep sleep mode. 115
7.4 Fragmenter state time as a function of the SCHC Packet size. 116
7.5 Current consumption profile of a LoPy4 device performing a U-procedure. In this measurement, the Sigfox UL frame payload size is 12 bytes, equivalent to a Regular (not All-0) SCHC Fragment carrying one tile. 116
7.6 Current consumption profile of a LoPy4 in a B-procedure, with an UL frame payload of 12 bytes, equivalent to an All-0 SCHC Fragment or an All-1 SCHC Fragment carrying one tile. A DL frame is received by the device, which subsequently sends a confirmation control frame. 117
7.7 Current consumption of two SCHC Packet transfer examples: (a) a 22-byte SCHC Packet is sent completely and a SCHC ACK is received; (b) six SCHC Fragments are sent back-to-back before the device returns to the Sleep state in the first transfer cycle of a 77-byte SCHC Packet. 119
7.8 Average current consumption of a SCHC Packet transfer over Sigfox. 121
7.9 Energy consumed by a device to perform a SCHC Packet transfer over Sigfox. . . 122
7.10 Average current consumption of a device performing periodic SCHC Packet transfers over Sigfox, for different T_p, N_{pC}, and SCHC Packet size values. 123
7.11 Energy consumption of a SCHC Packet transfer over a period T_p, for different T_p, N_{pC}, and SCHC Packet size values. 123
7.12 Lifetime of a battery-operated device performing periodic SCHC Packet transfers over Sigfox, for different T_p, N_{pC}, and SCHC Packet size values. 124

8.1 Gateway-based communication between Device A (IEEE 802.15.4 radio interface) and Device B (IEEE 802.11 radio interface). 126
8.2 WuR-CTC-based communication between Device A (an IEEE 802.15.4 device) and Device B (an IEEE 802.11 device). 126
8.3 WuR-CTC communication between Device A (with an IEEE 802.15.4 main radio) and Device B (with an IEEE 802.11 main radio). The main radio of a device works as the WuTx. 129
8.4 WuR-CTC PPDU frame format. 129
8.5 WuR-CTC MPDU frame format. 130
8.6 Communication between two nodes using WuR-CTC. 131
8.7 SCHC over WuR-CTC ACK message format. 132
8.8 SCHC over WuR-CTC message exchange between Node 0x001 and Node 0x002, with one retransmission. 133

8.9 SCHC over WuR-CTC implementation architecture and main WuR-CTC blocks. The WuRx hardware is external to the device. The application generates a payload, *e.g.*, a smart thermostat measurement, which is carried by an IPv6 packet. The latter is compressed/decompressed and fragmented/reassembled by SCHC. 134
8.10 SCHC over WuR-CTC testbed implementation. The Front-End Radio and the Baseband modules are framed in dashed blue and red lines, respectively. The IEEE 802.11 Device is framed in dashed yellow lines. The IEEE 802.15.4 Device is framed in dashed green lines. 134
8.11 Compression ratio by using SCHC header compression as in our implementation. . 136
8.12 Error rate as a function of IPv6 packet size, for 30-ms and 1000-ms RT. 137
8.13 (a) Number of SCHC over WuR-CTC messages and (b) overhead (in bytes) of a SCHC over WuR-CTC IPv6 packet transfer, with a RT of 30 ms (with lines) and 1000 ms (without lines). 137
8.14 IPv6 packet transfer time, for different IPv6 Packet sizes, with (a) 30-ms RT and (b) 1000-ms RT. 138
8.15 Throughput results for different IPv6 packet sizes, with (a) 30-ms RT and (b) 1000-ms RT. 139

9.1 Compound ACK message format, modifying the failure SCHC ACK format, when more than M padding bits are required to complete the L2 word. 141
9.2 Compound ACK message format, modifying the failure SCHC ACK format, when less than M padding bits are required to complete the L2 word. Losses are found in windows $W = w_1, ..., w_i$; where $w_1 \leq w_2 \leq ... \leq w_i$ 142
9.3 SCHC Packet transfer example with SCHC ACKs. The SCHC Packet size is 14 tiles and window size is 7 tiles. Red frames highlight the ACK messages. 143
9.4 SCHC Packet transfer example with Compound ACK messages. The SCHC Packet size is 14 tiles and window size is 7 tiles. Red frames highlight the ACK messages. 143
9.5 Comparison of SCHC protocol stack (a) with the proposed SCHC Convergence Profile protocol stack (b) 145
9.6 Comparison of current SCHC architecture (a) with SCHC Convergence Profile architecture (b). 146
9.7 Single SCHC device ID translation table diagram 147
9.8 Regular SCHC fragment 149
9.9 All-1 SCHC fragment (with tile) 149
9.10 All-1 SCHC fragment (no tile) 149
9.11 Successful SCHC ACK 149
9.12 Failure SCHC Compound ACK message. Losses are found in windows $W = w_1, ..., w_i$; where $w_1 \leq w_2 \leq ... \leq w_i$ 150
9.13 SCHC Receiver-Abort message 150
9.14 SCHC Sender-Abort message 150
9.15 SCHC ACK Request message 150

LIST OF TABLES

2.1 DRs and configurations for EU863-870 band channels [3]. 26
2.2 DRs and related configuration for US902-928 band channels [3]. 26
2.3 Sigfox uplink and downlink number of messages, payload and throughput [19–21]. . 29
2.4 Radio Configuration (RC) for different Sigfox geographical zones. 29
2.5 Uplink frame transmission time as a function of the payload size 31
2.6 NB-IoT characteristics [26]. 34
2.7 Technical specifications of various LPWAN technologies [13]. 36
2.8 Technical specifications of various LPWAN technologies [13]. 36

4.1 ACK-on-Error configuration parameters for w_{opt} in Table 4.1 for LoRaWAN EU868 68
4.2 ACK-on-Error configuration parameters for w_{opt} in Table 4.2 for LoRaWAN US915 69
4.3 ACK-on-Error configuration parameters for w_{opt} for Sigfox 69
4.4 Maximum bitmap and window sizes for LoRaWAN and Sigfox 71

5.1 LoRaWAN MTU Values 82
5.2 LoRaWAN Error Distributions in Literature 83
5.3 LoRa and LoRaWAN configuration parameters 85

6.1 U-Procedure Transmission States and Their Corresponding Durations 101
6.2 B-Procedure Fragment Transmission States and Their Corresponding Durations . . 102
6.3 Number of UL and DL Messages in a No Loss Scenario 103
6.4 SCHC Packet Sizes Evaluated 105

7.1 List of references that evaluate Sigfox energy performance. 111
7.2 List of references related to SCHC F/R performance evaluation. 112
7.3 Wake-up and Sleep states characterization for the light sleep mode. 114
7.4 Wake-up and Sleep states characterization for the deep sleep mode. 115
7.5 SCHC Fragmentation states. 115
7.6 U-procedure substates and their corresponding duration and current consumption values. 116
7.7 B-procedure substates and their corresponding duration and current consumption values. 118
7.8 SCHC Packet sizes used in the energy performance evaluation. 122

8.1 WUR-CTC frame types 130
8.2 IPv6 Packet sizes evaluated 135

ACRONYMS

3GPP	3rd Generation Partnership Project.
6Lo	IPv6 over Networks of Resource-constrained Nodes.
6LoWPAN	IPv6 over Low Power Wireless Personal Area Networks.
ACK	Acknowledgment.
AP	Access Point.
API	Application Program Interfaces.
BER	Bit Error Rate.
BF	Bidirectional Flag.
BLE	Bluetooth Low Energy.
BPSK	Binary Phase Shift Keying.
BS	Base Station.
BW	Channel Bandwidth.
C/D	Compression/Decompression.
CB	Compressed Bitmap.
CBOR	Concise Binary Object Representation.
CO	Channel Occupancy.
CoAP	Constrained Application Protocol.

CR	Coding Rate.
CRC	Cyclic Redundancy Check.
CSS	Chirp Spread Spectrum.
CTC	Cross-Technology Communication.
DBPSK	Differential Binary Phase-Shift Keying.
DE	Low Data Rate Optimization Enable.
DECT	Digital Enhanced Cordless Telecommunications.
DMRS	Demodulation Reference Signal.
DR	Data Rate.
EC-GSM	Extended Coverage GSM.
ECC	Error Correction Code.
eMTC	enhanced MTC.
ETSI	European Telecommunications Standard Institute.
F/R	Fragmentation/Reassembly.
FEC	Forward Error Correction.
FER	Frame Error Rates.
FHDR	Frame Header.
FLR	Fragment Loss Rate.
FRM	Frame.
FT	Field Type.
GFSK	Gaussian Frequency Shift Keying.
GSM	Global System for Mobile Communications.
H	Header Enable.
HSS	Home Subscriber Server.
IEEE	Institute of Electrical and Electronics Engineers.

IETF	Internet Engineering Task Force.
IoT	Internet of Things.
IPv6	Internet Protocol version 6.
ISM	Industrial Scientific Medical.
L2	Layer 2.
LI	Length Indicator.
LLF	List of Lost Fragments.
LoD	List of Deltas.
LoRa	Long Range.
LoRaWAN	Long Range Wide Area Network.
LPWAN	Low Power Wide Area Networks.
LRLPWN	Low Rate Low Power Wireless Network.
LTE	Long Term Evolution.
LTN	Low Throughput Networks.
LwM2M	Lightweight Machine to Machine.
MAC	Medium Access Control.
MAUTH	Message Authentication Code.
MBS	Maximum Bitmap Size.
MCU	Microcontroller Unit.
MHDR	MAC Header.
MIB	Master Information Block.
MIC	Message Ingrity Code or Check.
MME	Mobility Management Entity.
MTC	Machine Type Communication.
MTU	Maximum Transmission Unit.

MWS	Maximum Window Size.
NB-IoT	Narrow band IoT.
NFC	Near Field Communication.
NPBCH	Narrowband Physical Broadcast Channel.
NPDCCH	Narrowband Physical Downlink Control Channel.
NPDSCH	Narrowband Physical Downlink Shared Channel.
NPRACH	Narrowband Physical Random Access Channel.
NPSS	Narrowband Primary Synchronization Signals.
NPUSCH	Narrowband Uplink Shared Channel.
NRS	Narrowband Reference Signal.
NS	Network Server.
NSSS	Narrowband Secondary Synchronization Signals.
OFDMA	Orthogonal Frequency Division Multiple Access.
OOK	On-Off-Keying.
P-GW	Packet Data Network Gateway.
PDF	Probability Density Function.
PHDR	Physical Header.
PHDR_CRC	Physical Header Cyclic Redundancy Check.
PL	Number of Payload Bytes.
PPP	Point-to-Point.
PSM	Power Saving Mode.
RC	Radio Configuration.
RCS	Reassembly Check Sequence.
REP	Repeated Flag.
RF	Radio Frequency.

RFT	Receiver-Feedback Technique.
RSSI	Received Signal Strength Indicator.
RT	Retransmission Timer.
RU	Resource Unit.
SC-FDMA	Single Carrier Frequency Division Multiple Access.
SDNV	Self-Delimiting Numeric Values.
S-GW	Serving Gateway.
SCHC	Static Context Header Compressor.
SF	Spreading Factor.
SIB	System Information Block.
STA	Station.
ToA	Time-on-Air.
UB	Uncompressed Bitmap.
UE	User Equipment.
ULE	Ultra Low Energy.
UNB	Ultra Narrow Band.
WG	Working Group.
WuR	Wake-up Radio.
WuR-CTC	Cross-Technology Communication using Wake-up Radio.
WuRx	Wake-up Radio Receiver.
WuS	Wake-up Signal.
WuTx	Wake-up Transmitter.

CHAPTER 1 INTRODUCTION

<<Light is focused on the source.>>

This chapter provides the motivation, contributions and organization of the PhD thesis. Section 1.1 presents the motivation of this thesis investigation. The main thesis contributions are listed in Section 1.2. Finally, Section 1.3 describes the structure of this thesis document.

1.1 Motivation

Low Rate Low Power Wireless Networks (LRLPWNs) refer to network technologies designed for the Internet of Things (IoT). The IoT can be defined as the interconnection of various objects (sensors, actuators, or goal-specific applications) to the global Internet. LRLPWNs can be divided in two categories: Low Power Wide Area Networks (LPWANs) and Low Rate Channel over Wake-up Radio (WuR) systems. Both LPWANs and Low Rate Channel over WuR have a unique set of characteristics, not only low rate and low power, but also a limited data frame payload size. LPWANs also provide long-range communication and low infrastructure cost [1]. LRLPWN technologies have attracted the attention of the industry, academia and standard development organizations, focused on the study of Internet Protocol version 6 (IPv6) [2] over LPWAN networks, and in power consumption and latency reductions of sleepy nodes with WuR systems.

The huge number of heterogeneous devices brought by IoT networks stresses the need for strong interoperability. The global Internet already provides this interoperability feature using a common set of protocols. However, these protocols were not designed for LRLPWNs. This is especially true for IPv6, even though its large address space would be a desirable feature to handle large number of IoT devices. As a result, using IPv6 on LPWANs implies numerous challenges, mainly due to its large header overhead (40 bytes) and that it requires lower layers to support a Maximum Transmission Unit (MTU) of 1280 bytes [2]. However, the MTU sizes supported by LPWAN technologies are typically much smaller. For instance, LoRaWAN [3] and Sigfox [4] support MTU sizes of up to 242 bytes and 12 bytes, respectively. Similar challenges are present in low rate channel over WuR systems [5].

To tackle these two problems, the LPWAN Working Group (WG) of the Internet Engineering Task Force (IETF) has standardized a new framework called Static Context Header Compression and fragmentation (SCHC, pronounced "sheek") [6]. This framework provides IPv6 header Compression/Decompression (C/D) mechanisms, and several Fragmentation/Reassembly (F/R) modes to tackle the IPv6 header size problem and satisfy its MTU size requirements.

The growing interest in LPWAN technologies, and specially in the IPv6 adaptation layers, has pushed the IETF LPWAN WG to further customize SCHC to optimally work over flagship LPWAN technologies. New standards focused on enabling SCHC over Sigfox [7] and NB-IoT [8] are work in progress. SCHC over LoRaWAN [9] is already published as Internet standard. All these documents objective is to optimally adapt SCHC to the special characteristics of each technology and leverage the available resources in terms of MTU size and how the device sends and receives messages.

However, the SCHC framework optimal configuration parameters and performance was unknown. Therefore, research that focus on modeling, analyzing, evaluating, and optimizing SCHC is required. This evaluation is required for the generic SCHC framework [6] and for SCHC over the LPWAN technologies too. This thesis investigation is aligned with such requirements, and aims at advancing the state of the art in the area of IPv6 support over LRLPWNs, through the contributions presented in the following section.

1.2 Results and contributions

The main contributions of this thesis are:

a) Performance evaluations of all SCHC Fragmentation modes over LoRaWAN.

b) Optimally tuned SCHC ACK-on-Error mode configuration values for SCHC Fragmentation over LoRaWAN and Sigfox.

c) Improvements in SCHC Acknowledgment (ACK) size with new Receiver-Feedback Techniques (RFTs) and evaluation in all LoRaWAN world regions.

d) Performance evaluation of SCHC over Sigfox in all world regions.

e) Energy and current consumption model for SCHC F/R over Sigfox.

f) Proposal of the Compound ACK message, which reduces and optimizes the ACK traffic.

g) Proposal of the SCHC Convergence Profile to enable full LPWAN interoperability.

h) Design, implementation, and evaluation of IPv6 support over Cross-Technology Communication with Wake-up Radios, extending the usage of the SCHC framework between IEEE 802.11 and IEEE 802.15.4 devices.

1.3 Thesis organization

This thesis is composed of 11 chapters. Chapter 1 (current chapter) and Chapter 2 are an introduction and technical background, respectively, of the investigation performed in this thesis, which is covered in Chapters 3 to 9. Chapter 10 presents the conclusions of this thesis and future work that has sprouted from this thesis investigation. Chapter 11 shows a list of published contributions, including journal articles, conference papers, Internet Drafts, open source projects, simulators built, Projects, international research visit, participation in the IETF, recognitions and grants received during this thesis investigation.

The structure of this thesis is presented in Fig. 1.1. Below each section, a roman number cites the publication involved. On top, the Internet Drafts that have been impacted or have been produced as a result of this thesis investigation are shown with a box that covers the thesis chapters related to such documents.

Chapters 3 to 9 are organized as follows. In Chapter 3, we provide an overview of the SCHC F/R modes and evaluate their trade-offs over LoRaWAN by simulations. The analyzed parameters are the total channel occupancy, goodput and total delay at the SCHC layer. The results of these evaluations have led to the publication of a conference paper [VI].

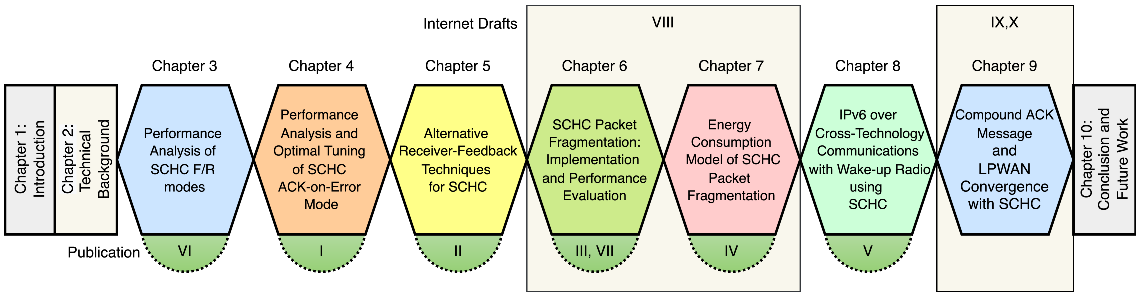

Internet Drafts
VIII
IX,X
Chapter 1: Introduction
Chapter 2: Technical Background
Chapter 3
Performance Analysis of SCHC F/R modes
Chapter 4
Performance Analysis and Optimal Tuning of SCHC ACK-on-Error Mode
Chapter 5
Alternative Receiver-Feedback Techniques for SCHC
Chapter 6
SCHC Packet Fragmentation: Implementation and Performance Evaluation
Chapter 7
Energy Consumption Model of SCHC Packet Fragmentation
Chapter 8
IPv6 over Cross-Technology Communications with Wake-up Radio using SCHC
Chapter 9
Compound ACK Message and LPWAN Convergence with SCHC
Chapter 10: Conclusion and Future Work
Publication
VI
I
II
III, VII
IV
V

Figure 1.1: Thesis structure.

In Chapter 4, we develop a mathematical model to compute the most critical performance parameters for the SCHC ACK-on-Error mode, namely, the ACK traffic incurred by a fragment receiver for the successful delivery of a fragmented packet. The model is used to evaluate the SCHC ACK-on-Error mode performance, as well as to optimally tune its main parameters when used over LoRaWAN and Sigfox, for different packet sizes. Additionally, we illustrate how our derived optimal settings allow to reduce the ACK traffic in a number of scenarios. The results of this investigation provided useful insights in the development of the IETF SCHC framework as well as in the design of the technology specific SCHC Profiles, and have been published in a journal article [I]. Part of the investigation of this chapter was performed as part of a research visit at the IMT-Atlantique in Rennes, France.

SCHC defines an RFT, called Compressed Bitmap (CB), by which a receiver reports to the sender whether the fragments carrying a packet have been received or not. Such information is carried as ACK payload. In Chapter 5, we compare the performance of CB with that of several alternative RFTs, namely List of Lost Fragments (LLF), List of Deltas (LoD), and Uncompressed Bitmap (UB), where the latter is used as a benchmark. We evaluate the considered RFTs in terms of ACK size, number of Layer 2 (L2) frames needed to carry an ACK, and ACK Time on Air. Furthermore, we provide guidance on which RFT should be used for different packet sizes, error rates and error patterns. Results obtained and guidance from this evaluation have been published in a journal article [II].

In Chapter 6, we provide a performance evaluation of SCHC over Sigfox, a flagship LPWAN technology. We focus on the main SCHC over Sigfox fragmentation mode, called ACK-on-Error, which offers low overhead, reliability, and reassembly functionalities. We provide a theoretical analysis and an experimental evaluation in real environments that correspond to two geographical zones with different Sigfox radio settings: Barcelona (Spain) and Santiago (Chile). The study focuses on modeling and evaluating packet transfer times, and the required number of uplink and downlink messages. This work was performed in collaboration with the Sigfox company and the University of Chile. The results of this evaluation led to the publication of a journal article [III] and collaboration in a conference paper [VII]. Moreover, collaboration with the IETF LPWAN WG led to co-authorship of the SCHC over Sigfox Profile Internet Draft [VIII], and the creation of a new Internet Draft called SCHC Compound ACK message [IX].

In Chapter 7, we present a current and energy consumption model of SCHC Packet transfer over Sigfox. The model, which is based on real hardware measurements, allows to determine the impact of several parameters and fragment transmission strategies on the energy performance of SCHC Packet transfer over Sigfox. The evaluation results have been published in a journal article [IV].

In Chapter 8, we present the design, implementation and evaluation of a solution to provide IPv6 support over Cross-Technology Communication using Wake-up Radio (WuR-CTC), by leveraging the IETF SCHC framework. SCHC over WuR-CTC is performed between IEEE 802.15.4 and

IEEE 802.11, two examples of popular technologies in some crucial IoT domains (*e.g.*, smart home, smart buildings, smart factories and smart cities, among others). This way, IoT devices supporting different wireless technologies are interoperable without the need of a gateway. The designed solution supports typical real-time interactions in smart environments (*e.g.*, smart homes) with a human user in the loop. The evaluation results led to a journal article which is currently under review [V].

In Chapter 9, we present current work at the IETF LPWAN WG that has stemmed from the investigation and evaluation performed in this thesis. First, the Compound ACK message format is described. The Compound ACK message is a new message added to the IETF SCHC framework, which was built from the conclusions of Chapter 6, that favors the optimization of downlink traffic. The Compound ACK was presented and adopted by the IETF LPWAN WG [XI] as a proposed standard. Second, LPWAN convergence using the IETF SCHC framework is presented. LPWAN convergence is defined as the usage of multiple LPWANs in cooperation. This is achieved using the SCHC Convergence Profile, which aims to provide interopability at the SCHC fragment level between LPWAN technologies, by providing a single SCHC F/R sublayer. The idea was presented to the IETF LPWAN WG and submitted as an Individual Submission to the IETF [X].

CHAPTER 2

TECHNICAL BACKGROUND

<<Light does not exist if it is not on the go.>>

In this chapter we present the technical background in LRLPWN technologies relevant to this thesis investigation.

2.1 Introduction

In this chapter we present the most relevant LRLPWN technologies and the adaptation of IPv6 over these networks. LRLPWNs can be classified in two categories: LPWAN and low rate channel over WuR systems. Each technology targets different use cases and has different systems. LPWANs are characterized by supporting long range communications while low rate channel over WuR has a limited range. However, both technologies are optimized for battery lifetime and have similar characteristics: low bit rates, low power and a reduced L2 MTU size. IPv6 has desirable characteristics, such as a large address space, support of security protocols and interoperability. To be compatible with IPv6, the underlying low rate low power technology must support a 1280-byte MTU. Moreover, IPv6 has a header size of 40 bytes, that in some cases can be larger than the L2 MTU of LPWAN technologies. To obtain the benefits of IPv6 in LRLPWNs, the need to perform header compression and fragmentation to transmit large packet sizes becomes critical. The IETF LPWAN WG has proposed a new framework that performs header compression and fragmentation, especially designed to support IPv6 for LPWAN technologies. The framework is called SCHC. In this section we first present the most relevant LPWAN technologies. Then we detail the WuR system and how the secondary radio can be used as a low rate low power communication channel. Finally, a SCHC framework overview is shown, specially focused on the three F/R modes offered by SCHC.

2.2 Low rate low power wireless network technologies

2.2.1 LPWAN technologies

LPWANs refer to network technologies designed for IoT that are characterized by a long-range and low-energy operation [1,10,11]. They complement traditional cellular and short range wireless technologies in addressing diverse requirements of IoT applications. With a long range, in the order of a few to tens of kilometers [12], and battery life of ten years and beyond, LPWAN technologies are promising for an Internet of low power, low cost, and low-throughput things. The long range low power characteristic of LPWAN technologies is achieved with a trade-off between low data rate and higher latency. This makes LPWAN not suitable for all IoT use cases, but for delay tolerant, low rate, low power IoT applications [13]. LPWAN technologies are typically based on star topology deployments, where a potentially high number of IoT devices are directly connected to a radio gateway. In this section we present the most relevant LPWAN technologies. First, the flagship LoRaWAN, Sigfox, and NB-IoT are described in detail. Then, a summary of other LPWAN technologies is presented.

2.2.1.1 Long Range Wide Area Network (LoRaWAN)

Long Range Wide Area Network (LoRaWAN) is a technology designed for long range and low power communications. LoRaWAN defines the communication protocol and system architecture for the network while the Long Range (LoRa) physical layer enables the long-range communication link [14]. Furthermore, the LoRa layer features low power operation (*e.g.*, around 10 years of battery lifetime), low data rate (*i.e.*, from 250 bps to 50 kbps depending on selected configuration) and long communication range (*e.g.*, 2-5 km in urban areas and 15 km in suburban areas) [12].

LoRaWAN networks are organized in a star-of-stars topology [12] (see Fig. 2.1). The topology is composed of three basic elements: end-nodes (end-devices), gateways, and a central Network Server (NS) [15]. The end-nodes can be sensors or actuators (*e.g.*, enabling applications such as tracking, alarms, metering, monitoring). They communicate with the NS using one or more gateways which receive the sent message. On the other hand, the NS only sends downlink messages using a specific gateway. While the gateway uses LoRa physical layer to communicate with the end-nodes, the communication with the NS uses the IP protocol stack as shown in Fig. 2.2. The LoRaWAN layer provide end-to-end encryption and data integrity. The Application Session Key provides confidentiality for the upper layer payloads, and the Network Session Key is used to provide data integrity [15].

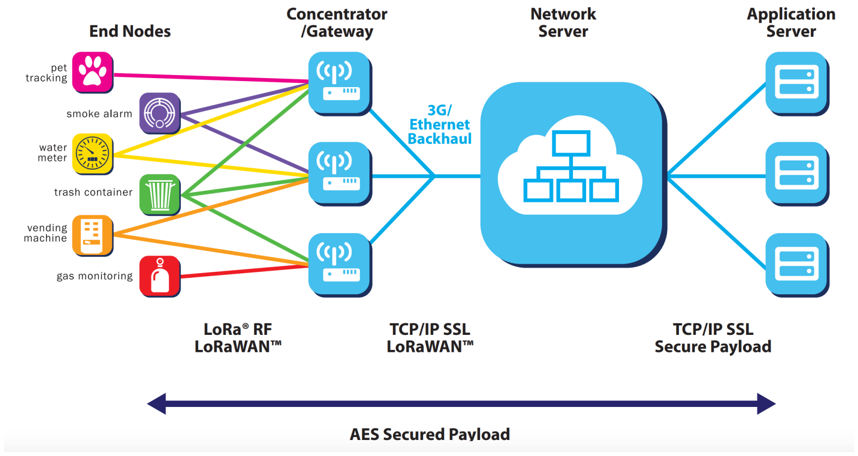


Figure 2.1: LoRaWAN architecture [14].

Three functional classes are defined in the LoRaWAN specifications for the end-devices: Class A, Class B, and Class C. All LoRaWAN devices must implement at least Class A. Class A end-devices are bi-directional whereby after each uplink transmission it opens two short downlink windows to receive messages from NS. Following classes are based on Class A. Class B, differently from Class A, supports scheduled receiving slots, allowing more opportunities to communicate between the gateway and the end-device. In order to synchronize the end-device reception window with the gateway transmission, a time synchronized Beacon is sent from the gateway. Class C end-devices have the receiver window open continuously, only closing it when performing uplink

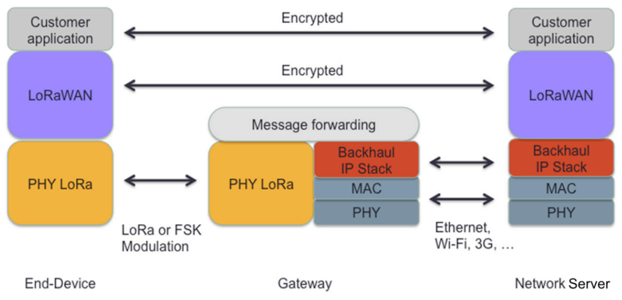


Figure 2.2: LoRaWAN protocol architecture [15].

transmissions. When comparing the different functionality classes, Class C devices consume more power than Class A and B, but offer lower latency in downlink transmissions [3].

Physical Layer In LoRaWAN, the physical transmission between the end-device and the gateway uses LoRa modulation and Gaussian Frequency Shift Keying (GFSK). The LoRa modulation is based on the Chirp Spread Spectrum (CSS) mechanism [15].

The duration of a packet or fragment transmission is defined as Time-on-Air (ToA) and depends on a given Spreading Factor (SF) and channel bandwidth (BW). The SF is defined as the logarithm in base 2 of the number of chips per symbol used for modulation [16]. A LoRa symbol has 2^{SF} chips that cover the entire frequency band. The SF can take values from 7 to 12 [3] giving different Data Rates (DRs), each with certain spectral efficiency and network capacity. As the SF value increases, efficiency and capacity increase as well. LoRa provides recovery against bit errors by using forward error correction with a penalty of a small overhead. This recovery feature is implemented using different Coding Rates (CR), ranging from 4/5 (CR=1) to 4/8 (CR=4). Moreover, an optimization mechanism is used in lower DRs to avoid problems regarding crystal drifts of the reference oscillator. This increases robustness to frequency variation over the time required to receive a LoRa message and is only used for SF11 and SF12.

Depending on the country, different Radio Frequency (RF) bands are used for LoRaWAN deployments. In the EU region, LoRaWAN operates in the EU863-870 Industrial Scientific Medical (ISM) band. Three default channel are defined: 868.10, 868.30, 868.50 MHz [3]. Each channel has a BW of 125 kHz, and must allow DRs from 0.3 kbps to 5 kbps (DR0 to DR5), as shown in Section 2.2.1.1. Every LoRaWAN device must implement these channels if working in the European region. The different DR and the corresponding SF, BW, modulation, and physical bit rate are shown in Section 2.2.1.1.

In the EU region, the 868 MHz ISM band has a *duty-cycle* restriction of 1% per channel. The *duty-cycle* is used to describe the percentage of time a device can actively transmit. T_{off} can be defined as the amount of time the end-device has to sleep after the transmission of data because of the *duty-cycle* regulation, and can be calculated as:

$$T_{\text{off}}[s] = ToA[s] \times \frac{100 - \textit{duty-cycle}}{\textit{duty-cycle}}, \tag{2.1}$$

where the *duty-cycle*, in %, is dependent on packet length and its corresponding transmission duration, *i.e.,* its ToA [16].

Section 2.2.1.1 shows the DR and the corresponding SF, BW, modulation, and physical bit rate. The 915 MHz ISM band is divided into different channel plans: an upstream plan with 64 channels (0 to 63) using LoRa modulation with a BW of 125 kHz, varying from DR0 (902.3 MHz) to DR3 (914.9 MHz) incrementing linearly by 200 kHz steps. Another upstream plan with 8

Table 2.1: DRs and configurations for EU863-870 band channels [3].

DR	Configuration			Physical Bit Rate (bit/s)
	Modulation	Spreading Factor (SF)	Bandwidth	
0	LoRa	SF12	125 kHz	250
1	LoRa	SF11	125 kHz	440
2	LoRa	SF10	125 kHz	980
3	LoRa	SF9	125 kHz	1760
4	LoRa	SF8	125 kHz	3125
5	LoRa	SF7	125 kHz	5470
6	LoRa	SF7	250 kHz	11000
7	FSK	50 kbit/s		50000
8-15	Reserved for Future Use			

channels (64 to 71) using LoRa modulation with a BW of 500 kHz, starting at DR4 (903.0 MHz) and incrementing by 1.6 MHz steps to 914.2 MHz, and a downstream plan with 8 channels (0 to 7) using LoRa modulation with a BW of 500 kHz at DR10 (923.3 MHz) to DR13 (927.5 MHz) incrementing linearly by 600 kHz steps [3].

End-devices that operate in the US902-928 feature the channel data structure of the 72 channels. In the US region, there is no *duty-cycle* restriction.

Table 2.2: DRs and related configuration for US902-928 band channels [3].

DR	Configuration			Physical Bit Rate (bit/s)
	Modulation	Spreading Factor (SF)	Bandwidth	
0	LoRa	SF10	125 kHz	980
1	LoRa	SF9	125 kHz	1760
2	LoRa	SF8	125 kHz	3125
3	LoRa	SF7	125 kHz	5470
4	LoRa	SF8	500 kHz	12500
5-7	Reserved for Future Use			
8	LoRa	SF12	500 kHz	980
9	LoRa	SF11	500kHz	1760
10	LoRa	SF10	500kHz	3900
11	LoRa	SF9	500kHz	7000
12	LoRa	SF8	500kHz	12500
13	LoRa	SF7	500kHz	21900
14-15	Reserved for Future Use			

Physical Layer message format The LoRaWAN specification defines the physical layer message format. The message comprises a preamble, a physical header (PHDR), a physical header Cyclic Redundancy Check (PHDR_CRC), a physical payload (PHY Payload), and an error detection field at the end (CRC). PHDR and PHDR_CRC fields have a combined total size of 20 bits. The 2 bytes of CRC are present only in uplink messages [15].

The ToA is calculated as the sum of the preamble duration and payload duration, converting the respective packet or fragment length from bytes to symbols [17].

The preamble is used to synchronize the receiver and enable the detection of the LoRa chirps.

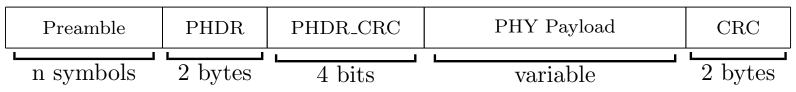


Figure 2.3: LoRaWAN physical layer message format [15]. The length of each field is denoted below.

The preamble is defined as a sequence of programmable number of symbols. The symbol period depends on the SF and BW selected with the following relation:

$$T_{sym} = \frac{2^{SF}}{BW}. \tag{2.2}$$

The duration of the preamble can be calculated as follows:

$$T_{preamble} = (n_{praeamble} + 4.25) \times T_{sym}, \tag{2.3}$$

where $n_{preamble}$ is the number of programmed preamble symbols [17]. The packet and header can be converted into a number of symbols ($payloadSymbNb$) and equals:

$$payloadSymbNb = 8 + max\left(ceil\left(\frac{8PL - 4SF + 28 + 16 - 20H}{4(SF - 2DE)}\right)(CR + 4), 0\right), \tag{2.4}$$

where PL is the number of payload bytes, $H = 1$ when the header is enable and $H = 0$ when no header is present, $DE = 1$ when the low DR optimization is enable and 0 when disable. CR is the Coding Rate and may take values from 1 to 4.

The payload duration can be calculated as the symbol period multiplied by the number of payload symbols as follows:

$$T_{payload} = payloadSymbNb \times T_{sym}. \tag{2.5}$$

Finally, the ToA can be obtained as the sum of the durations of the preamble and the payload:

$$T_{packet} = T_{preamble} + T_{payload}. \tag{2.6}$$

MAC Layer message format The LoRaWAN specifications [3] define three Medium Access Control (MAC) messages that are carried in the physical layer message (*i.e.*, in the physical layer payload). The messages are: the Join message, the Confirmed Data message and the Unconfirmed Data message. Fig. 2.4 shows the LoRaWAN MAC message format.

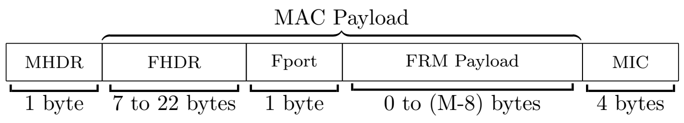


Figure 2.4: LoRaWAN MAC message format [15]. The length of each field is denoted below.

The MAC message is composed of the MAC Header (MHDR), which indicates the MAC message type, the MAC payload, which can carry the application data or a Join message, and the MAC Message Integrity Code (MIC), which allows for an integrity check at the receiver of the MAC message [15]. The MAC payload can carry MAC commands using the frame header (FHDR) that depends on the Fport value. The radio configuration and MAC layer parameters can be changed using the MAC commands. The application data is carried in the FRM Payload field [15].

2.2.1.2 Sigfox

Sigfox technology is designed for the development of IoT applications when the volume of data sent (data from sensor device in most cases) is low (ranging from a few bytes to tens of bytes) and infrequent. Sigfox coverage range is large (tens of kilometers) and its current consumption is very low [18]. Sigfox operates in unlicensed ISM bands and efficiently uses the frequency bandwidth with very low noise level, higher receiver sensitivity, and low cost antenna design by using Binary Shift Keying (BPSK) modulation [19].

Sigfox network architecture is a star topology as shown in Fig. 2.5. The devices (*i.e.*, sensors and actuators) communicate via neighboring Base Stations (BSs). A device can send a message at any time, and it can be received by one or many BSs, as the device is not associated to any BS, reducing the need for signaling and handover [20]. BSs are connected to a single core network, which is located in the cloud, through the public Internet. The core network comprises two elements: the Service Center and the Registration Authority. The Service Center is in charge of controlling and managing the BSs and devices. The Registration Authority is the one responsible for authorizing the network access to the devices. The Service Center provides a web interface and a number of Application Program Interfaces (APIs) to interact with the devices and the data collected by them [20].

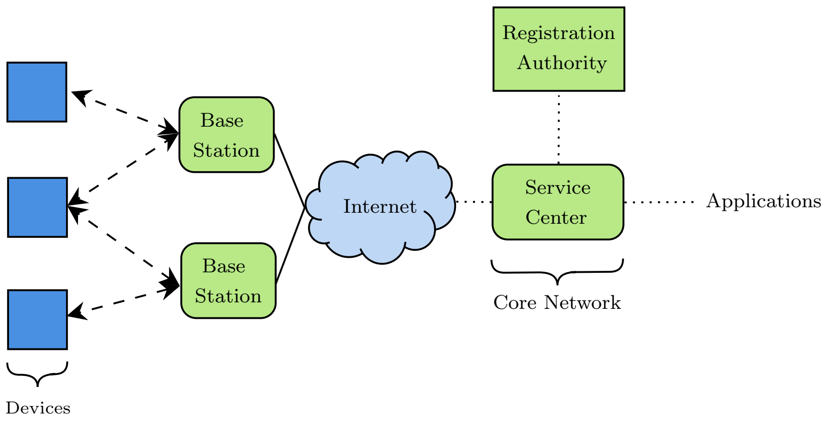


Figure 2.5: Sigfox architecture [1].

Initially, Sigfox only supported uplink communication, but has evolved to support bidirectional traffic [19] as shown in Section 2.2.1.2. In the EU region, Sigfox utilizes the bands from 868.00 MHz to 868.60 MHz for uplink and 869.40 MHz to 869.65 MHz for downlink. In the US region it uses the 902 MHz band [20].

Physical Layer Sigfox uses Ultra Narrow Band (UNB) radio that allows a long link range with limited transmission power for both uplink and downlink communication. The uplink channel bandwidth depends on the region. In the EU region it is 100 Hz and in the US region it is 600 Hz. The downlink channel bandwidth is 1.5 kHz in both regions. Sigfox modulation for the uplink is Differential Binary Phase-Shift Keying (DBPSK) and GFSK for downlink. DBPSK is more efficient than GFSK in bandwidth management, favoring an increase in the uplink range, as the uplink is more power sensitive. Moreover, DBPSK concentrates the received power in a very narrow bandwidth and obtains a high received power level, which yields a good protection against interference. In the EU region, the uplink bit rate is 100 bits/s and in the US region is 600 bit/s. The downlink physical layer bit rate is the same in both regions, *i.e.,* 600 bit/s.

As Sigfox uses a license-free spectrum, *i.e.*, the ISM band, it must comply with the regulation regarding its use. In the EU region, the downlink and uplink bands must enforce a *duty-cycle* of 1% and 10%, respectively [20]. To comply with the *duty-cycle* regulation, Sigfox allows, typically, up to 140 uplink messages and 4 downlink messages per day (see Section 2.2.1.2) [21]. Depending on the region and use case, Sigfox may relax the message volume limitation or give less priority to messages over the limits [1].

Table 2.3: Sigfox uplink and downlink number of messages, payload and throughput [19–21].

Number of Messages over the uplink	140 messages / day
Number of Messages over the downlink	4 messages / device / day
Maximum payload length for every uplink message	12 bytes
Maximum payload length for every downlink message	8 bytes
Maximum uplink throughput (Europe Region)	100 bps
Maximum uplink throughput (US Region)	600 bps
Maximum downlink throughput (Both Regions)	600 bps

Radio configuration and features Sigfox is designed to operate in license-free frequency bands. Sharing unlicensed spectrum presents technical constraints, as use of these frequency bands is subject to local spectrum access regulations. To better handle the diversity of regulations across different world zones, Sigfox defines 7 geographical zones [4]. Each geographical zone is characterized by a specific set of radio features and parameter settings called Radio Configuration (RC) [22]. Table 2.4 shows the RCs that correspond to the 7 Sigfox geographical zones.

Table 2.4: Radio Configuration (RC) for different Sigfox geographical zones.

	RC1	RC2	RC3	RC4	RC5	RC6	RC7
Regions or countries	Europe Middle East Africa	Brazil Canada Mexico USA	Japan	Latin America Asia Pacific	South Korea	India	Russia
Uplink frequency (MHz)	868	902	923	920	923	865	868
Downlink frequency (MHz)	869	905	922	922	922	866	869
Uplink data rate (bit/s)	100	600	100	600	100	100	100
Downlink data rate (bit/s)	600	600	600	600	600	600	600
Spectrum access	Duty Cycle (UL 1%, DL 10%)	Frequency Hopping	Listen Before Talk	Frequency Hopping	Listen Before Talk	-	Duty Cycle (UL 1%, DL 10%)

As shown in Table 2.4, Sigfox uses sub-GHz frequency bands in all RCs. The uplink data rate may be 100 bit/s or 600 bit/s, depending on the considered RC. For example, in RC1 (Europe and

Africa) the uplink data rate is 100 bit/s, while in RC2 and RC4 (North/South America, Asia and Australia) the uplink data rate is 600 bit/s. The downlink data rate is 600 bit/s in all RCs.

A relevant feature of each RC is its associated spectrum access techniques. For example, in RC1 there is a 1% *duty-cycle* constraint that must be enforced in the uplink, while in RC4 a radio transmitter must implement frequency hopping. The *duty-cycle* restriction, under the strictest regulation, allows a maximum of 140 uplink messages and a very limited number of downlink messages per day (*e.g.*, less than 10, depending on the RC) [1]. However, constraints may vary depending on regulatory and system conditions.

Uplink procedure (U-procedure) A device can send data towards the Sigfox Network anytime, provided that radio regulations are enforced. In Sigfox, there are two types of methods to perform an uplink frame transmission: U-procedure, and B-procedure. In the former, the uplink frame does not trigger a response from the Sigfox Network. In the latter, the device requests such a response. This subsection describes the U-procedure.

The uplink frame is transmitted three times using different frequencies. This approach provides time and frequency diversity. Therefore, Sigfox offers a triple diversity, *i.e.*, diversity in time, in frequency, and in space [4]. A U-procedure comprises three different states (Fig. 2.6):

1. transmission (of duration T_{Tx}),
2. wait for next transmission (of duration $T_{Wait_{Tx}}$) and
3. cooldown (of duration T_{Cool}).

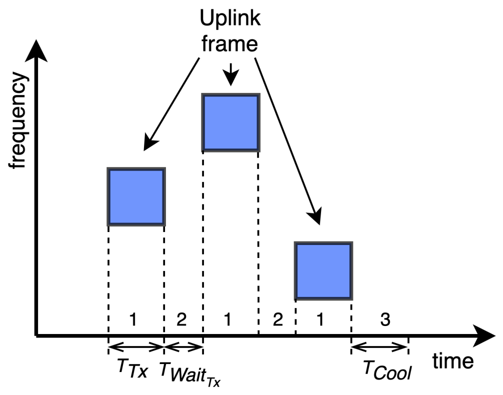


Figure 2.6: Illustration of a Sigfox U-procedure example. State numbers and their corresponding durations are indicated at the lower part of the figure. In the U-procedure, states 1 and 2 are present three and two times, respectively.

Fig. 2.7 shows the Sigfox uplink frame format, which is composed of 10 fields. The Preamble is a 19-bit predefined sequence. The Field Type (FT) is a 13-bit field which signals the message type (*i.e.*, application or control messages). The Length Indicator (LI) is a 2-bit field that provides the length of the Message Authentication Code (MAUTH) (see Table 2.5). The Bidirectional Flag (BF) is a 1-bit field that indicates if the uplink frame is starting a U-procedure (0b0) or a B-procedure (0b1). The Repeated Flag (REP) is a 1-bit field set to 0b0.

The Sigfox uplink frame format also includes a 12-bit Sequence Number. This number is incremented by the device for every uplink frame transmission. At reception, the number is registered by the Sigfox Cloud, which keeps a record of the sequence numbers received. The Device ID is a unique 32-bit value that identifies each device in the Sigfox Network.

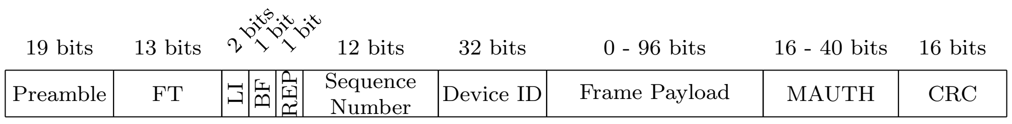


Figure 2.7: Sigfox uplink frame format.

The maximum uplink frame payload size ($L_{UL_{MAX}}$) is 12 bytes. The MAUTH provides frame authentication and has a variable length (L_{MAUTH}) which depends on the frame payload size. The total uplink frame size is fixed to a set of values, *i.e.*, 14, 15, 18, 22, and 26 bytes, which lead to only 5 possible radio burst durations (for each data rate). This small and fixed set of total uplink frame sizes simplifies the radio transmitter and receiver design, and allows to better predict the power consumption (*i.e.*, battery lifetime) of the device for a specific application. To accomplish this, considering that the uplink frame payload field is of variable size (between 0 and 12 bytes), the L_{MAUTH} is selected so that its combination with the uplink frame payload always yields one of the possible total uplink frame sizes (see Table 2.5). The uplink frame integrity is secured by using a CRC of 16 bits.

Table 2.5 shows the Sigfox uplink frame transmission time (T_{Tx}) for 100 bit/s and 600 bit/s, for all possible uplink frame payload sizes (L_{UL}). Note that T_{Tx} has a variable duration and, due to a dependency between L_{MAUTH} and L_{UL}, it exhibits a stepwise behavior as a function of the payload size.

Table 2.5: Uplink frame transmission time as a function of the payload size

L_{UL} (bytes)	L_{MAUTH} (bytes)	LI value (MSB, LSB)	Total frame size (bytes)	T_{Tx} 100 bit/s (ms)	T_{Tx} 600 bit/s (ms)
0	2	00	14	1120	186.67
1	2	00	15	1200	200
2	4	10			
3	3	01	18	1440	240
4	2	00			
5	5	11	22	1760	293.34
6	4	10			
7	3	01			
8	2	00			
9	5	11	26	2080	346.67
10	4	10			
11	3	01			
12	2	00			

Bidirectional procedure (B-procedure) The B-procedure allows a device to send data to, and receive data from, the Sigfox Network. Fig. 2.8 shows a B-procedure example. In contrast with the U-procedure, the 3 uplink frames are now followed by the transmission of a downlink frame and an uplink confirmation frame. The confirmation frame is only sent by the device if the downlink frame is correctly received. The different states involved in a B-procedure, when a downlink frame is sent by the Sigfox Network and received by the device, are the following:

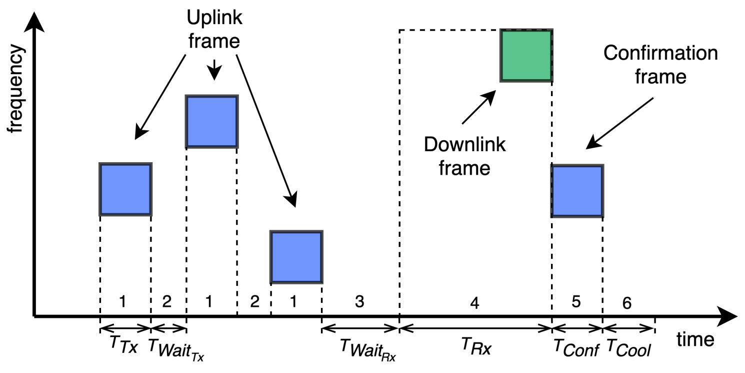


Figure 2.8: Sigfox B-procedure example illustration. State numbers and their corresponding durations are indicated at the lower part of the figure. In this example, a downlink frame is sent by the Sigfox Cloud and received by the device. The latter sends a confirmation frame. In a B-procedure, states 1 and 2 are present three and two times, respectively. T_{Rx} can take values between T_{RxMIN}, when the downlink frame is received at the start of the reception window, and T_{RxMAX}, when no downlink frame is received.

1. transmission (of duration T_{Tx}),
2. wait for next transmission (of duration $T_{Wait_{Tx}}$),
3. wait for next reception (of duration $T_{Wait_{Rx}}$),
4. reception (of duration T_{Rx}),
5. confirmation transmission (of duration T_{Conf}),
6. cooldown (of duration T_{Cool}).

The reception state has a variable duration (T_{Rx}), which depends on the moment in which the downlink frame is transmitted. If the downlink frame is not received by the device, or no downlink frame is sent by the Sigfox Network, the reception state will last for its maximum value ($T_{Rx_{MAX}}$) and the confirmation transmission state will not be present.

Fig. 2.9 shows the Sigfox downlink frame format, distributed as follows. The first field is a 91-bit predetermined Preamble. This field is followed by a 13-bit FT field that indicates a B-procedure is taking place. A downlink Error Correction Code (ECC) of 32 bits, which is the result of a BCH(15,11) error correction code applied over the concatenation of the downlink frame payload and the downlink frame MAUTH. The downlink frame payload size is fixed to 8 bytes. The downlink frame MAUTH has a size of 16 bits and it provides authentication for the downlink frame. The downlink frame integrity is protected by using an 8-bit CRC.

The confirmation frame has the same format as the uplink frame (see Fig. 2.7), with a fixed payload size of 8 bytes. This payload contains information regarding the device battery voltage and Received Signal Strength Indicator (RSSI) estimation made by the device based on the downlink frame received from the BS.

2.2.1.3 NB-IoT

Narrow band IoT (NB-IoT) is the 3rd Generation Partnership Project (3GPP) proposal for the long range, low power, low data rate IoT market [23]. NB-IoT provides an extended coverage of 164 dB maximum coupling loss, battery life of over 10 years, up to 55000 devices per cell and

91 bits	13 bits	32 bits	64 bits	16 bits	8 bits
Preamble	FT	ECC	Frame Payload	MAUTH	CRC

Figure 2.9: Sigfox downlink frame format.

an uplink latency of less than 10 seconds [1]. Fig. 2.10 shows how the capabilities of NB-IoT are located between current Long Term Evolution (LTE) and LPWAN technologies. It was published at Release 13 of 3GPP on June 2016. LTE design has been reused for NB-IoT to a large extent, including numerologies, channel coding and modulation schemes, and higher layer protocols, which allows quick and easy deployment and further development of NB-IoT products within existing LTE networks [24]. Since NB-IoT operates in a licensed spectrum, it has no channel access restrictions allowing up to a 100% *duty-cycle*.

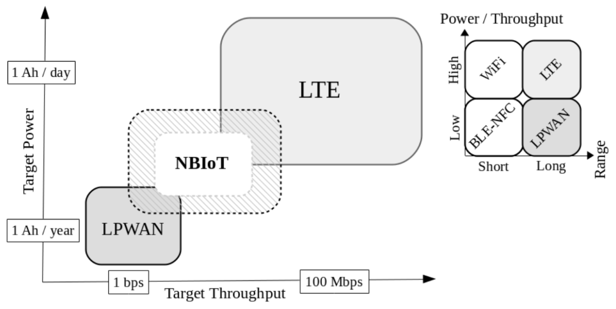


Figure 2.10: NB-IoT features, in comparison with other technologies [23].

Fig. 2.11 shows the 3GPP network architecture, which also applies to NB-IoT. The Mobility Management Entity (MME) is responsible for handling the mobility of the User Equipment (UE). The MME tasks include tracking and paging UEs, session management, choosing the Serving Gateway (S-GW) for the UE during initial attachment and authenticating the user [1]. The S-GW routes and forwards the user data packets through the access network. The Packet Data Network Gateway (P-GW) works as an interface between the 3GPP and external networks. The Home Subscriber Server (HSS) contains the user-related and subscription-related information. It is in charge of user authentication and access authorization. The eNodeB is the BS that controls the UEs in one or several cells.

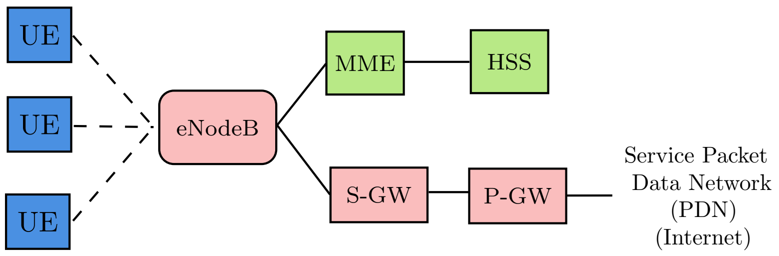


Figure 2.11: NB-IoT architecture [1].

Physical Layer NB-IoT technology uses a frequency band of 180 kHz bandwidth, which corresponds to one resource block in LTE transmission. For uplink, it uses Single-Carrier Frequency Division Multiple Access (SC-FDMA) and Orthogonal Frequency Division Multiple Access (OFDMA) for the downlink. NB-IoT defines three operation modes (see Fig. 2.12):

- *Stand-alone operation*: Uses currently available 200 kHz GSM frequencies. There is a guard interval of 10 kHz.
- *Guard band operation*: Uses unused resource blocks within an LTE carrier's guard-band.
- *In-band operation*: Uses resource blocks within the LTE carrier.

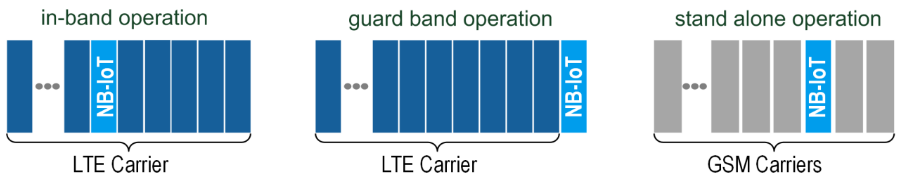


Figure 2.12: NB-IoT operation modes [25].

There are two types of transmissions in NB-IoT: multi-tone and single-tone. Multi-tone transmission uses 15 kHz sub-carrier spacing, 0.5 ms slot and 1 ms subframe, like LTE, and it is based on SC-FDMA. On the other hand, single-tone transmission supports two sub-carrier spacing, namely, 15 kHz and 3.75 kHz. For the sub-carrier spacing of 3.75 kHz, the slot duration is of 2 ms [26]. Section 2.2.1.3 illustrates the different types of NB-IoT. The data rate is limited to 250 kbps for the multi-tone downlink communication and to 20 kbps for the single-tone uplink communication [13]. The payload size is 1600 bytes [27].

Table 2.6: NB-IoT characteristics [26].

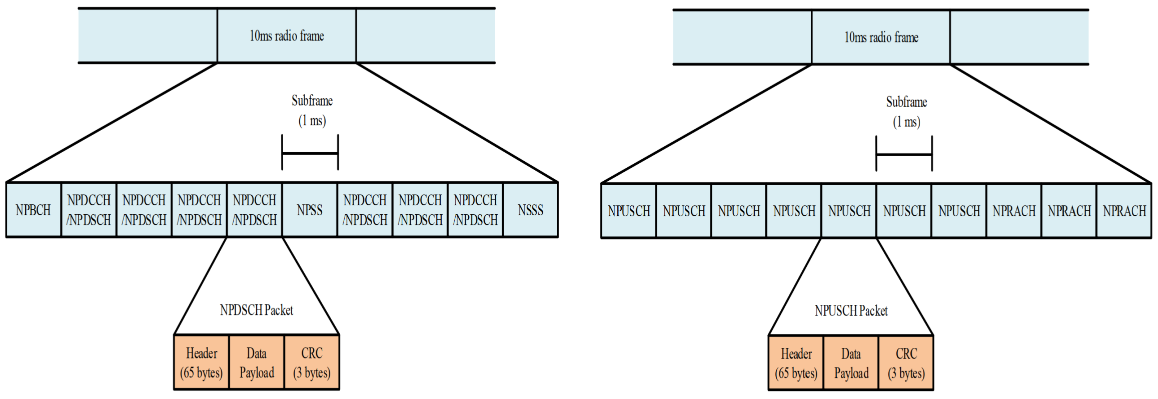
Sub-carrier spacing	**No of tones**	**No of SC-FDMA symbols**	**Transmission time**
15 kHz	12	14	1 ms
15 kHz	6	28	2 ms
15 kHz	3	56	4 ms
15 kHz	1	112	8 ms
3.75 kHz	1	112	32 ms

Physical Layer message format NB-IoT downlink has two physical signals and three physical channels. Fig. 2.13a illustrates how the NB-IoT subframes are allocated to different physical channels and signals in downlink. The two physical signals are the Narrowband Reference Signal (NRS), which provides phase reference for the demodulation of the downlink channel, and the Narrowband Primary and Secondary Synchronization signals (NPSS and NSSS), which are used to perform cell search using time and frequency synchronization and cell identity detection.

The three physical channels are:

- The Narrowband Physical Broadcast Channel (NPBCH), which carries the Master Information Block (MIB) and is transmitted in subframe 0 in every frame.

- The Narrowband Physical Downlink Control Channel (NPDCCH) is the core element of the downlink control channels as it carries control information such as paging, UL/DL assignment, random access channel (RACH) response, type of modulation being used for transmission and power control. Moreover, it controls the data transmission between the BS and the user equipment (UE).
- The Narrowband Physical Downlink Shared Channel (NPDSCH) is the main bearing channel. It consists of the System Information Block (SIB), user unicast data and control information. The downlink frame is located in the NPDSCH channel, and has a Header size of 65 bytes and a 3-byte CRC [28].



(a) Downlink Frame Structure (b) Uplink Frame Structure

Figure 2.13: NB-IoT downlink and uplink frame structure [28]

In uplink, NB-IoT defines a Resource Unit (RU), which is a resource mapping unit, combining the number of subcarriers (frequency domain) and the number of slots (time domain). Furthermore, NB-IoT supports single-tone and multi-tone transmissions and has one physical signal and two physical channels (see Fig. 2.13b). The physical signal channel is called Demodulation Reference Signal (DMRS), which is multiplexed with the data so that it is only transmitted in RUs containing data. The two physical channels are the Narrowband Physical Random Access Channel (NPRACH) which enables the UE to connect to a BS, and the Narrowband Uplink Shared Channel (NPUSCH) which carries both the data and the control information, differently from LTE. The uplink frame structure is composed of a 65-byte header and a 3-byte CRC [28].

2.2.1.4 Other LPWAN technologies

Many other technologies have emerged in the LPWAN spectrum. Some are proprietary and others are standards, developed by well-known organizations. The most relevant proprietary LPWAN technologies are: INGENU RPMA [29], Telensa [30], Qowisio [31], and the previously explained Sigfox. Regarding the LPWAN standards, there are different organizations that have defined different standards: The Institute of Electrical and Electronics Engineers [32] (IEEE) with 802.15.4k, 802.15.4g and 802.11ah; the European Telecommunications Standard Institute [33] (ETSI) with Low Throughput Networks (LTN); the 3GPP [34] with enhanced MTC (eMTC), Extended Coverage GSM (EC-GSM) and the previously explained NB-IoT; the Weightless-SIG [35] with Weightless-W, Weightless-N and Weightless-P; and the DASH7 Alliance with DASH7 [13]. Table 2.8 gives the most important characteristics of some of the LPWAN proprietary and standard technologies [13].

Table 2.7: Technical specifications of various LPWAN technologies [13].

Standard /	IEEE		WEIGHTLESS-SIG			DASH7 Alliance	INGENU	TELENSA
Technology	802.15.4k	802.15.4g	WEIGHTLESS-W	WEIGHTLESS-N	WEIGHTLESS-P	DASH7		
Modulation	DSSS, FSK	MR-(FSK, OFDMA, OQPSK)	16-QAM, BPSK, QPSK, DBPSK	UNB DBPSK	GMSK, offset-QPSK	GFSK	RPMA-DSSS(UL), CDMA(DL)	UNB 2-FSK
Band	ISM SUB-GHZ & 2.4GHz	ISM SUB-GHZ & 2.4GHz	TV white spaces 470-790MHz	ISM SUB-GHZ EU (868MHz), US (915MHz)	SUB-GHZ ISM or licensed	SUB-GHZ 433MHz, 868MHz, 915MHz	ISM 2.4GHz	SUB-GHZ bands including ISM: EU (868MHz), US (915MHz), Asia (430MHz)
Data rate	1.5 bps-128 kbps	4.8 kbps-800 kbps	1 kbps-10 Mbps	30 kbps-100 kbps	200 bps-100kbps	9.6,55.6,166.7 kbps	78kbps (UL), 19.5 kbps(DL)	62.5 bps(UL), 500 bps(DL)
Range	5 km (URBAN)	up to several kms	5 km (URBAN)	3 km (URBAN)	2 km (URBAN)	0-5 km (URBAN)	15 km (URBAN)	1 km (URBAN)
No of Channels / orthogonal signals	multiple channels. Number depends on channel & modulation		16 or 24 channels(UL)	multiple 200 Hz channels	multiple 12.5 kHz channels	3 different channel types (number depends on type & region)	40 1MHz channels, up to 1200 signals per channel	multiple channels
Forward error correction	YES	YES	YES	NO	YES	YES	YES	YES
MAC	CSMA/CA, CSMA/CA or ALOHA with PCA	CSMA/CA	TDMA/FDMA	slotted ALOHA	TDMA/FDMA	CSMA/CA	CDMA-like	-
Topology	star	star, mesh, peer-to-peer (depends on upper layers)	star	star	star	tree, star	star, tree	star
Payload length	2047B	2047B	>10B	20B	>10B	256B	10KB	-
Authentication & encryption	AES 128b	AES 128b	AES 128b	AES 128b	AES 128/256b	AES 128b	16B hash, AES 256b	-

Table 2.8: Technical specifications of various LPWAN technologies [13].

2.2.2 Wake-up radio systems as low rate data channel

The WuR system is a MAC based technique that reduces the power consumption by maintaining the main radio off most of the time and only ”wakes up” when a message needs to be transmitted or received. This reduces the *duty-cycle* of the main radio of the device, reducing the power consumption and optimizing the battery lifetime. One characteristic of the WuR system is the use of a secondary radio, whose main feature is low power consumption and main function is to detect a Wake-up Signal (WuS). In this section, we explain the basic WuR system and how it can be used as a secondary low rate channel for data exchange, focusing on Wi-Fi radio.

2.2.2.1 Wake-up Wi-Fi and secondary radio

Wi-Fi has been considered as an option to fulfill the connectivity requirements for the IoT. In spite of the massive adoption Wi-Fi device compliant family (*i.e.*, IEEE 802.11), with 628 million public access points (APs) [36], the power consumption of IEEE 802.11 devices is too high for many use cases considered in the IoT. To solve this problem, the IEEE issued IEEE 802.11 Power Saving Mode (PSM), which makes the device radio periodically toggle, turning it off for most of the time. The off period ranges from several beacon intervals in IEEE 802.11 up to several years in IEEE 802.11ah. Some extra functionality is required in the Access Point (AP) to avoid losing frames sent to a sleeping Station (STA). When the STA wakes up, according to the program schedule, it requests all the frames queued in the AP. PSM introduces a trade-off between latency and power consumption, as the communication has to wait until the next scheduled wake-up. Therefore, PSM is not suitable for low-latency applications [5].

To solve the low-latency problem introduced by PSM in IEEE 802.11, but still targeting low power devices, the WuR concept was introduced. When using WuR, the main radio is only activated to receive incoming transmissions and is kept off for the rest of the time. Since the main radio is off, WuR proposed the implementation of a secondary radio receiver, the Wake-up Radio Receiver (WuRx), which receives asynchronous wake-up requests from other devices. The wake-up requests are called Wake-up Signal (WuS) and are generated by the main or secondary transmitter, called Wake-up Transmitter (WuTx), of other devices in the network. The WuRx must be of low power consumption to allow a continuous operation, making the communication available on-demand and reducing the trade-off between latency and power consumption, as the STA main radio can be sleeping when not needed. Due to the growing interest on WuR technologies, the IEEE started standardization efforts within IEEE 802.11 with the IEEE 802.11ba amendment.

A coordination mechanism is required as devices need to turn on their radios in a coordinated manner to maintain communication. Such a mechanism is called *rendez-vous* scheme and determines the method used to coordinate the wake-up of devices in the network [5]. Fig. 2.14 shows a WuR implementation of a purely asynchronous *rendez-vous* scheme. In WuR systems, devices turn off their main radio but keep a low power secondary radio continuously active. The *rendez-vous* on WuR occurs as follows: 1) the Initiator device uses its WuTx (*i.e.*, the main or secondary radio) to send a WuS to the Target device; 2) the Target device receives the WuS with its WuRx; 3. the Target device WuRx wakes up its controller from sleep; 4) the Target device controller activates its main radio; and finally, 5) the main radios of both devices are active and the *rendez-vous* has occurred.

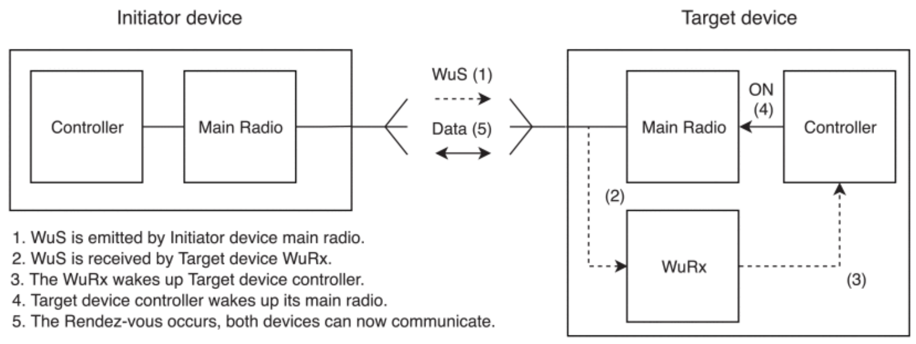

Initiator device
Target device
Controller
Main Radio
WuS (1)
Data (5)
Main Radio
ON
(4)
Controller
(2)
WuRx
(3)
1. WuS is emitted by Initiator device main radio.
2. WuS is received by Target device WuRx.
3. The WuRx wakes up Target device controller.
4. Target device controller wakes up its main radio.
5. The Rendez-vous occurs, both devices can now communicate.

Figure 2.14: Diagram illustrating the operation of WuR system. [5].

Fig. 2.14 shows a *rendez-vous* where both devices may operate under the same technology, *i.e.*, IEEE 802.11. However, the possibility of the main radio to communicate with a secondary low power radio, using another modulation scheme over the same communication channel (*e.g.*, in IEEE 802.11ba using On-Off-Keying (OOK) or Peak-Flat modulation [5]) provides a new low rate channel, not only for wake-up proposes, but also for data exchange.

Fig. 2.15 shows two devices (Device A and B) that communicate using the secondary radios as WuRx and the main radio as WuTx. The communication is initiated when a WuS is sent from Device A to Device B (see Fig. 2.16). Device B controller wakes up the main radio and sends an ACK indicating that it is ready. The ACK is sent using its main radio as WuTx. Device A receives the ACK using the secondary radio and proceeds to send the data. Unlike IEEE 802.11ba, which only provides a wake-up mechanism to initiate the communication over the main radio (using the same standard *i.e.*, IEEE 802.11), the communication channel between main and secondary radios allows devices using different standards to communicate between each other. For example, in Fig. 2.15, even if Device A main radio is IEEE 802.11 and Device B main radio is IEEE 802.15.4, they are still able to communicate using the secondary radio as a low rate channel. The use of this low rate channel to interconnect different network technologies may leverage new possibilities, as the creation of an adaptation layer for IPv6 support over secondary radio channels. By using an adaptation layer for IPv6 support over secondary radio channels, it is possible to interconnect and provide full interoperability between wireless technologies without the use of a gateway.

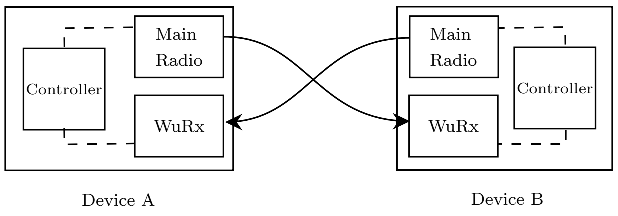


Figure 2.15: WuR system using the secondary radio as a low rate data channel. Each device main radio communicates with the other device WuR

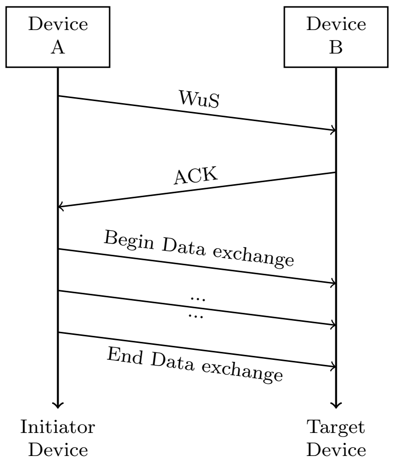


Figure 2.16: Example of a low rate data channel using the secondary radio for data exchange and the WuS to begin communication. An ACK is sent by the Target Device to notify the Initiator Device that the WuRx is ready for data reception.

2.3 IPv6 over low rate low power wireless network technologies

2.3.1 Motivation

LRLPWN technologies, overviewed in Section 2.2, will provide connectivity to billions of IoT devices. IPv6 is the optimal protocol in order to provide Internet connectivity to this vast number of devices, connected to heterogeneous networks. The characteristics of IPv6, such as its large set of available addresses, tools for unattended operation, intrinsic interoperability and improved security, make it suitable to solve the interoperability problem present in the IoT domain. However, IPv6 was not designed for networks with constrained resources *i.e.*, constrained energy, memory, processing and communication, as it was designed for networks where resources are available, such as Ethernets or Wi-Fi. New problems arise when using IPv6 over LRLPWNs as they are characterized by a reduced data unit length [37], sometimes smaller than the IPv6 header itself.

Therefore, an adaptation functionality is required to support and optimize IPv6 over constrained-node networks [37]. This functionality can be modeled as an adaptation layer, *i.e.*, a protocol stack layer placed between the IPv6 and the MAC or link layer of the wireless radio technology. This adaptation layer must be designed to efficiently enable IPv6 over that technology, providing lightweight encoding formats (*e.g.*, IPv6 header compression), support for data transport (*e.g.*, fragmentation and reassembly, especially in technologies with short frame payload size) and energy efficiency, for devices that operate with limited power resources (see Fig. 2.17).

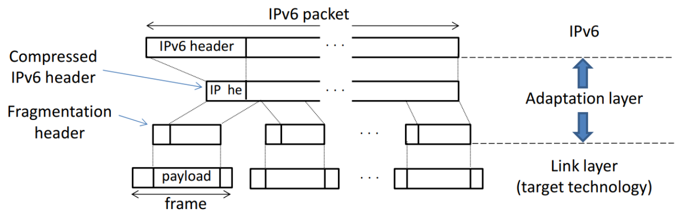


Figure 2.17: Adaptation layer functionality example [37].

The IETF IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN) WG completed in 2012 an adaptation layer to support IPv6 over IEEE 802.15.4, low rate, low power wireless radio interface and the WG was closed [38]. From 2012 until now, the number of low rate, low power radios have grown considerably, as the need to extend IPv6 support over these new networks. To tackle this problem, the IETF IPv6 over Networks of Resource-constrained Nodes (6Lo) WG was created in 2013, targeting IPv6 adaptation for a set of technologies, called 6Lo technologies, such as Bluetooth Low Energy (BLE), Digital Enhanced Cordless Telecommunications Ultra Low Energy (DECT ULE), Near Field Communication (NFC) and IEEE 802.11ah [37].

With the emergence of LPWAN technologies, the adaptation proposed by the 6Lo WG becomes insufficient, since the extreme constraints of LPWAN are different from the ones of 6Lo technologies (see Fig. 2.18).

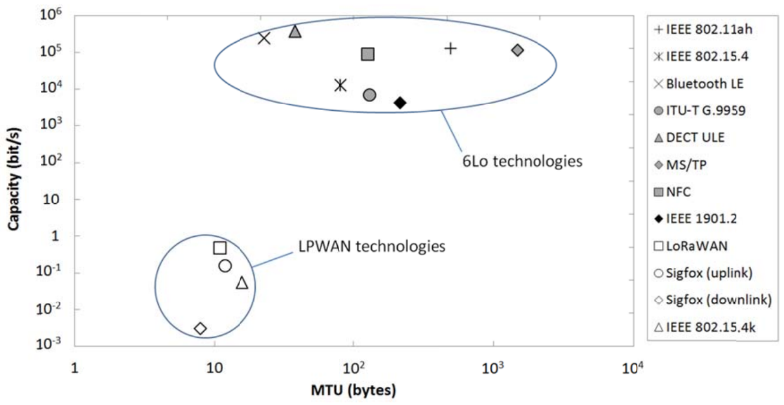


Figure 2.18: MTU and capacity comparison between 6Lo and LPWAN technologies [37].

The challenges of supporting IPv6 over LPWAN can be summarized as: 1) small MTU and low bit rate, 2) lack of layer-two fragmentation, 3) severe message rate limitations due to regulatory constraints, and 4) uplink/downlink asymmetry [37]. For example, IPv6 has a header size and a minimum MTU of 40 and 1280 bytes, respectively. Indeed, a new adaptation mechanism is required to enable IPv6 over such extraordinary challenging scenarios, as the level of adaptation provided by existing standards, *i.e.*, 6LoWPAN/6Lo, is insufficient, as LPWAN technologies are more constrain, especially in MTU size. To fill this gap in IPv6 support, the IETF created a new WG to enable IPv6 over LPWAN technologies (IETF LPWAN WG). The main result of the LPWAN WG is a new framework called Static Context Header Compression (SCHC) [6]. SCHC provides header compression for the IPv6 headers using a static context that is shared between the end-device and the gateway/BS. Furthermore, SCHC offers F/R mechanisms that not only provides support for the 1280-byte MTU of IPv6, but also provides optional reliability.

2.3.2 Static Context Header Compression (SCHC)

This section provides an overview of the SCHC framework C/D and F/R modes.

2.3.2.1 SCHC adaptation layer overview

Flagship LPWAN technologies, such as LoRaWAN and Sigfox, are characterized by a reduced L2 MTU [1]. Furthermore, these technologies do not provide a native fragmentation mechanism for transferring larger packets. The SCHC framework provides header C/D and F/R functionalities specifically designed for LPWAN [6]. SCHC defines a set of Rules, each identified with a RuleID, which determine how to perform the compression and fragmentation and allow the sender and receiver to determine the operation mode and configuration parameters.

SCHC is composed of two sublayers, namely the C/D and the F/R sublayers. Fig. 2.19 shows those sublayers between the IPv6 layer and the LPWAN technology layer. When an IPv6 packet

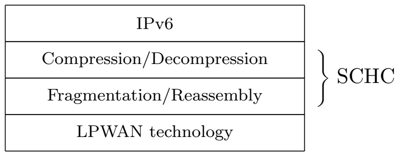


Figure 2.19: Protocol stack illustrating the location of the SCHC sublayers between the IPv6 layer and the underlying LPWAN technology [6].

needs to be sent, compression is performed. The compressed IPv6 packet is called a SCHC Packet. If the SCHC Packet size is greater than the L2 MTU, SCHC fragmentation is performed at the sender that, depending on the operation mode, will respond with a SCHC ACK. At the receiver, the SCHC Packet is reassembled and the IPv6 packet is decompressed. Fig. 2.20 illustrates the operation of SCHC.

2.3.3 SCHC header compression

SCHC header compression is based on a static context shared between the C/D entities located at the sender and receiver. The static context exploits the predictability of IoT traffic (*e.g.*, a device, with known identifiers sending temperature measurements every 10 minutes to a previously configured server), which allows compression of packet header field values known a priori. If a field value is known by both the sender and receiver, it is not necessary to send such field value in uncompressed form. Fields that cannot be compressed entirely are transferred as a compression residue.

The static context is composed of a set of rules, each one identified by a RuleID. A rule contains partial or full values of all the fields of a given packet header, along with matching operators that describe how the field values of a packet to be compressed are compared with the ones in the rule. When an IPv6 packet needs to be sent, the compressor will replace the IPv6 packet header with

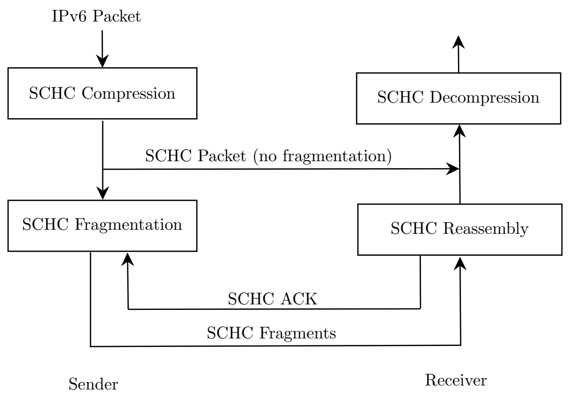


Figure 2.20: SCHC compression and fragmentation process [6].

only the RuleID of the best matching rule, plus a compression residue (if any). The output of the compression process is a SCHC Packet with a header that contains the RuleID and the compression residue, as shown in Fig. 2.21. A rule defines how each field is compressed according to matching operators such as *equal*, *ignore* and *match mapping*. Each field and operator have an associated action applied over the header field, *e.g.*, *not-sent*, *sent*, or *mapping-sent*, among others.

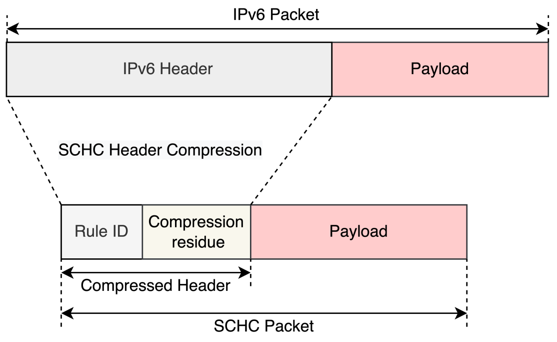


Figure 2.21: The IPv6 header is compressed using SCHC. The output of SCHC compression is called a SCHC Packet.

2.3.4 SCHC fragmentation

In SCHC F/R, a SCHC Packet is fragmented into units called tiles (see Section 2.3.4.3). One or more tiles are carried by one SCHC Fragment, which is sent in an LPWAN frame. In some SCHC F/R modes, a determined number of tiles are grouped into a window (see Section 2.3.4.4), and the receiver generates SCHC ACKs to tell the sender which tiles of that window have been received or not. Missing fragments or tiles are retransmitted. Tiles and windows are numbered in a way that each tile can be identified for further retransmissions (see Fig. 2.22).

2.3.4.1 SCHC Packet format

The SCHC framework specifies a packet format composed of the compressed header and the payload of the original packet. The compressed header is composed of the Rule ID and the Compression

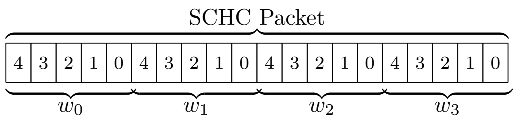


Figure 2.22: Example of a SCHC Packet fragmented into 20 tiles, with 5 tiles per window. The tile index in the window, called the Fragment Compressed Number (FCN), is indicated for each tile.

Residue (if any) which, if present, is the output of compressing the packet header with a determined Rule [6]. Fig. 2.23 shows the structure of the SCHC Packet.

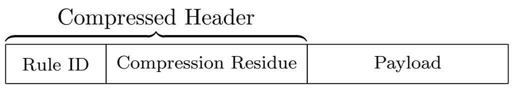


Figure 2.23: Example of a SCHC Packet.

2.3.4.2 SCHC F/R messages and headers

The SCHC framework defines different messages that are used to carry out the SCHC F/R process between the sender and the receiver. The main messages are the SCHC Fragment and the SCHC ACK. Each message has a SCHC F/R Header with the following fields:

- Rule ID: this field identifies whether a SCHC message is a SCHC Fragment. In a SCHC Fragment, it indicates which F/R mode and settings are used.
- Datagram Tag (DTag): this field is used to identify –along with the Rule ID– a SCHC Packet. The length of the DTag field is T in bits.
- W: this field identifies the window number a fragment belongs to and has a length of M bits. W is only present in SCHC F/R modes that use windows.
- Fragment Compressed Number (FCN): this N-bit field is used to identify the progress of the sequence of tile(s) being transmitted in a SCHC Fragment message.
- Reassembly Check Sequence (RCS): this field, of U bytes, is used to check the integrity of a reassembled SCHC Packet. It protects the complete SCHC Packet.
- Integrity Check (C): this one-bit field ($L_{C_{bit}} = 1$) equals 1 if the integrity check of the reassembled SCHC Packet succeeded, and 0 otherwise.

A SCHC Fragment carries a part of a SCHC Packet from the sender to the receiver. The FCN field of a SCHC Fragment has all bits set to 1 (it is then called an All-1 SCHC Fragment), to indicate it is the last fragment for the current SCHC Packet; that fragment carries the RCS for this SCHC Packet. Fig. 2.24 shows a regular and an All-1 SCHC Fragment. L_{SH} is the length of the SCHC Fragment Header in bytes. Padding bits are added at the end of the SCHC Fragment if needed by the LPWAN technology. A SCHC ACK is sent by the receiver to the sender to acknowledge the complete or partial reception of the fragmented SCHC Packet. In the latter case, a SCHC ACK reports whether the tiles of a given window have been received or not, in the form of a bitmap (see Section 2.3.4.5). Fig. 2.25 shows the SCHC ACK format. A SCHC ACK carries the C field.

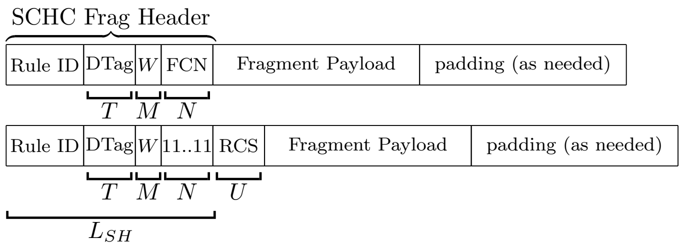


Figure 2.24: Illustration of a regular SCHC Fragment, an All-1 SCHC Fragment with the RCS field and an All-0 SCHC Fragment. The length in bits of each header field is indicated below each field.

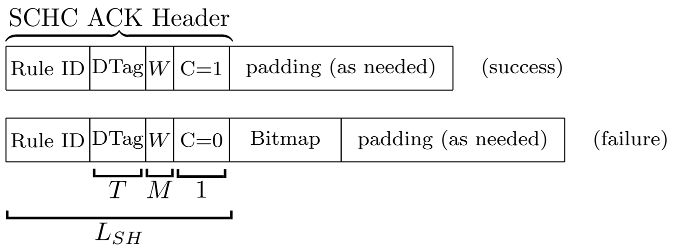


Figure 2.25: Illustration of a SCHC ACK message. The top SCHC ACK message notifies successful reassembly of a SCHC Packet by carrying a $C = 1$. The bottom one indicates a failed SCHC Packet reassembly ($C = 0$) and carries a bitmap. The length of each header field (in bits) is indicated below each field.

2.3.4.3 Tiles

A tile has a size of t bytes. If the payload field is present in a SCHC Fragment, it must carry at least one tile. The FCN of the SCHC Fragment, together with the window index W, identifies the first tile carried by the SCHC Fragment.

2.3.4.4 Windows

A group of w successive tiles is called a window. Each window in a fragmented SCHC Packet transmission, except the last one, must have the same number of tiles. Windows are numbered from 0 upwards. The window field (W) has a size of M bits and the window size ($window\_size$) has to be less than $2^N - 1$ (in each window, the tiles are numbered from $window\_size - 1$ downwards). Fig. 2.22 shows the fragmentation of a SCHC Packet in 4 windows, with 5 tiles per window.

2.3.4.5 Bitmap

A bitmap is a sequence of bits where each bit indicates the received status of a tile within a specific window. The bitmap has a size of $2^N - 1$. The rightmost and leftmost bits will correspond to tile numbers 0 and $window\_size - 1$, respectively. The receiver will set a 1 in the bitmap when the corresponding tile is received successfully and a 0 when the tile was not received, as exemplified in Fig. 2.26. The C field is set to 0, indicating the packet reassembly was not successful, mostly because the SCHC Packet was not received completely.

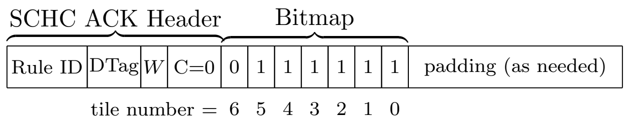


Figure 2.26: Example of a SCHC ACK, where a window of 7 tiles ($w = 7$) was sent and tile FCN # 6 was not received successfully.

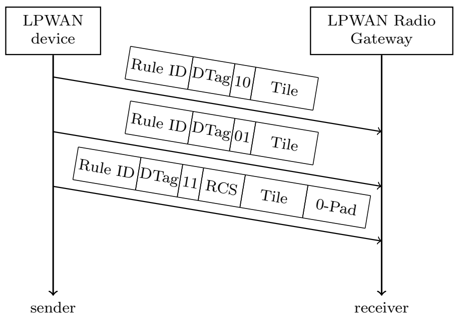


Figure 2.27: Example of a SCHC Packet transfer that requires fragmentation in the No-ACK mode. The SCHC Packet needs 3 fragments to be carried.

2.3.5 SCHC Fragmentation and Reassembly modes

SCHC offers 3 SCHC F/R modes to perform the SCHC F/R process: No-ACK, ACK-Always and ACK-on-Error. If a reliable communication is required, ACK-Always and ACK-on-Error use ACKs to support potential retransmission upon failure. This section provides a brief description of No-ACK and ACK-Always modes, and a more detailed explanation of the ACK-on-Error mode.

2.3.5.1 No-ACK mode

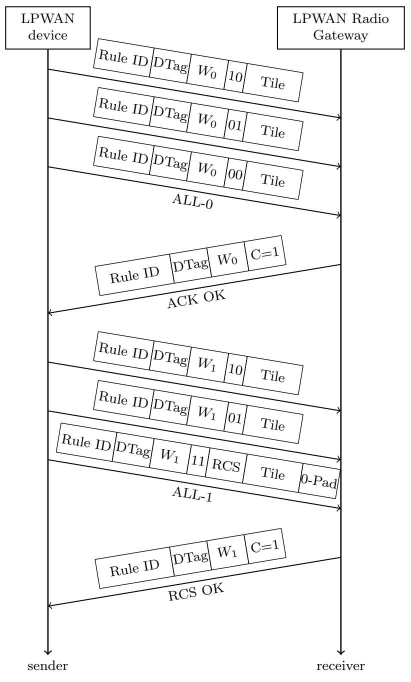
The No-ACK mode provides a mechanism for in-sequence delivery of SCHC Fragments between the sender and the receiver. This mode does not provide reliability when errors are present, since there is no feedback from the receiver and the sender cannot perform SCHC Fragment retransmissions (see Fig. 2.27). Variable L2 MTU size is supported. Tiles can be of different sizes, while windows are not used.

2.3.5.2 ACK-Always mode

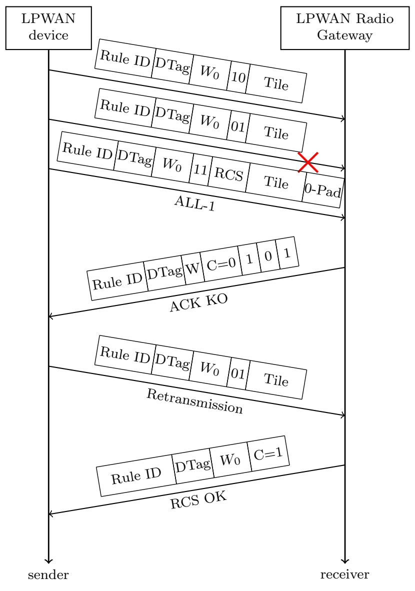
The ACK-Always mode is a window-based mechanism for in-sequence delivery of SCHC Fragments that supports reliability (see Fig. 2.28). At the end of the transmission of each window, a SCHC ACK is sent by the receiver to the sender to report on the tiles received for the current window. The sender only begins the transmission of the next window, once the receiver confirms the correct reception of all tiles of the current window. Variable L2 MTU size is not supported. Tiles can have different sizes.

2.3.5.3 ACK-on-Error mode

The ACK-on-Error mode is a window-based mechanism that supports reliable and out-of-order delivery of SCHC Fragments and variable L2 MTU. This SCHC F/R mode reduces the number of SCHC ACKs, when compared to ACK-Always, since in all windows except for the last one carrying


LPWAN device
LPWAN Radio Gateway
Rule ID DTag W_0 10 Tile
Rule ID DTag W_0 01 Tile
Rule ID DTag W_0 00 Tile
ALL-0
Rule ID DTag W_0 C=1
ACK OK
Rule ID DTag W_1 10 Tile
Rule ID DTag W_1 01 Tile
Rule ID DTag W_1 11 RCS Tile 0-Pad
ALL-1
Rule ID DTag W_1 C=1
RCS OK
sender
receiver

(a) Without transmission errors.



(b) With transmission errors.

Figure 2.28: Example of a SCHC Packet transfer that requires fragmentation in the ACK-Always mode. In (a), the SCHC Packet transfer requires 2 windows of 3 tiles, for a total of 6 tiles. In (b) the SCHC Packet transfer requires 1 window of 3 tiles, as the SCHC Packet needs only 3 fragments to be carried. Losses are present in the first (and only) window in (b).

a SCHC Packet, SCHC ACKs are only sent when at least one tile is lost. For the last window, in order to ensure that the sender can expect to receive feedback on the fragmented SCHC Packet transmission, a SCHC ACK is unconditionally sent.

All tiles must be of the same size, except for the last one, which can be smaller. One or more tiles can be carried in a SCHC Fragment and they can be from multiple windows.

In a SCHC Fragment, the window number (W) needs to be set to identify each window number unambiguously during the transmission of a SCHC Packet. The sender can retransmit the SCHC Fragments for any lost tiles, from previous windows. This allows the sender and the receiver to work in a loosely coupled manner.

Transmission of a fragmented SCHC Packet may end in three ways: first when the integrity check for the SCHC Packet shows a correct reassembly at the receiver, second when too many retransmission attempts were made, and finally when an inactivity timer at the receiver indicates that the transmission has been inactive for too long. Fig. 2.29a shows the transmission of the fragmented SCHC Packet of Fig. 2.22 with no errors, where the SCHC Fragment size allows 4 tiles per fragment.

If the receiver receives the All-1 SCHC Fragment, it performs the integrity check for the SCHC Packet. This is carried out by comparing the RCS calculated with the RCS received in the All-1 SCHC ACK Fragment. With the result of the Integrity Check, the C field is populated with 1 or 0, indicating success or failure of the SCHC Packet reassembly, respectively. In case some tiles of a window or a complete window were lost, the receiver prepares the bitmap for the lowest-numbered window that was not entirely received.

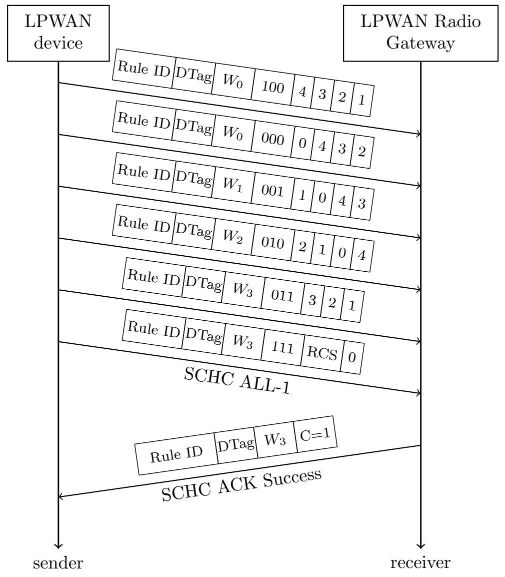
The sender has to listen for a SCHC ACK from the receiver after sending the All-1 SCHC Fragment. Moreover, a technology-oriented profile specification can establish other times when the sender may need to listen for a SCHC ACK, for example after sending a complete window of tiles. The sender can terminate the transmission of a SCHC Packet when receiving a SCHC ACK with $C = 1$. If the SCHC ACK carries $C = 0$, the sender must resend the SCHC Fragments corresponding to the missing tiles indicated in the bitmap. As an example, Fig. 2.29b shows the transmission of the SCHC Packet presented in Fig. 2.22 with an error in the 4th SCHC Fragment. Even though the SCHC Fragment carries tiles from 2 windows, the SCHC ACK indicates the window number of the lower-numbered window and the sender is able to identify which SCHC Fragment has to be retransmitted. After the retransmission of the missing SCHC Fragment, the receiver computes the RCS, performs the integrity check and sends a SCHC ACK reporting successful SCHC Packet reassembly.

2.3.6 Counters and Timers

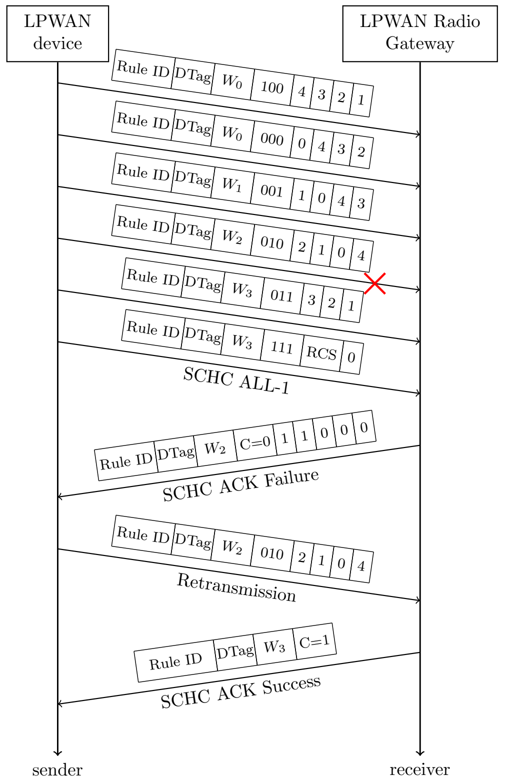
In all SCHC F/R modes, upon SCHC Fragment reception, the receiver resets a timer called Inactivity Timer. When this timer expires, the SCHC Packet transmission is aborted due to inactivity. Furthermore, in reliable SCHC F/R modes, the sender and the receiver keep a counter that is incremented when a SCHC ACK is requested and sent, respectively. The SCHC ACK will be requested and sent a maximum of MAX_ACK_REQUESTS times. The sender will wait, after requesting each SCHC ACK, for a time given by a timer called the Retransmission Timer (RT). If MAX_ACK_REQUESTS is reached, the SCHC Packet transfer is aborted by either the sender or the receiver.

2.3.7 SCHC Profiles

The SCHC framework specification was purposefully defined following a generic approach, in order to offer flexibility for adapting to different underlying LPWAN technologies and scenarios [6]. As shown earlier in this section, SCHC provides several F/R modes and a number of parameters, but it does not specify which F/R modes or parameter settings should be used over each particular


LPWAN device
LPWAN Radio Gateway
Rule ID | DTag | W_0 | 100 | 4 | 3 | 2 | 1
Rule ID | DTag | W_0 | 000 | 0 | 4 | 3 | 2
Rule ID | DTag | W_1 | 001 | 1 | 0 | 4 | 3
Rule ID | DTag | W_2 | 010 | 2 | 1 | 0 | 4
Rule ID | DTag | W_3 | 011 | 3 | 2 | 1
Rule ID | DTag | W_3 | 111 | RCS | 0
SCHC ALL-1
Rule ID | DTag | W_3 | C=1
SCHC ACK Success
sender
receiver

(a) Example of a transmission of the SCHC Packet shown in Fig. 2.22 with no errors. The transmission ends with a SCHC ACK that indicates successful SCHC Packet reassembly.


LPWAN device
LPWAN Radio Gateway
Rule ID | DTag | W_0 | 100 | 4 | 3 | 2 | 1
Rule ID | DTag | W_0 | 000 | 0 | 4 | 3 | 2
Rule ID | DTag | W_1 | 001 | 1 | 0 | 4 | 3
Rule ID | DTag | W_2 | 010 | 2 | 1 | 0 | 4
Rule ID | DTag | W_3 | 011 | 3 | 2 | 1
Rule ID | DTag | W_3 | 111 | RCS | 0
SCHC ALL-1
Rule ID | DTag | W_2 | C=0 | 1 | 1 | 0 | 0 | 0
SCHC ACK Failure
Rule ID | DTag | W_2 | 010 | 2 | 1 | 0 | 4
Retransmission
Rule ID | DTag | W_3 | C=1
SCHC ACK Success
sender
receiver

(b) Example of a transmission of a SCHC Packet with one lost SCHC Fragment in ACK-on-Error mode. Once the All-1 SCHC Message is received by the receiver, a SCHC ACK with $C = 0$ and its corresponding bitmap is generated. The sender performs the SCHC Fragment retransmission. When the lost SCHC Fragment is received, the receiver performs the Integrity Check and sends the corresponding SCHC ACK Success indicating the end of the transmission of the SCHC Packet.

Figure 2.29: ACK-on-Error mode.

LPWAN technology. Such details are deferred to SCHC Profiles, which are technology-specific documents that specify the parameter settings and F/R modes to be used over a given LPWAN technology. As of the writing, SCHC Profiles have been or are being defined for LoRaWAN, NB-IoT, and Sigfox [7–9]. Exploiting the generic design of SCHC [6], and its efficiency, SCHC Profiles are also being defined for use of SCHC over protocols and technologies beyond the LPWAN field, such as the Point-to-Point Protocol (PPP) or IEEE 802.15.4 [39, 40].

A SCHC Profile needs to provide information regarding the most common intended use cases, implementation recommendations, mapping of SCHC elements into the corresponding LPWAN architecture, and recommendations for C/D and/or F/R utilization. SCHC Profiles for LPWAN technologies focus mainly on specifying details for F/R functionality [7–9]. Such details comprise the F/R modes to be used and the size of each SCHC Fragment header field, among others.

CHAPTER 3

PERFORMANCE ANALYSIS OF SCHC F/R MODES

<<*Light gets the shape of what it enlightens.*>>

In this chapter we present a performance evaluation of all SCHC F/R modes. As the SCHC framework was evolving into becoming a standard, we evaluated its performance with different parameter settings. We collaborated in the development of SCHC simulation tools and contributed with a statistical module to obtain the values of the different performance metrics. The evaluation was performed for three critical network and application performance parameters, such as Channel Occupancy (CO), Goodput and Total delay. We believe that the results of this investigation provided valuable insights on how the SCHC framework can be configured, and the impact of those configuration values on network and application performance. One SCHC F/R mode outperforms the others for reliable communication: the SCHC ACK-on-Error mode. The contributions have been published in a conference paper [VI].

3.1 Introduction

There are numerous applications for IoT, with various constraints, many of them involving a large number of nodes, each of which having a limited amount of data to send (*e.g.*, one message per day). Another common constraint is the low energy of the nodes, as many IoT devices operate on battery, which should last for extended periods of time (months, years or even decades). These constraints led to the deployment of LPWANs.

The huge number of heterogeneous devices brought by IoT stress the need for strong interoperability. The global Internet already provides this interoperability feature using a common set of protocols. However, these protocols were not designed for low bitrate and high latency networks. This is especially true for IPv6, even though its large address space would be a desirable feature to handle big amounts of IoT devices. As a result, using IPv6 on LPWAN implies numerous challenges, mainly due to its large header overhead (40 bytes). Indeed, having low bitrates means that it takes long ToA to send a large amount of data. This high CO, in turn, reduces the probability of a node to access the radio channel in the case of frequency bands regulated by a *duty-cycle*. For example, the 1% *duty-cycle* constraint of the European 868 MHz band requires devices to keep their radio quiet for 99% of the time.

Another issue with IPv6 is that it requires lower layers to support a MTU of 1280 bytes. However, the maximum payload sizes supported by LPWAN technologies are typically much smaller. For instance, LoRaWAN and Sigfox support MTUs of 242 bytes and 12 bytes, respectively.

To tackle these two problems, the LPWAN WG of the IETF has defined a new framework called SCHC [6]. This framework provides a header compression mechanism, as well as several F/R algorithms to satisfy both the large header and the large MTU size required by IPv6. SCHC follows a technology-agnostic approach and, as a new framework, performance and optimal configuration values are unknown at the moment of writing.

Next sections are organized as follows. In Section 3.2, we provide a comprehensive analysis of the state of the art regarding LPWAN, as well as fragmentation for LPWAN networks. In Section 3.3, we develop and justify the performance metrics that will be used to evaluate the three SCHC F/R modes over LoRaWAN by means of simulation. Results are presented and discussed in Section 3.4.

3.2 Related work

The emergence and popularity of IoT requires the interoperability between heterogeneous systems. This interoperability is made possible with IPv6 as the glue between different wireless technologies on the one hand, and with different applications on the other hand. However, running IPv6 over low powered and lossy network is challenging, and adaptation layers are often required.

6LoWPAN was developed by the eponymous IETF WG to provide IPv6 support on top of IEEE 802.15.4 [41]. This adaptation layer mainly provides fragmentation and header compression to fit the maximum layer 2 MTU [42]. Papadopoulos *et al.* [43] highlighted problems when using fragmentation over a multi-hop network: in some route-over [44] implementations, each intermediate node reassembles the initial data packets, and there is inter-fragment interference on the forwarding path. Later on, the IETF 6Lo WG has defined specific adaptation layers for other short range wireless technologies (*e.g.* BLE, ITU-T G.9959, etc.) [45].

In the LPWAN realm (with technologies such as LoRaWAN, Sigfox or NB-IoT), there is also a need to provide adaptation layers to support IPv6, but the network characteristics are different. LPWANs are operated networks in which devices are usually organized in a star topology, around a radio gateway. The downlink is critical in these networks, and usually comes at a high cost in energy consumption and resources. This constrains the feedback that can be given to connected IoT devices (*e.g.* sensors or actuators) to its minimum. In practice, it prevents negotiations over the wireless medium, and greatly reduces the number of ACKs that can be sent. Furthermore, the maximum layer 2 MTU offered by most LPWAN technologies is significantly smaller than that of 6LoWPAN or 6Lo technologies. These are the main reasons why the IETF LPWAN WG has defined the SCHC framework [6] in order to provide fragmentation [16] [46] [47], and header compression [48] over LPWAN technologies such as LoRAWAN.

Suciu *et al.* [16] analyzed the tradeoff between packet sizes, *i.e.*, the optimal number of fragments, and goodput in LPWAN. While we are conducting a performance evaluation of the LoRaWAN adaptation layer to support IPv6 (*i.e.*, data packets up to 1280 bytes), Suciu *et al.* evaluated whether sending several fragments is more efficient than sending large data packets (up to 250 bytes). They carried out Matlab simulations to study if fragmentation of a 250-bytes packet can show benefits in comparison to not using fragmentation. When considering a *duty-cycle* of 1%, they actually showed that packet fragmentation increases reliability, with a higher impact in denser and slower networks (*e.g.*, with a higher SF). The study highlights that there is a trade-off between goodput performance, energy consumption and latency. However, in non-*duty-cycle*-restricted networks, they showed that throughput decreases when using thirty fragments per data packet or more.

Moons *et al.* [46] and Ayoub *et al.* [47] studied the benefits of using SCHC or 6LoWPAN for LoRaWAN networks. They performed the computation of the overhead in terms of headers and number of packets, and proposed an implementation for the OSS-7 operating system in [46]. They showed that the overhead is twenty times smaller with SCHC than with 6LoWPAN. In [47] the authors described an implementation using the network simulator ns-3, but did not provide a performance analysis of the ACK mechanisms.

While there are many of LoRa performance studies in terms of DR and coverage (*e.g.* [49] [50] [51] [52] [53]), to the best of our knowledge, only Suciu *et al.* [16], Moons *et al.* [46] and Ayoub *et al.* [47] discuss SCHC. In this chapter, we study in depth the performance of SCHC F/R modes for data packets with a size up to 1280 bytes (IPv6 MTU), and we provide an overview of the ACK mechanisms which have never been studied previously.

3.3 Performance metrics

In this section, we present the definition of the metrics used to evaluate the performance of the SCHC F/R modes presented in Section 2.3.5. These metrics are: total CO, goodput, and total delay at the SCHC layer.

3.3.1 Total channel occupancy

The total Channel Occupancy ($\mathrm{CO_T}$) is defined as the amount of time the channel is busy during the transmission of a SCHC Packet. That is, the time required by the sender to transmit all SCHC Fragments plus, in the case of ACK-Always and ACK-on-Error, the time needed to transmit all SCHC ACKs. $\mathrm{CO_T}$ is calculated using the CO of each transmission and can be divided in two parts: the CO of the sender ($\mathrm{CO_{tx}}$) and the CO of the receiver ($\mathrm{CO_{rx}}$). The general formula of $\mathrm{CO_T}$ is given by:

$$\mathrm{CO_T}[s] = \mathrm{CO_{tx}}[s] + \mathrm{CO_{rx}}[s], \tag{3.1}$$

where $\mathrm{CO_{tx}}$ (respectively, $\mathrm{CO_{rx}}$) is the channel occupancy of all the fragments transmitted by the sender (respectively, the receiver).

Note that in LoRa/LoRaWAN networks, CO mainly depends on physical layer parameters, such as spreading factor (SF), channel bandwidth [17], and coding rate [16] [15]. The larger the SF, the lower the LoRaWAN MTU size, and the higher the CO required for the transmission.

3.3.2 Goodput

We define the goodput as the ratio between the size of the original SCHC Packet and the size of all fragments and ACKs transmitted. It takes into account the SCHC Headers, payload and padding bits:

$$\text{Goodput}[\%] = (\text{Packet size}/\text{Total data sent}) \times 100. \tag{3.2}$$

3.3.3 Total delay

Total delay at the SCHC layer ($T_{\mathrm{d\ SCHC}}$) is defined as the duration between the start of the transmission of the first SCHC fragment, and the end of the transmission (No-ACK), or the reception of the confirmation that the SCHC Packet has been successfully transmitted (ACK-Always and ACK-on-Error).

In more details, for ACK-Always and ACK-on-Error, $T_{\mathrm{d\ SCHC}}$ includes the CO of all the SCHC Fragments and the corresponding T_{off} between transmission, except for the T_{off} after the transmission of the last SCHC fragment. This is the time spend until the sender receives the SCHC ACK, signaling a successful reception of the whole SCHC Packet. For the No-ACK mode, $T_{\mathrm{d\ SCHC}}$ is the time spend between the beginning and the end of the transmission.

In Europe, LoRaWAN frequency band is restricted by *duty-cycle* [49] and the T_{off} can be expressed as:

$$T_{\mathrm{off}}[s] = \mathrm{CO}[s] \times \frac{100 - \textit{duty-cycle}}{\textit{duty-cycle}}. \tag{3.3}$$

3.4 Simulations

In this section, we present the simulation results for the performance metrics presented in Section 3.3. We used the OpenSCHC simulator [54] to evaluate the performance of the three fragmentation methods, in an ideal scenario, with no channel error nor collision. OpenSCHC is an open source implementation of the SCHC Framework written in micropython. For this study, we created a program to obtain statistics of each SCHC fragment transmission and adapted the simulator to support ACK-Always, which was not implemented in the simulator.

Rules are configured with $L_r = 6$ bits, $T = 2$ bits, $N = 3$ (which implies a maximum window size of $N_{\text{tiles/window}} = 7$), and $U = 32$ bits. This yields a SCHC header of 11 bits for No-ACK. For ACK-Always, M is supposed to be 1. However, because of the way we implemented ACK-Always in OpenSCHC, the simulations were actually performed with $M = 3$ (that is, a penalty of 2 bits per fragment with respect to a correct implementation). Consequently, the total SCHC header length for ACK-Always is 14 bits. Finally, ACK-on-Error was configured with $M = 3$ bits, which means that there is at most 8 windows per SCHC Packet. This also gives a SCHC header length of 14 bits.

Regarding tile sizes, we selected 49 bytes when using a LoRa SF of 12 (SF12), and 240 bytes using a LoRa SF of 8 (SF8), in order to match LoRaWAN MTU for the EU 868-880 MHz ISM band. This last setting does not apply for No-ACK since OpenSCHC does not require a tile size in that mode.

$N_{\text{tiles/window}}$ is set to 2 and 5 tiles for ACK-Always. A window size of 2 tiles requires more windows per packet than a window size of 5 tiles. Thus, we expect a larger overhead with smaller values of $N_{\text{tiles/window}}$, as SCHC ACKs are sent at the end of each window. In the ACK-on-Error mode, $N_{\text{tiles/window}}$ is set to 5 tiles, but the actual value has no impact in the absence of noise, as only one SCHC ACK is sent for the whole transmission (no negative ACK).

As radio technology below SCHC, we consider LoRaWAN, working in the EU 863-880 MHz ISM band with one uplink channel (for data transmission) and one downlink channel. This configuration imposes a *duty-cycle* of 1% by regulatory restrictions.

OpenSCHC provides a realistic simulation of the SCHC protocol implementation. However, it does not provide an implementation for ACK-Always. To simulate the ACK-Always mode, we considered that the All-0 SCHC fragment is carrying an RCS for the currently transmitted window. We also needed to set $M = 3$ instead of 1 as required by the standard. This increases the overhead in the ACK-Always mode, but was necessary to perform simulations.

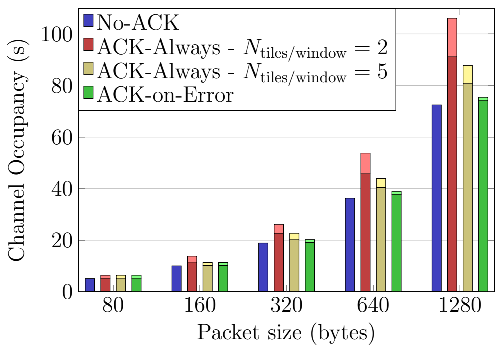
3.4.1 Total channel occupancy

The CO_T is directly related to the SF and the number of fragments exchanged. Figs. 3.1a and 3.1b show the CO_T for SF8 and SF12, respectively. As expected, the only difference between ACK-on-Error and No-ACK over an ideal communication channel is the SCHC ACK at the end of the transmission. Comparing ACK-Always and ACK-on-Error, we notice no difference when the packet size is smaller than the window size (*i.e.*, only one window is required). Such a difference only appears when the packet size is larger than the window size (*i.e.*, more than one window is needed), as a result of the additional SCHC ACKs required for ACK-Always. The difference in number of SCHC ACKs between ACK-on-Error and ACK-Always is proportional to the number of windows required (*i.e.*, the number of additional SCHC ACKs in ACK-Always), and so is the case for CO_{rx}. Moreover, we considered that ACK-Always must send an RCS in the All-0 SCHC Fragment (at the end of each window), implying a larger CO_{tx} when $N_{\text{tiles/window}}$ is smaller.

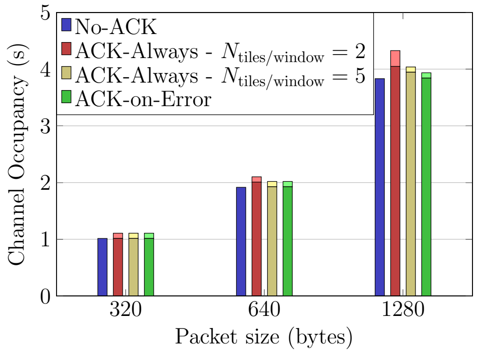
CO_T varies significantly with SF because of the difference in MTU sizes. For instance, sending a 1280-bytes IPv6 packet with SF8 will require a CO_T approximately 20 times greater than with SF12.

No-ACK mode does not use windows nor SCHC ACKs. Thus, CO_T is only composed of the CO_{tx} component, which is related to the total number of fragments transmitted by the sender. For this reason, and as expected, No-ACK is the SCHC F/R mode with the lowest CO_T. The results in Fig. 3.1 confirm that ACK-Always is the method with the largest receiver overhead. Also, CO_T in ACK-Always depends directly on the windows size: as more windows are required to transmit a SCHC Packet, the SCHC ACK overhead and the CO_T increase.

Note that in terms of CO_T, considering the scenarios of Fig. 3.1, the cost of the extra reliability of ACK-Always over No-ACK ranges from 9% ($N_{\text{tiles/window}} = 5$, SF8, packet size of 320 bytes) to 46% ($N_{\text{tiles/window}} = 2$, SF12, packet size of 1280 bytes). This extra CO_T is however much more moderate between ACK-on-Error and No-ACK: ranging from 4% (SF12, packet size of 1280 bytes) to 9% (SF8, packet size of 320 bytes).



(a) SF12, tile size of 49 bytes.



(b) SF8, tile size of 240 bytes.

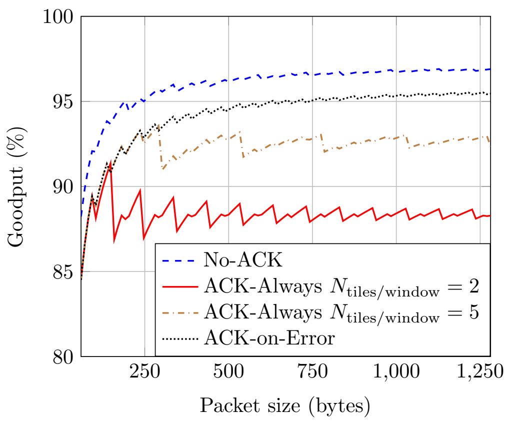
Figure 3.1: Channel Occupancy vs packet size. Darker colors correspond to CO_{tx}, and lighter colors correspond to CO_{rx}.

3.4.2 Goodput

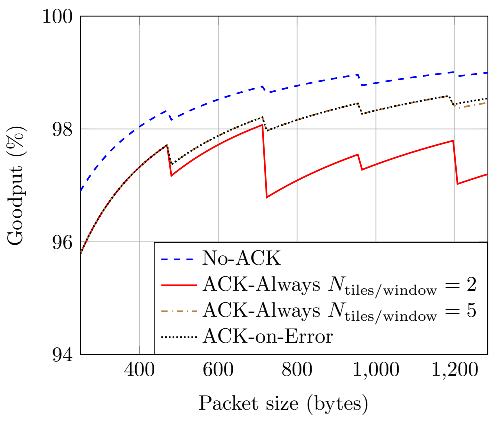
Goodput results are presented in Fig. 3.2 for SF12 and SF8. As a trend, goodput is negatively impacted by the overhead induced by the SCHC F/R mechanisms and tends to an upper limit as the SCHC Packet size grows higher. In the details, though, all three modes show a sawtooth behaviour. This sawtooth profile is due to the extra overhead induced when the fragmentation process requires one more fragment to send the SCHC Packet. In ACK-Always, this sawtooth behaviour is amplified when more windows are needed, as more All-0 Fragments (including a RCS, in our implementation) are emitted.

As expected, No-ACK is the method with the lowest overhead, due to the absence of SCHC ACKs, therefore it is the method with the best goodput ratio. On the opposite side, ACK-Always yields the lowest goodput. However, as more fragments are sent in the same window, a better goodput is obtained, because more data is transferred and acknowledged for the same amount of SCHC ACKs. Finally, the results show that ACK-on-Error provides a trade-off between No-ACK and ACK-Always. Furthermore, Fig. 3.2a confirms that, in the best case, ACK-Always reaches the same goodput as ACK-on-Error. This happens when only one window is needed for ACK-Always (thus only one SCHC ACK is sent).

Comparing Figs. 3.2a and 3.2b, one can observe that the lower the SF, the greater the goodput. This happens because higher SF implies smaller layer-2 MTU (hence, a smaller fragment sizes). In this case, more fragments are required for a given IPv6 packet size, which increases overhead.



(a) SF12, tile size of 49 bytes.



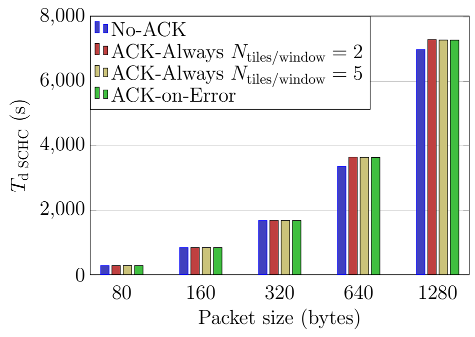
(b) SF8, tile size of 240 bytes.

Figure 3.2: Goodput vs packet size.

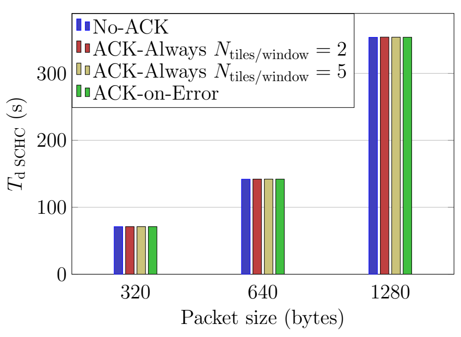
3.4.3 Total delay

Total delay for the three SCHC F/R modes is presented in Figs. 3.3a and 3.3b for SF12 and SF8, respectively. In the simulated scenario, the three SCHC F/R modes perform more or less the same. Only No-ACK presents a somewhat lower $T_{\text{d SCHC}}$ for high SF and SCHC Packet sizes, as a result of a lower CO_{tx} (hence, a lower T_{off}). In simulations of Fig. 3.3, we considered that SCHC ACKs generated between windows were received by the sender during its T_{off}. Under this assumption, there is a small difference between the $T_{\text{d SCHC}}$ of ACK-Always and that of ACK-on-Error.

The main teaching of Fig. 3.3 is that delays are high, even for moderate packet sizes and low SFs. Fig. 3.3a shows that even if SCHC F/R mechanisms can provide support for the maximum IPv6 MTU, there are cases where such packet sizes are not practical. For instance, $T_{\text{d SCHC}}$ for SF12 and a 1280 byte IPv6 packet is 2.0 hours for the ACK-on-Error Mode. Using SF8, the time drops to 5.9 minutes (0.1 hours, 20 times smaller than SF12). For smaller packet sizes, the $T_{\text{d SCHC}}$ difference is still considerable. For example, using a 320-byte IPv6 packet, the difference of $T_{\text{d SCHC}}$ between SF12 and SF8 is in the same order of magnitude: $T_{\text{d SCHC}}$ is 23 times higher with SF12, compared to SF8. In our scenario, these high values of $T_{\text{d SCHC}}$ are mainly due to *duty-cycle* restrictions. As an example, in a non-*duty-cycle*-restricted network, a 1280 byte packet



(a) SF12, tile size of 49 bytes.



(b) SF8, tile size of 240 bytes.

Figure 3.3: $T_{\text{d SCHC}}$ vs packet size.

would only require a $T_{\mathrm{d\ SCHC}}$ of 72.65 s using the ACK-on-Error mode and SF12, instead of the 2 hours of our 1% *duty-cycle* scenario.

Interestingly, while experimenting with SCHC parameters, we found out that a small variation in the SCHC header size can have large implications in terms of $T_{\mathrm{d\ SCHC}}$. Not only because of the extra CO associated with the transmission of the additional bits in the SCHC header, but also because of the subsequent T_{off} the sender must await after each transmission.

CHAPTER 4

PERFORMANCE ANALYSIS AND OPTIMAL TUNING OF SCHC ACK-ON-ERROR MODE

<<Light makes visible all origin and destination sources.>>

LPWANs have a bottle-neck in downlink traffic (*i.e.*, from gateway to device), mainly due to duty-cycle restrictions and network capacity. Therefore, optimizing the downlink messages number and size becomes critical, especially for ACK traffic. In this chapter, a mathematical model of the SCHC ACK-on-Error ACK burden was built, and the configuration values for the main SCHC ACK-on-Error parameters, when using SCHC over Sigfox and LoRaWAN, were proposed and optimized. We define ACK pooling (pooling a large number of fragment errors in a single ACK), which provides benefits for high error rates, while smaller ACKs are better for low error rates. Then, we quantify the advantages of using the optimized values in terms of ACK number and ACK size. Contributions in the optimal configuration parameters discovery have provided useful insights for the development of the SCHC standard, and for use of SCHC over LoRaWAN and over Sigfox. Part of the results in this chapter were obtained during a research stay in the IMT-Atlantique in Rennes, France, and have been published in the journal article [I].

4.1 Introduction

This chapter focuses on the the SCHC F/R ACK-on-Error mode, which is promising due to its reliability and high efficiency by minimizing the number of ACKs compared to ACK-Always. In this mode, ACKs are sent by the fragment receiver only when the latter detects fragment losses. Then, the fragment sender selectively retransmits any lost fragments reported in the ACKs. To maintain consistency, a final ACK is also unconditionally sent at the end of the fragmented packet transmission.

A key constraint for LPWAN is the amount of downlink traffic, *i.e.*, from the gateway to the IoT device. Indeed, in the spectrum band used by unlicensed LPWAN technologies in Europe (*e.g.*, the 868 MHz band), each device must respect a *duty-cycle* limit, which can be especially binding for gateways, that manage many flows. Even if the downlink traffic is not limited by the *duty-cycle* constraint, one may still want to minimize it for economic reasons: the amount of downlink traffic is now used by some operators as a basis for charging IoT users in some plans[1] or the number of downlink messages is limited [20]. Noting that IoT flows are mostly uplink flows (*e.g.*, from the IoT device, say a sensor, sending its data readings), the focus in this chapter is on the *ACK traffic* incurred by the SCHC F/R process –the main expected type of downlink traffic– which needs to be minimized in order to reduce costs and/or respect gateway *duty-cycle* constraints [12].

[1]See, *e.g.*, `https://businessapps.swisscom.ch/en/apps/25128/iot-connectivity-lorawan/editions` (accessed on 19/01/2023)

More specifically, in this chapter we present a mathematical model to compute the volume of ACK traffic (relative to the IoT device data volume) based on the quality of the radio link and the SCHC F/R parameters. We then use the model to optimize those parameters in order to minimize the ACK traffic. For a direct practical use of our results, we provide the optimal parameter values for LoRaWAN [3] and Sigfox [4] technologies.

The remainder of this chapter is organized as follows. In Section 4.2, we present work related to LPWAN fragmentation and its performance. The mathematical model used to analyze the ACK-on-Error mode is developed in Section 4.3, and it is used in Section 4.4 to evaluate the performance metrics, mostly regarding the amount of ACK messages. Section 4.5 explains how our results can be applied to state-of-the-art LPWAN technologies such as LoRaWAN and Sigfox.

4.2 Related work

Recent attention on LPWAN technologies has partly focused on the evaluation of the physical and link layers [12,15,18,20]. In [12], the authors analyzed LoRaWAN and explored its limitations. The results showed that the application and network design must minimize the number of acknowledged frames to avoid capacity drain, because the LPWAN gateway must enforce a time-off following the transmission to comply with *duty-cycle* regulations. Other studies provide a mathematical model that characterizes LoRaWAN and Sigfox end-device energy consumption, lifetime and provided results for the energy involved in data delivery [15, 20]. In [18], a performance evaluation of Sigfox scalability is presented. Together, these studies provide important insights into the physical and link layers of LPWAN technologies, but do not consider the compatibility with IPv6, nor fragmentation mechanisms.

Several studies compare different LPWAN technologies [19, 26, 27]. While Mroue et al. [26] perform an evaluation using the packet error rate for Sigfox, LoRa, and NB-IoT, a comprehensive and comparative study for a number of performance metrics is presented in [27]. The study in [19] evaluates, by simulation, the influence of the number of devices on the packer error rate, collisions and spectrum utilization for Sigfox and LoRa. None of these studies address the problem of transmitting a fragmented IPv6 packet over LPWAN.

Regarding the upper layer functionalities in LPWAN, the study in [16] evaluates the effect of fragmentation and its efficiency in terms of energy consumption, throughput, goodput, and average delay in dense networks. Suciu et al [16] showed that fragmentation increases reliability, especially when sending several fragments instead of only one of the MTU size, *i.e.*, when no fragmentation is needed. Other contributions focus on IPv6 over LPWAN using SCHC [46–48, 55–57]. Some of these propose enhancements to SCHC Header compression, but do not consider SCHC F/R [48, 56]. On the other hand, Moons et al. [46] compared SCHC and 6LoWPAN compression and fragmentation functionalities. Their results show that SCHC has a smaller footprint, uses less memory, and the overhead is twenty times smaller when compared with 6LoWPAN. Ayoub et al. [47] present an implementation of SCHC using the ns-3 network simulator and also compare SCHC with 6LoWPAN, finding the same performance advantage for SCHC in terms of header overhead. The authors in [55] provided an overview of SCHC and a simple evaluation of the different F/R modes, but it is a high level study due to its tutorial purpose. In [57], the authors compared the different SCHC F/R modes in terms of total channel occupancy, goodput, and total delay at the SCHC layer in an ideal communication channel. The authors showed that, when comparing the reliable SCHC F/R modes, namely, ACK-Always and ACK-on-Error, this latter provides a better goodput.

To the best of our knowledge, no previous work provides a mathematical model to estimate the ACK volume and its relation with key configuration parameters of SCHC F/R ACK-on-Error mode, nor contribute with configuration guidance based on radio link quality and packet size.

We think that this mode is worth analyzing, because it provides reliable communication while minimizing the number of ACKs when compared to ACK-Always.

4.3 Mathematical model and analysis

This section provides the mathematical model used to calculate the expected number of SCHC ACKs, hereafter called ACKs, required to successfully transfer a fragmented SCHC Packet in ACK-on-Error mode, in presence of losses. On this basis, the section also introduces a number of related crucial performance metrics. Sections 4.3.1 to 4.3.4 present how to compute the performance metrics, namely: ACK message overhead, ACK bit overhead, ACK bit overhead with L2 Headers, and percentage of used bits per fragment, respectively. In addition, Section 4.3.5 presents useful parameters such as the maximum window size and the maximum bitmap sizes.

In our analysis, we consider an infinite maximum number of fragment retries, all tiles of the same size and no padding in the bitmap. We do not count the unconditional SCHC ACK (see Fig. 2.25) generated by the receiver to notify that all tiles of a SCHC Packet have been received successfully because it does not provide more information to the analysis, as it is always sent, whether there are fragment losses or not.

4.3.1 ACK Message Overhead

The ACK message overhead, E_k is defined as the expected number of ACKs required to successfully transfer a given window. E_k depends on the average number of SCHC Fragments necessary to transmit all the tiles of a window and on the probability that each SCHC Fragment is successfully received.

If we assume that all bits must be received without errors for the transmission to be successful, the probability of success, P_s, for a SCHC Fragment can be related to the Bit Error Rate (BER) through the relation

$$P_\mathrm{s} = (1 - BER)^{8F}, \tag{4.1}$$

where F is the fragment size in bytes of the L2 MTU of the underlying LPWAN technology. (Note that we implicitly assume a uniform BER that refers to the residual BER after application of physical layer error correcting techniques; the impact of the frame header size is considered separately, see Section 4.3.3.) Assuming that P_s is fixed and all transmissions are independent, the number of transmission attempts N_TA needed to successfully deliver a SCHC Fragment from the sender to the receiver follows a geometric distribution, *i.e.*,

$$\mathbb{P}(N_\mathrm{TA} = n) = (1 - P_\mathrm{s})^{n-1} P_\mathrm{s}. \tag{4.2}$$

Recall that at most only one ACK per window is generated *i.e.*, the ACK reports all the tiles not received in that window. Hence, until all SCHC Fragments containing all tiles of a window have been successfully received, a *negative* ACK is sent for that window and the missing SCHC Fragments are re-sent.

As a result, the number of such negative ACKs that are sent per window is the maximum number of retransmissions over all the SCHC Fragments in the window. The expected value of that number can be computed recursively, using a renewal argument. To do so, let A_j be the expected number of transmission attempt cycles (a cycle meaning that we send all the missing SCHC Fragments of a window once) until successful reception of all SCHC Fragments, when $j > 0$ SCHC Fragments remain to be sent. Then, consider the situation after a transmission attempt cycle of all those j SCHC Fragments: with probability $\binom{j}{i} P_\mathrm{s}^{j-i}(1 - P_\mathrm{s})^i$, there have been $j - i$ SCHC Fragments successfully received, and i SCHC Fragments that need retransmission. From that situation, the expected number of transmission attempt cycles is A_i, hence the recursive relation

$$A_j = 1 + \sum_{i=0}^{j} \binom{j}{i} P_\mathrm{s}^{j-i}(1-P_\mathrm{s})^i A_i, \tag{4.3}$$

where the first term accounts for the transmission attempt we considered. Rearranging, we get

$$A_j = \frac{1}{1+(1-P_\mathrm{s})^j}\left[1 + \sum_{i=0}^{j-1} \binom{j}{i} P_\mathrm{s}^{j-i}(1-P_\mathrm{s})^i A_i\right] \tag{4.4}$$

Using (4.4) with $A_0 = 0$, the number of ACKs required per window is simply $A_k - 1$, removing the last successful transmission attempt cycle, when all the SCHC Fragments with all the tiles of that window were correctly received, with k the number of SCHC Fragments per window.

A SCHC Fragment can contain tiles from several consecutive windows, hence the number of SCHC Fragments to successfully send a window may vary. For a fixed window size w (in tiles) and fragment size f (in tiles), the number of SCHC Fragments per window, k, can be modeled as a random variable, as follows:

$$k = \begin{cases} k_1 = \lfloor \frac{w}{f} \rfloor, & \text{with probability } 1 + \lfloor \frac{w}{f} \rfloor - \frac{w}{f} \\ k_2 = \lceil \frac{w}{f} \rceil, & \text{with probability } \frac{w}{f} - \lfloor \frac{w}{f} \rfloor, \end{cases} \tag{4.5}$$

as illustrated in Fig. 4.1.

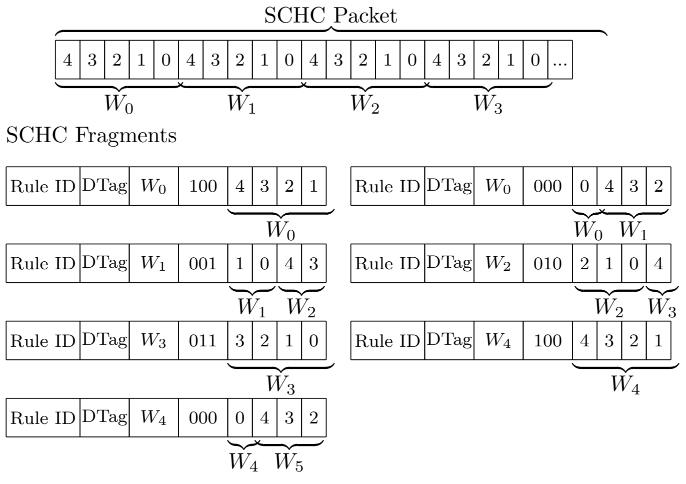

SCHC Packet
4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 ...
W_0 W_1 W_2 W_3
SCHC Fragments
Rule ID DTag W_0 100 4 3 2 1
Rule ID DTag W_0 000 0 4 3 2
Rule ID DTag W_1 001 1 0 4 3
Rule ID DTag W_2 010 2 1 0 4
Rule ID DTag W_3 011 3 2 1 0
Rule ID DTag W_4 100 4 3 2 1
Rule ID DTag W_4 000 0 4 3 2
W_0 W_1 W_2 W_3 W_4 W_5

Figure 4.1: Example of a SCHC Packet and SCHC Fragments. The SCHC Packet is fragmented in 5 tiles per window ($w = 5$) and the SCHC Fragments can transport 4 tiles per fragment ($f = 4$). For a successful transmission of the packet, W_0 needs $k_2 = 2$ successful SCHC Fragment transmissions, then W_1, W_2, and W_3 need each $k_1 = 1$ more successful SCHC Fragment transmission, in compliance with (4.5): a proportion $\frac{w}{f} - \lfloor \frac{w}{f} \rfloor = 1/4$ of windows need $k_2 = \lceil \frac{w}{f} \rceil$ new SCHC Fragment transmissions, whereas the other windows need only $k_1 = \lfloor \frac{w}{f} \rfloor = 1$ new SCHC Fragment transmission.

To compute E_k, the expected number of transmission attempts for k_1 and k_2 are first obtained from (4.4). E_k is the weighted average of $A_{k_1} - 1$ and $A_{k_2} - 1$, with the weights of (4.5). This gives us E_k for a given window size, fragment size, and tile size:

$$E_k = \left(1 + \lfloor \tfrac{w}{f} \rfloor - \tfrac{w}{f}\right) \times A_{\lfloor \frac{w}{f} \rfloor} - 1 + \left(\tfrac{w}{f} - \lfloor \tfrac{w}{f} \rfloor\right) \times A_{\lceil \frac{w}{f} \rceil} - 1. \tag{4.6}$$

4.3.2 ACK Bit Overhead

The ACK bit overhead (O_{ACK}) is defined as the average number of (negative) ACK bits sent for each data bit sent and provides the trade-off between the amount of data that can be transferred in a window and the resulting ACK(s) volume.

O_{ACK} is obtained by dividing E_k by the window size and then multiplying the result by the ACK size (L_{ACK}). The window size can be obtained as the tile size (t) multiplied by the number of tiles in a window (w). L_{ACK} is equal to the bitmap size, that exactly equals w (one bit per tile in a window) plus the ACK header L_{SH} (in bytes) allowing to express the ACK bit overhead as:

$$O_{\text{ACK}} = \frac{(8L_{\text{SH}} + w) \cdot E_k}{8wt}. \tag{4.7}$$

O_{ACK} quantifies the relation between the quantity of data and the quantity of ACKs that need to be sent. The ACK size (L_{ACK}) is related to the tile size. For the same window size (in bytes), larger tiles produce smaller bitmaps, thus smaller ACKs. Recall that O_{ACK} is pertains to the SCHC framework layer, as it only considers the SCHC Headers.

4.3.3 ACK Bit Overhead with L2 headers

As the SCHC framework is on top of a LPWAN technology, there is an extra L2 overhead. To consider the additional cost involved in sending an ACK, the L2 header (with size L_{L2H} in bytes) is added to O_{ACK} in the downlink, *i.e.*, a penalty for sending an ACK, and can be calculated as follows:

$$O_{\text{ACK}_{\text{L2}}} = \frac{(8 \cdot (L_{\text{L2H}} + L_{\text{SH}}) + w) \cdot E_k}{8wt}. \tag{4.8}$$

The ACK bit overhead with L2 headers ($O_{\text{ACK}_{\text{L2}}}$) analyzes the impact of the L2 overhead and its relation with the window and tile sizes. Minimizing $O_{\text{ACK}_{\text{L2}}}$ maximizes the uplink data, whereas optimizing the ACK size and volume, reduce the utilization of the LPWAN gateway and leverages the available resources in *duty-cycle*-constrained networks.

4.3.4 Percentage of used bits per fragment

The percentage of used bits per SCHC Fragment provides information on how efficient a tile size is for a given L2 MTU (F) considering the SCHC Headers. Due to LPWAN capacity constraints, the tile size must be set to maximize the SCHC Fragment payload.

Therefore, the percentage of usage of a SCHC Fragment, which we will denote by P_U, is given by:

$$P_U = \frac{ft}{F - L_{\text{SH}}} \times 100. \tag{4.9}$$

In LPWAN technologies such as LoRaWAN, F can change during an on-going fragmented packet transmission [3] and the tile size previously chosen may not be multiple of the modified F as it was in the previous one, leaving some bytes unused. The number of unused bytes in a SCHC Fragment (f_{unused}), for a given tile size, is given by:

$$f_{\text{unused}} = (F - L_{\text{SH}}) - (ft). \tag{4.10}$$

For example, a 9-byte tile ($t = 9$) will use all the bytes when $F = 11$, $f = 1$ with $L_{\text{SH}} = 2$, but when having 5 tiles ($f = 5$) of 9 bytes each and $F = 53$, 6 bytes per SCHC Fragment will not be used.

4.3.5 Maximum window and bitmap sizes

The SCHC framework defines a method for F/R on the uplink to transfer SCHC Packets, but it does not propose a SCHC ACK F/R method. Without a fragmentation method, the ACK is limited to one L2 MTU. Therefore, there is a maximum bitmap size (MBS) that leads to a maximum window size (MWS). Moreover, the tile size will limit the maximum amount of data on a window. The maximum bitmap size in tiles, *i.e.*, the number of bits in the bitmap, is therefore given by:

$$\text{MBS} = 8 \cdot (F - L_{\text{SH}}), \tag{4.11}$$

and the maximum window size in bytes is then:

$$\text{MWS} = (t \times \text{MBS}) - U. \tag{4.12}$$

4.4 Performance evaluation

In this section, we use the models derived in Section 4.3 to evaluate the ACK-on-Error mode in terms of the performance metrics presented. The section pursues two main goals. The first one is determining the impact of the main SCHC F/R ACK-on-Error mode parameters, *i.e.*, window and tile sizes, and loss rate, on the considered performance metrics. The second goal is deriving the optimal settings for both window and tile sizes in a wide range of scenarios.

The fragment sizes F used in the performance evaluation range between the minimum ($F = 11$) and maximum ($F = 242$) MTU values in LoRaWAN for US 915 MHz band (US915) and EU 868 MHz band (EU868), respectively. These settings allow analyzing the performance of ACK-on-Error mode for the whole range of values for F in LoRaWAN. Note that the physical layer configuration in LoRaWAN (including data unit size, and thus fragment size) is determined by the DR and SF [3]. The SCHC ACK and SCHC Fragment Header size used is 2 bytes ($L_{\text{SH}} = 2$).

4.4.1 ACK Message Overhead

Fig. 4.2a shows the ACK message overhead E_k as a function of P_{s}, for different window sizes, and for $F = 11$ and $t = 9$ bytes so that $f = 1$ (one tile per fragment), and thus $k = w$ as per Eq. (4.5), and there are no unused bits. As expected, E_k decreases with P_{s}. The difference in E_k between a small window size (*e.g.*, $w = 1$) and a large window (*e.g.*, $w = 143$) is larger for small P_{s} than with high P_{s} values. This happens because E_k has a logarithmic behavior as a function of k (see Fig. 4.2b) and "flattens" when P_{s} increases: for small values of k and P_{s}, a small variation in k produces large changes in E_k. The E_k variation decreases as k increases. For larger P_{s} values, E_k is less dependent on k since only a few ACKs are sent.

For $P_{\text{s}} = 1$, only one positive ACK is generated at the end of the fragmented packet transmission because no SCHC Fragments are lost, but $E_k = 0$ because it only counts negative ACKs, *i.e.*, failed window transmission cycles.

4.4.2 ACK Bit Overhead

We now consider the ACK bit overhead metric O_{ACK} defined in (4.7), with the aim of minimizing it to obtain an optimal tile and window sizes.

4.4.2.1 Optimal tile size

Fig. 4.3a shows the impact of the tile size on O_{ACK} for $F = 11$ and $P_{\text{s}} = 0.9$. For small tiles, O_{ACK} increases faster with the window size than for large tiles. Indeed, a small tile size requires

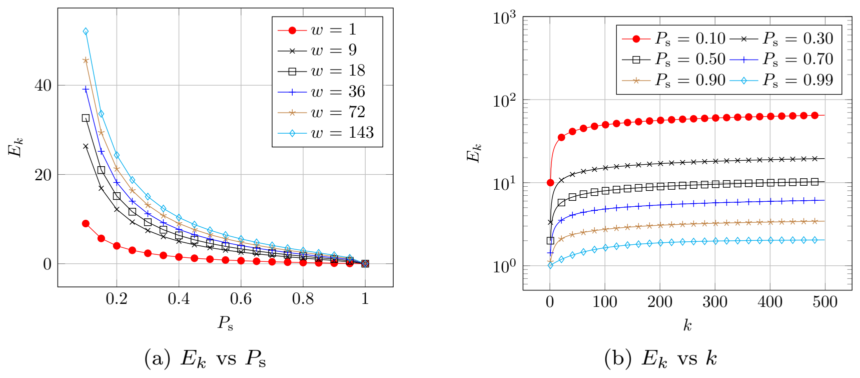


Figure 4.2: E_k for $F = 11$, $t = 9$ and $L_{SH} = 2$ for different window sizes as a function of P_s (a) and k (b).

a large bitmap for each data bit transferred since the bitmap size equals the number of tiles in a window. Hence, the bitmap size is inversely proportional to the tile size for a fixed window (in data volume). As the tile size increases, the ACK size required for the same window size decreases. When the window size increases, the ACK size becomes more relevant in O_{ACK} than the average number of ACKs, because the average number of ACKs has a logarithmic behavior (see Fig. 4.2b) while the ACK size increases linearly with the window size. The tile size that yields the smallest ACK size and the lowest O_{ACK} ratio is the largest possible tile size.

The results for $P_s = 0.9$ are shown in Figs. 4.3c and 4.3d, for $F = 51$ and $F = 242$ respectively.

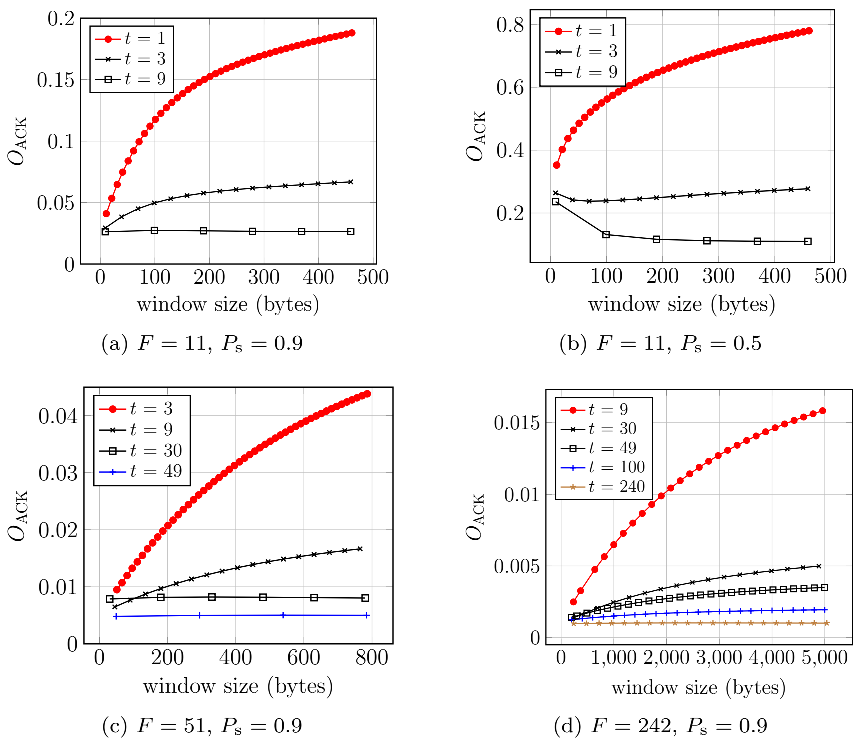


Figure 4.3: O_{ACK} vs window size. Impact of the tile size t on the O_{ACK}. $L_{SH} = 2$.

As in the previous case, a tile size equal to the SCHC Fragment payload size minimizes O_{ACK}. As expected, there is a large difference between using a 9-byte and 49-byte tile, when compared to a 240-byte tile, as Fig. 4.3d shows.

Fig. 4.3b shows O_{ACK} for $P_{\text{s}} = 0.5$ and $F = 11$, evaluated for different window sizes. For larger tile sizes, as the window size increases, O_{ACK} decreases because larger windows are more efficient when reporting many lost fragments.

For all the displayed settings, the optimal tile size is the largest possible tile, *i.e.*, a tile size of the SCHC Fragment payload size, independently of P_{s}. This occurs because the largest tile size maximizes the SCHC Fragment payload while reducing the number of bits in the bitmap, *i.e.*, only one bitmap bit is required for each SCHC Fragment. Hence, as a practical recommendation, for technologies with fixed L2 MTU such as Sigfox, the optimal tile size is simply the SCHC Fragment payload size. For technologies with variable L2 MTU size, such as LoRaWAN, it is not possible to use a tile of the SCHC Fragment payload size in the larger fragment sizes because the tile has to fit in the smallest fragment size to support variable L2 MTU. In the case where the SCHC Fragment size is known beforehand, the Rule can be chosen with the optimal tile and window sizes. If the L2 MTU changes, then the tile size can change accordingly.

4.4.2.2 Optimal window size

Now, the tile size is set to a fixed value, the SCHC Fragment payload size. Figs. 4.4a and 4.4b illustrate the O_{ACK} as a function of P_{s} for different window sizes, for $F = 11$, $t = 9$ and $F = 51$, $t = 49$, respectively. When P_{s} is low, many fragments are lost and a larger window size leads to lower O_{ACK}.

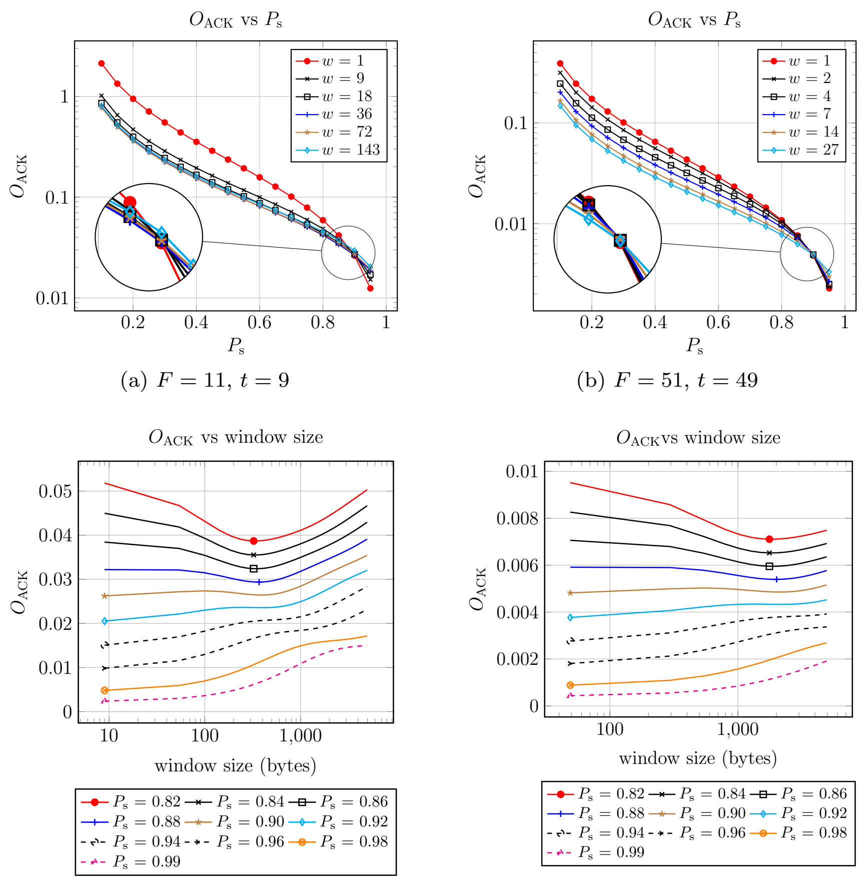
The interest of having large windows lies in what we can call *ACK pooling*, that is the fact that when several fragments (of the same window) fail, the information of the failures is pooled into a single ACK message. Large window sizes benefit from ACK pooling when P_{s} is low because one large ACK can report many lost fragments. Conversely, smaller windows sizes will generate a greater number of ACKs leading to higher overhead.

Henceforth, the tradeoff is between ACK pooling (larger windows mean less ACKs) and ACK size (larger windows mean larger ACKs), that we manage through the total ACK volume metric O_{ACK}.

As P_{s} increases, there exists one point (see zoom in Figs. 4.4a and 4.4b) beyond which smaller windows outperform larger window sizes. From this point onwards, having smaller ACKs becomes more optimal since the number of losses is low. For example, the window size of 1 tile performs worst for small P_{s}, but as P_{s} increases, it becomes the one yielding the lowest O_{ACK}. When P_{s} is greater than 0.9, the window size of just 1 tile is optimal because rarely a SCHC Fragment will be lost, in which case a smaller ACK will be sent (almost no ACK pooling, and smaller ACKs). In contrast, large windows require a large ACK to report the few lost fragments.

As a consequence, there exists an optimal window size that minimizes O_{ACK}, which depends not only on P_{s}, but also on F and t. Figs. 4.4c and 4.4d show the values for O_{ACK} for different windows sizes, and for $F = 11$, $t = 9$ and $F = 51$, $t = 49$, respectively. When P_{s} is 0.8, ACK pooling benefits the larger windows and the optimal window size is large as, for example, 342 bytes ($w = 38$) in Fig. 4.4c. As P_{s} increases, O_{ACK} increases with the window size and for $P_{\text{s}} \geq 0.9$, the benefit of ACK pooling is lost, leading to an optimal window size of one tile ($w = 1$).

Figs. 4.5a and 4.5b illustrate the results for $F = 240$ with $t = 49$ and $t = 240$, respectively. When the tile size is small compared to the SCHC Fragment payload size ($f > 1$), there is no gain from ACK pooling for $P_{\text{s}} > 0.5$. Hence, smaller windows perform better, as Fig. 4.5a shows. As the window size increases, smaller tiles per SCHC Fragment are required when compared to larger tile sizes (see Fig. 4.5b), making the larger window size yield a greater O_{ACK} when using smaller tiles. Fig. 4.5c illustrates how O_{ACK} is minimized for small window sizes when $P_{\text{s}} > 0.80$ for $F = 240$ and $t = 49$: the optimal window size is the smallest window, *i.e.*, $w = 4$. In contrast,



(a) $F = 11$, $t = 9$

(b) $F = 51$, $t = 49$

(c) $F = 11$, $t = 9$

(d) $F = 51$, $t = 49$

Figure 4.4: Impact of the window size w (in tiles) or wt (in bytes) on the ACK overhead O_{ACK}. The optimal window size (w_{opt}) is indicated in (c) and (d) with a mark. $L_{\text{SH}} = 2$.

Fig. 4.5d illustrates for $F = 242$ and $t = 240$ how a larger tile size yields a larger optimal window size and benefits from ACK pooling for $P_s < 0.9$.

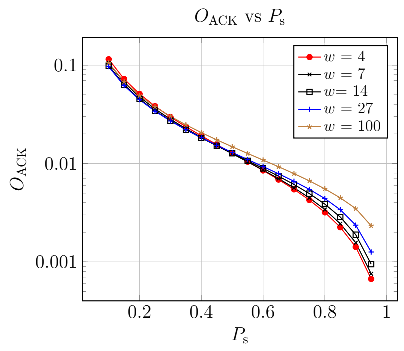
Hence, depending on the parameters and the channel conditions, the window size minimizing O_{ACK} varies. We now focus on that optimal window size.

Fig. 4.6a presents the optimal window size as a function of P_s, for different F values. When P_s is below 0.8, in most cases, the optimal window size is large. The optimal window size decreases as P_s increases. As expected, the importance of ACK pooling decreases with respect to the ACK size. Fig. 4.6b presents the optimal window size for different P_s and for larger F values. As for the case of small F values, as P_s becomes higher, the optimal window size becomes smaller.

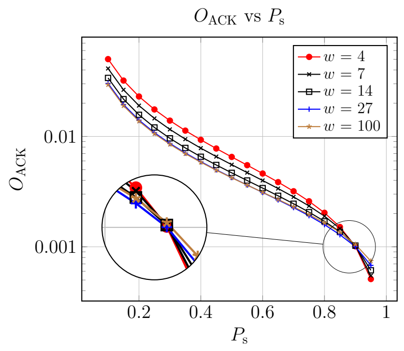
4.4.3 ACK Bit Overhead with L2 headers

The previous two subsections (*i.e.*, Section 4.4.1 and Section 4.4.2) considered the ACK overhead as defined in (4.7), *i.e.*, ignoring L2 headers in the size of ACKs. This may be justified if the operator charges the user based on the volume of L2 payloads. In this section we investigate the impact of counting the whole L2 ACK size by counting those headers, as suggested in (4.8).

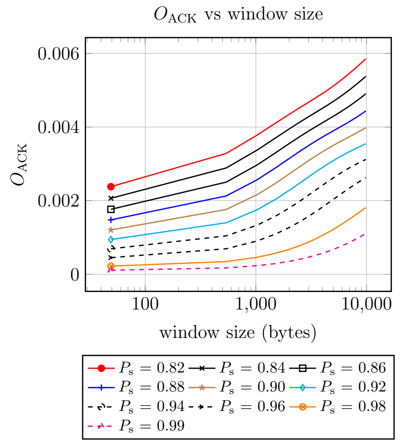
The L2 header size of Sigfox according to [20] is 21 bytes in downlink with a payload of 8 bytes. The L2 headers of LoRaWAN according to [3] are 13 byte long, hence we will especially focus on



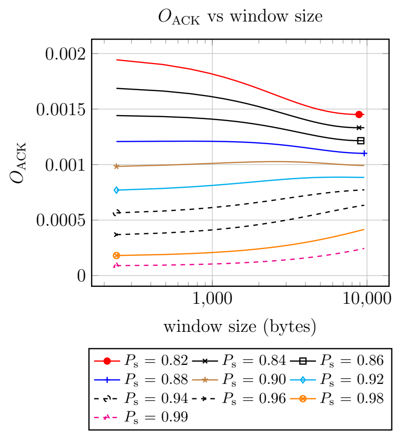
(a) $F = 242$, $t = 49$



(b) $F = 242$, $t = 240$



(c) $F = 242$, $t = 49$



(d) $F = 242$, $t = 240$

Figure 4.5: Impact of the window size w (in tiles) or wt (in bytes) on the ACK overhead O_{ACK}. The optimal window size (w_{opt}) is indicated in (c) and (d) with a mark. $L_{\mathrm{SH}} = 2$.

those values.

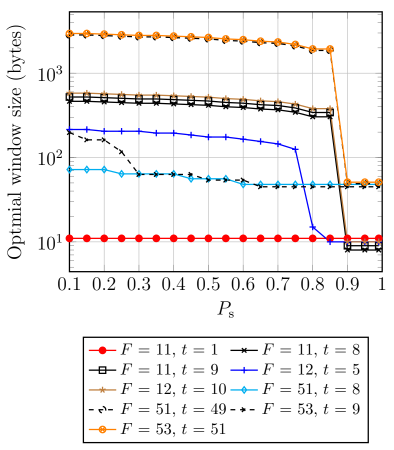
For the sake of clarity of the plots, we define P_{f} as $1 - P_{\mathrm{s}}$. Figs. 4.7a and 4.7b illustrate $O_{\mathrm{ACK_{L2}}}$ for $F = 11$, $t = 9$ and for $F = 242$, $t = 49$, with $L_{\mathrm{L2H}} = 13$, respectively.

Smaller windows outperform larger ones for $P_{\mathrm{s}} > 0.99$ (see the zoom in Figs. 4.7a and 4.7b), whereas in Section 4.4.2 it is for $P_{\mathrm{s}} > 0.90$. This happens because the additional constant length added to the ACK size favors the ACK pooling effect of large windows over the additional ACK length, making it more efficient to send one large ACK than several smaller ones (and their large L2 headers). Hence, only with very large success probability $P_{\mathrm{s}} > 0.99$, *i.e.*, $P_{\mathrm{f}} < 10^{-2}$, does the ACK length effect take over the ACK pooling effect, as Fig. 4.7a shows. For $P_{\mathrm{s}} > 0.99$, the best window size is 1 tile ($w = 1$), but as P_{s} decreases, the optimal window size increases (*e.g.*, $w = 143$ for $P_{\mathrm{s}} < 0.99$).

Figs. 4.7c and 4.7d confirm our conclusions, by showing $O_{\mathrm{ACK_{L2}}}$ for $F = 11$, $t = 9$ and for $F = 242$, $t = 49$ with $L_{\mathrm{L2H}} = 13$ for different P_{s}, respectively.

4.4.4 Percentage of used bits per fragment

The percentage of used bits per fragment (P_U) provides an overview of how efficient a tile size is for a given SCHC Fragment size. Fig. 4.8a illustrates P_U for $F = 11, 12, 51, 53$ and different tile


(a) $F = 11, 12, 51, 53$

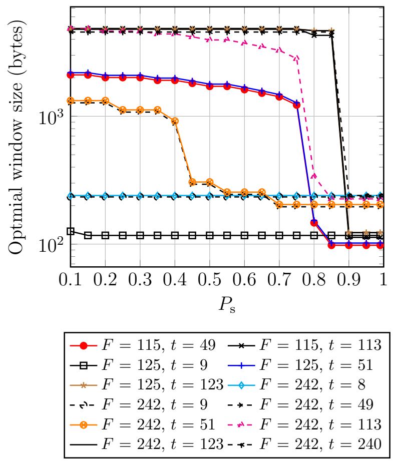
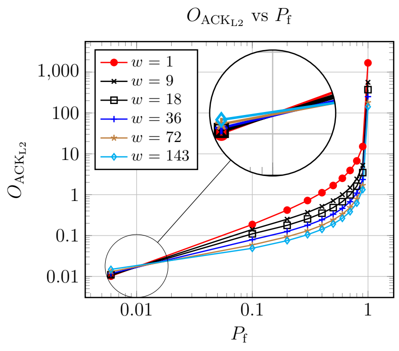
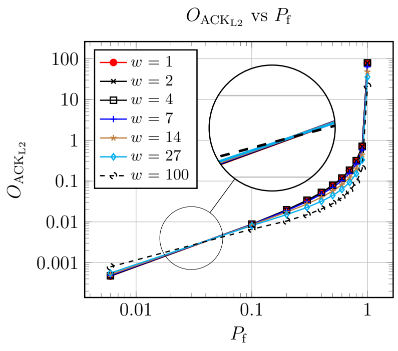
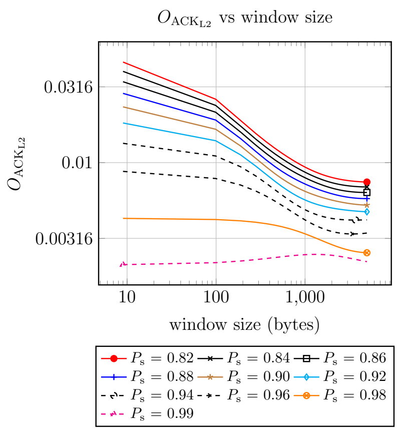

(b) $F = 115, 125, 242$

Figure 4.6: Optimal window size vs P_s for different settings.


(a) $F = 11$, $t = 9$


(b) $F = 242$, $t = 49$


(c) $F = 11$, $t = 9$

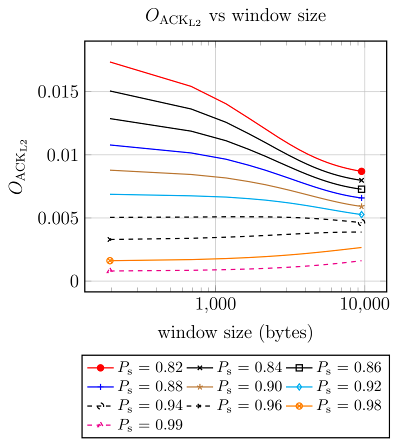
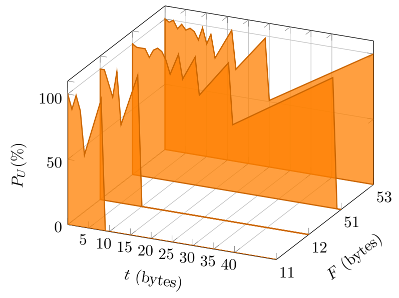

(d) $F = 242$, $t = 49$

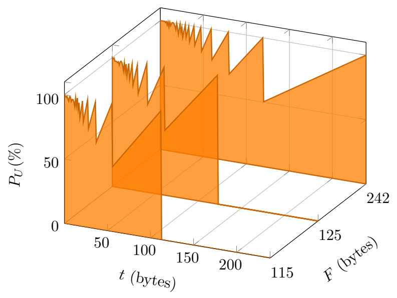
Figure 4.7: Impact of the window size w (in tiles) or wt (in bytes) on the ACK overhead $O_{\text{ACK}_{\text{L2}}}$. The optimal window size (w_{opt}) is indicated in (c) and (d) with a mark. $L_{\text{SH}} = 2$ and $L_{\text{L2H}} = 13$.

sizes. When fragment payload sizes are a multiple of the tile size, P_U is 100%, *e.g.*, $t = 1 \ \forall \ F$, $t = 3$ for $F = 11$, $t = 5$ for $F = 12$, $t = 7$ for $F = 51$ and $t = 17$ $F = 53$. Furthermore, there are tile sizes that have a low P_U. For example, $t = 5$ for $F = 11$ only uses the 55.56%, $t = 6$ for $F = 12$ with only 60%, $t = 25$ for $F = 51$ with 51.02% and $t = 26$ for $F = 53$ with 50.98%.

Fig. 4.8b shows the percentage of used bits for $F = 115, 125$ and 242. As with small SCHC Fragments, some tile sizes have a better P_U than others. Moreover, as the tile size gets larger and only one tile can be fitted in the SCHC Fragment payload, the percentage of used bits is reduced significantly. For example, $t = 57$ for $F = 115$ with 50.44%, $t = 62$ for $F = 125$ with 50.41% and $t = 121$ for $F = 242$ with 50.42%.



(a) $F = 11, 12, 51, 53$



(b) $F = 115, 125, 242$

Figure 4.8: Percentage of usage P_U vs t. $L_{\text{SH}} = 2$.

4.5 Technology-oriented evaluation

The SCHC framework specification does not offer recommendations about how the ACK-on-Error parameter values should be selected. This section provides configuration guidance for, and discusses the impact of, the main parameters of ACK-on-Error mode when used over LoRaWAN (EU868 and US915) and Sigfox.

This section is organized as follows: Section 4.5.1 describes the physical layer parameters of LoRaWAN and Sigfox relevant to the ACK-on-Error mode configuration. Section 4.5.2 provides our recommended values for the main ACK-on-Error parameters, along with the methodology used to determine such values. Section 4.5.3 discusses the optimal window size results obtained. Finally, Section 4.5.4 evaluates the global impact of limiting the ratio of LPWAN devices that use the optimal ACK-on-Error settings derived.

4.5.1 LoRaWAN and Sigfox relevant parameters

LoRaWAN defines a number of physical layer options for use in different world regions. In each region, a physical layer option corresponds to different settings of a parameter called the SF. Each SF defines a specific DR. In LoRaWAN, the L2 MTU depends on the SF/DR in use. All SF/DR settings in LoRaWAN are shown in the leftmost columns of Tables 4.1 and 4.2, for the EU868 and US915 bands, respectively.

In contrast, Sigfox physical layer options are fixed, with only one constant L2 MTU (see Table 4.3).

4.5.2 Determining SCHC ACK-on-Error mode parameter settings

This section provides our recommended values for the main parameter settings for ACK-on-Error mode, and explains how they are obtained, for each considered technology, and as a function of Ps. Tables 4.1 to 4.3 and Fig. 4.9 present the ACK-on-Error mode optimal parameter settings for LoRaWAN in EU868 and US915 bands, and for Sigfox, respectively. The main parameters comprise the fragment size (Section 4.5.2.1), the tile size (Section 4.5.2.2), the window size Section 4.5.2.3), the FCN field size (N) (Section 4.5.2.4), and the length of the Window field (M) (Section 4.5.2.5). Furthermore, reference values such as the maximum window size and the maximum bitmap size are also provided (Section 4.5.2.6).

4.5.2.1 Fragment size (F)

In order to maximize efficiency, the value of F needs to be equal to the current L2 MTU of the underlying LPWAN technology. Tables 4.1 to 4.3 show the optimal values of F for LoRaWAN (Tables 4.1 and 4.2), and for Sigfox (Table 4.3). For LoRaWAN, the value of column F depends on the current SF/DR settings (Tables 4.1 and 4.2). For Sigfox, F is a constant value (Table 4.3).

Table 4.1: ACK-on-Error configuration parameters for w_{opt} in Table 4.1 for LoRaWAN EU868

SF	**DR**	**F (bytes)**	t **(bytes)**	P_s	**N (bits)**	**M (bits)**	
						SCHC Packet size	
						320 (bytes)	**1280 (bytes)**
12, 11, 10	DR0, DR1, DR2	51	49	$0 < P_s \leq 0.98$	8	1	1
				$0.98 < P_s \leq 1$	3		2
9	DR3	115	49	$0 < P_s \leq 0.80$	9	1	1
				$0.80 < P_s \leq 0.90$	8		
				$0.90 < P_s \leq 0.97$	10		
				$0.97 < P_s \leq 1$	3		2
			113	$0 < P_s \leq 0.95$	9	1	1
				$0.95 \leq P_s < 0.98$	8		
				$0.98 \leq P_s < 0.99$	9		
				$0.99 \leq P_s < 1$	3		
8, 7	DR4, DR5	242	49	$0 < P_s \leq 0.30$	9	1	1
				$0.30 < P_s \leq 0.95$	8		
				$0.95 < P_s \leq 1$	3		2
			113	$0 < P_s \leq 0.80$	9	1	1
				$0.80 < P_s \leq 0.96$	8		
				$0.96 \leq P_s < 0.97$	9		
				$0.97 \leq P_s < 1$	3		
			240	$0 \leq P_s \leq 0.90$	9	1	1
				$0.90 < P_s < 0.97$	8		
				$0.97 \leq P_s < 0.98$	9		
				$0.99 \leq P_s < 1$	3		

$L_{\text{SH}} = 2$.

4.5.2.2 Optimal tile size

Column t of Tables 4.1 to 4.3 shows the most relevant tile sizes for each value of F over LoRaWAN and Sigfox. For each value of F, the greatest tile sizes in column t are the optimal sizes obtained in Section 4.4.2. Recall that the optimal tile size is the one that entirely fills the SCHC Fragment payload. On the other hand, because in LoRaWAN the value of F is variable, and SCHC F/R will still be possible when F changes, the optimal tile sizes for all possible smaller values of F are

Table 4.2: ACK-on-Error configuration parameters for w_{opt} in Table 4.2 for LoRaWAN US915

SF	**DR**	**F** (bytes)	t (bytes)	P_{s}	**N** (bits)	**M** (bits) SCHC Packet size **320** (bytes)	**M** (bits) SCHC Packet size **1280** (bytes)
10	DR0	11	9	$0 < P_{\text{s}} < 0.98$	6	1	2
				$0.98 \leq P_{\text{s}} \leq 1$	4	3	5
9	DR1	53	9	$0 < P_{\text{s}} \leq 1$	8	1	1
			51	$0 < P_{\text{s}} \leq 0.98$	8	1	1
				$0.98 < P_{\text{s}} \leq 1$	3		3
8	DR2	125	9	$0 < P_{\text{s}} \leq 0.60$	8	1	1
				$0.60 < P_{\text{s}} \leq 0.80$	7		
				$0.80 < P_{\text{s}} \leq 1$	4	3	4
			51	$0 < P_{\text{s}} \leq 0.80$	9	1	1
				$0.80 < P_{\text{s}} \leq 0.96$	8		
				$0.96 < P_{\text{s}} \leq 0.97$	9		
				$0.97 < P_{\text{s}} \leq 1$	3		2
			123	$0 < P_{\text{s}} \leq 0.90$	9	1	1
				$0.90 < P_{\text{s}} \leq 0.97$	8		
				$0.97 < P_{\text{s}} \leq 0.98$	9		
				$0.98 < P_{\text{s}} \leq 1$	3		
7, 8	DR3, DR4	242	9	$0 < P_{\text{s}} \leq 0.50$	7	1	1
				$0.50 < P_{\text{s}} \leq 0.60$	5		2
				$0.60 < P_{\text{s}} \leq 1$			3
			51	$0 < P_{\text{s}} \leq 0.30$	9	1	1
				$0.30 \leq P_{\text{s}} \leq 0.95$	8		
				$0.95 \leq P_{\text{s}} \leq 1$	3		2
			123	$0 < P_{\text{s}} \leq 0.90$	9	1	1
				$0.90 < P_{\text{s}} \leq 0.97$	8		
				$0.97 < P_{\text{s}} \leq 0.98$	9		
				$0.98 < P_{\text{s}} \leq 1$	3		
			240	$0 \leq P_{\text{s}} \leq 0.90$	9	1	1
				$0.90 < P_{\text{s}} < 0.97$	8		
				$0.97 \leq P_{\text{s}} < 0.98$	9		
				$0.99 \leq P_{\text{s}} < 1$	3		

$L_{\text{SH}} = 2$.

also evaluated. For example, the 49-byte tile size is optimal for $F = 51$ and it is also evaluated for $F = 115$ and $F = 242$, for LoRaWAN EU868.

4.5.2.3 Optimal window size

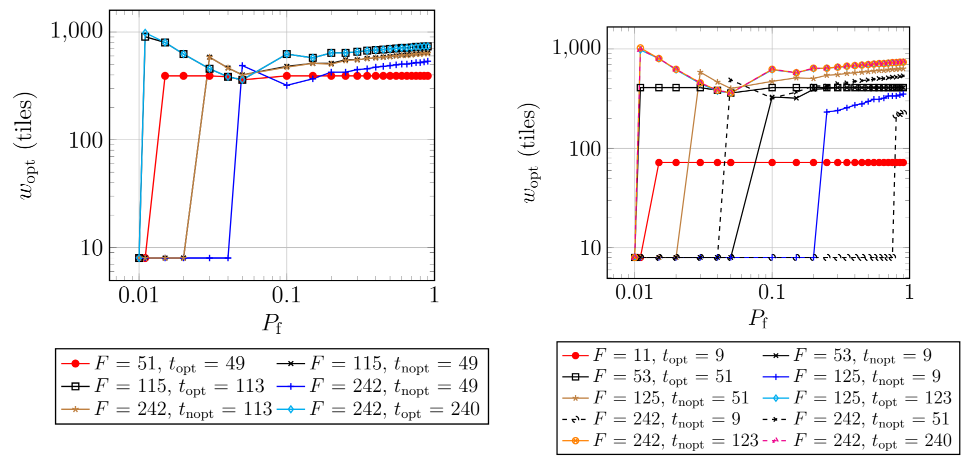
The optimal window size (w_{opt}) presented in Fig. 4.9 and in column w_{opt} of Table 4.3 is obtained by evaluating (4.8) from the minimum window size that generates an ACK size without padding (*i.e.*, $w = 8$) to the maximum window size in tiles derived from (4.11) (*i.e.*, maximum number of

Table 4.3: ACK-on-Error configuration parameters for w_{opt} for Sigfox

F (bytes)	t (bytes)	P_{s}	w_{opt} (tiles)	**N** (bits)	**M** (bits) SCHC Packet Size **320** (bytes)	**M** (bits) SCHC Packet Size **1280** (bytes)
12	10	$0 \leq P_{\text{s}} \leq 0.95$	48	6	1	3
		$0.95 < P_{\text{s}} \leq 1$	8	4	3	5

$L_{\text{SH}} = 2$.

bits in the bitmap). Then, the window size that yields the minimum $O_{\mathrm{ACK_{L2}}}$ is selected. Finally, as the ACK size has to be a byte multiple, the optimal window size value is the next window size that will generate a bitmap that will not require padding (*i.e.*, the next byte multiple). Fig. 4.9 presents the values of w_{opt} for LoRaWAN EU868 and LoRaWAN US915, evaluated for the corresponding values of F and t presented in Tables 4.1 and 4.2, respectively. Table 4.3 presents the optimal window size for Sigfox. For the sake of clarity, Fig. 4.9 and subsequent figures are shown as a function of $P_{\mathrm{f}} = 1 - P_{\mathrm{s}}$, instead of P_{s}.



(a) F for LoRaWAN EU868.

(b) F for LoRaWAN US915.

Figure 4.9: Optimal window size (w_{opt}) vs P_{f}.

4.5.2.4 FCN field size (N)

The optimal value of N is the minimum value that allows to unambiguously identify all tiles per window for a selected w, which is given by:

$$N = \lceil \log_2(w) \rceil \,. \tag{4.13}$$

In Tables 4.1 and 4.2, the value of column N is determined for the optimal window size (w_{opt}) presented in Figs. 4.9a and 4.9b, respectively. In Table 4.3, the value of column N is obtained for the values presented in the w_{opt} column.

4.5.2.5 Window field length (M)

The value of M must be set to identify the number of windows required to carry a SCHC Packet size (L_{SCHC}) and, it can be derived from:

$$M = \begin{cases} 1, & \text{if } L_{\mathrm{SCHC}} \leq w \cdot t \\ \left\lceil \log_2\left(\frac{L_{\mathrm{SCHC}}}{w \cdot t}\right) \right\rceil, & \text{if } L_{\mathrm{SCHC}} > w \cdot t. \end{cases} \tag{4.14}$$

The values of columns M in Tables 4.1 and 4.2 are obtained for the w_{opt} values presented in Figs. 4.9a and 4.9b, respectively. In Table 4.3, the value of column M is determined for the values shown in the w_{opt} column. In Tables 4.1 to 4.3, we considered two SCHC Packet sizes: 320 bytes and 1280 bytes. The first one is a relatively small size that will require fragmentation for all values of F evaluated. The second one is a larger size that corresponds to the minimum packet size that must be supported by the layer below IPv6 [2].

4.5.2.6 Maximum window and bitmap sizes

The SCHC framework does not define a fragmentation method for SCHC ACKs. Therefore, the L2 MTU of each technology limits the corresponding Maximum Bitmap Size (MBS) and Maximum Window Size (MWS) as explained in Section 4.3.5. Table 4.4 shows the MBS and MWS for LoRaWAN (EU868 and US912) and Sigfox with $L_{\mathrm{SH}} = 2$, and $U = 4$. Different tile sizes are presented considering different ACK-on-Error parameters configurations. Note that MBS does not depend on the tile size, but on the SCHC Fragment and SCHC Header sizes as shown in (4.11). MWS depends on MBS, the tile size and the RCS size (U), as shown in (4.12). Sigfox has a downlink payload size of 8 bytes [20], hence with $L_{\mathrm{SH}} = 2$, at most 6 bytes can be used for the bitmap.

Table 4.4: Maximum bitmap and window sizes for LoRaWAN and Sigfox

Region & Technology	F **(bytes)**	t **(bytes)**	**MBS (bits)**	**MWS (bytes)**	**N (bits)**
LoRaWAN EU868	51	49	392	19204	9
	115	49	904	44292	10
		113		102148	
	242	49	1920	94076	11
		113		216956	
		240		460796	
LoRaWAN US915	11	9	72	644	7
	53	9	408	3668	9
		51		20804	
	125	9	984	8852	10
		51		50180	
		123		121028	
	242	9	1920	17276	11
		51		97916	
		123		236156	
		240		460796	
Sigfox	12	10	48	476	6

$L_{\mathrm{SH}} = 2$, $U = 4$.

4.5.3 Optimal window size results: Discussion

In this subsection, we discuss the obtained optimal window size results for LoRaWAN and Sigfox, in terms of P_{f}, shown in Fig. 4.9 and Table 4.3, respectively.

As shown in Fig. 4.9, for high P_{f}, the optimal window size is large, due to the benefit of ACK pooling. As P_{f} decreases, ACK pooling becomes less advantageous. Therefore, it is better to configure smaller window sizes, which leads to smaller ACK sizes.

Another observation from Fig. 4.9 is that the optimal window size fluctuates for larger values of F, as function of P_{f}. In general, the $O_{\mathrm{ACK_{L2}}}$ curve as a function of P_{f} has a "U" shape (see Figs. 4.4, 4.5 and 4.7). This happens because for small w, a great number of small ACKs are generated, whereas for large w, few large ACKs are generated. Therefore, a minimum value of $O_{\mathrm{ACK_{L2}}}$ can generally be found between the two ends. However, the value of P_{f} affects the "U" shape. For very low P_{f}, there are very few ACKs. In such a case, small w produces short ACKs, producing a decay of the left side of the "U" shape of the curve as P_{f} decreases. Moreover, as P_{f} decreases, the value of $O_{\mathrm{ACK_{L2}}}$ will tend to decrease. On the other hand, for large w, E_k grows slowly with w (see Fig. 4.2b). The combination of all these effects produces changes in the "U" shape in such a way that small changes in P_f generate fluctuations in w_opt as a function of P_f. This behavior can be seen in Figs. 4.4 and 4.5 (except in Fig. 4.5c, where the considered tile size is non-optimal).

As shown in Fig. 4.9 and Table 4.3, this fluctuation is neither present for LoRaWAN ($F = 11$, $F = 51$) nor for Sigfox ($F = 12$). The reason is that the MBS (see Table 4.4) is not large enough to allow w_opt fluctuations to happen in the practical range given for those technology settings.

Note that w_opt fluctuation increases with the L2 header size (L_L2H). While in Fig. 4.6 (where $L_\mathrm{L2H} = 0$) there is little to no fluctuation of the optimal window size, in Fig. 4.9 the fluctuation is more pronounced. Indeed, the L2 header size contributes to the $O_{\mathrm{ACK}_\mathrm{L2}}$ metric as shown in (4.8).

As a practical remark, since the optimal window size depends on P_f, an enhanced implementation of SCHC can estimate P_f and select the optimal window size accordingly. When using SCHC over LoRaWAN, the current value of F needs to also be taken into account. Finally, note that the ACK-on-Error parameter optimizations proposed in this chapter do not require modifications to off-the-shelf radio platforms.

4.5.4 Impact of optimal ACK-on-Error parameter settings on downlink traffic

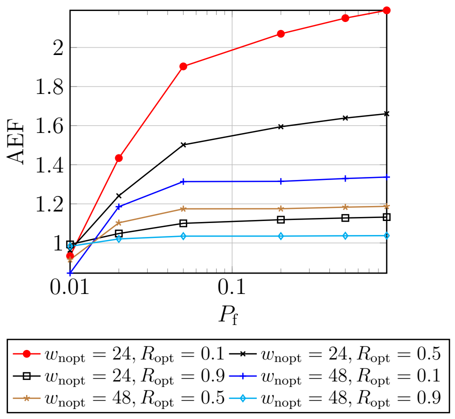
We next illustrate the impact of the derived optimal ACK-on-Error parameters values (shown in Tables 4.1 to 4.3 and Fig. 4.9) on the ACK-on-Error ACK traffic. To this end, we assume a network where LPWAN devices use ACK-on-Error mode to transmit SCHC Packets of the same size in the uplink, thus ACKs are sent in the downlink. We assume that only a fraction, R_opt, of these devices are optimally configured. We define and evaluate two performance metrics: the ACK Excess Factor (AEF) and the ACK Bytes Excess Factor (BEF) for a given SCHC Packet size (L_SCHC). The AEF is the number of ACKs actually transmitted (for $R_\mathrm{opt} \leq 1$) divided by the number of ACKs sent when all devices are optimally configured ($R_\mathrm{opt} = 1$). Analogously, the BEF is the number of ACK bytes actually transmitted divided by the number of ACK bytes sent when $R_\mathrm{opt} = 1$. Both metrics are evaluated considering the optimal window size (w_opt) and a non-optimal window size (w_nopt), for different values of P_f. Further, a comparison between an optimal tile size (t_opt) and a non-optimal tile size (t_nopt) is performed for the same w_opt.

Note that the global impact of the proposed optimized settings on the total downlink traffic in an LPWAN will depend also on other parameters that are independent of SCHC ACK-on-Error mode. These include the number of devices that do not use SCHC, how often such devices request downlink messages, how many unsolicited downlink messages (*e.g.*, management commands) are sent, etc.

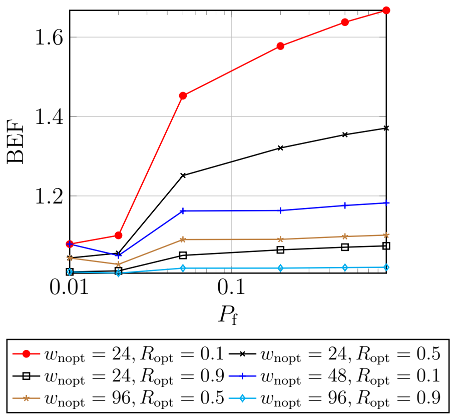
Figs. 4.10a and 4.10b show the AEF and BEF obtained for LoRaWAN US 915, SF10/DR0 ($F = 11$, see Table 4.2), respectively, for $L_\mathrm{SCHC} = 1280$ and different R_opt values, as a function of P_f. The optimal window values are retrieved from Fig. 4.9b, that is, for $P_\mathrm{f} \leq 0.011$, $w_\mathrm{opt} = 8$, and for $P_\mathrm{f} > 0.011$, $w_\mathrm{opt} = 72$. The non-optimal window sizes (w_nopt) evaluated are 24 and 48 tiles. The tile size used is the optimal one ($t_\mathrm{opt} = 9$).

As shown in Fig. 4.10, both AEF and BEF increases with P_f. If all nodes use optimal settings, the number of ACKs and the amount of ACK bytes decrease by up to factors greater than 2 and 1.6, respectively, compared to the scenario where $R_\mathrm{opt} = 0.1$ and $w_\mathrm{nopt} = 24$. As R_opt increases, the potential for improvement decreases (e.g. for $R_\mathrm{opt} = 0.9$ and $w_\mathrm{nopt} = 24$, the highest AEF and BEF values are 1.1 and 1.07, respectively). Note that ACK traffic improvement decreases as w_nopt approaches the w_opt value.

Another remarkable result shown in Fig. 4.10 is that, for $P_\mathrm{f} \leq 0.01$, the number of ACKs sent when optimal settings are used increases, as the AEF is lower than 1. However, there is a reduction in the excess of ACK bytes sent (*i.e.*, BEF is greater than 1), leading to benefits in channel occupancy. In this range of P_f values, all window sizes considered generate a similar number of ACKs for the same SCHC Packet size, but w_opt benefits from using the smallest ACK



(a) AEF vs P_f

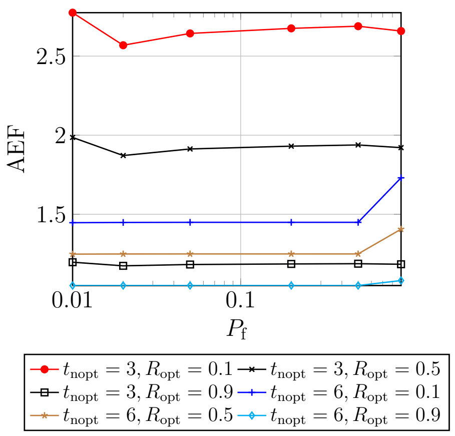


(b) BEF vs P_f

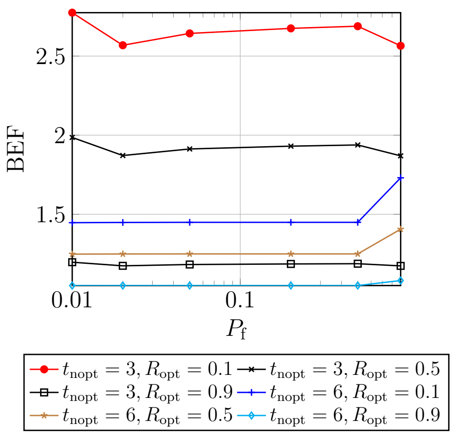
Figure 4.10: Impact of w_{nopt} on the AEF (a) and on the BEF (b) for different R_{opt} of devices using w_{opt}. Results for LoRaWAN US915, SF10/DR0, $F = 11$, $L_{\text{SCHC}} = 1280$, and $t_{\text{opt}} = 9$.

size.

Figs. 4.11a and 4.11b show the AEF and BEF derived for LoRaWAN US915, SF10/DR0 ($F = 11$), $L_{\text{SCHC}} = 1280$ when using w_{opt} with an optimal tile size ($t_{\text{opt}} = 9$), compared to using a non-optimal tile size (t_{nopt}), as a function of P_f. Both the number of ACKs and the amount of ACK bytes may decrease by a factor of up to 2.7, depending on R_{opt} and t_{nopt}. As R_{opt} increases, and as t_{nopt} approaches t_{opt}, the potential performance improvement decreases, since more devices use a t configuration closer to the optimal one.



(a) AEF vs P_f



(b) BEF vs P_f

Figure 4.11: Impact of t_{nopt} on the AEF (a) and on the BEF (b) for different R_{opt} of devices using t_{opt}. Results for LoRaWAN US915, $F = 11$, $L_{\text{SCHC}} = 1280$, $N_{\text{SCHC}} = 100$, and w_{opt}.

In general, when too many fragments are lost, it is better to send large ACKs, since in that case an ACK can be amortized to report many losses. Otherwise, using small ACKs is preferable. Recall that the ACK size is related to the number of tiles in a window. Assuming that fragmentation is performed in the uplink, the downlink channel usage by SCHC ACK-on-Error can be optimized. This means that ACK traffic per data traffic can be minimized. This translates into lower billing and longer LPWAN device battery lifetime. From the LPWAN gateway perspective, it reduces its load, which is critical in *duty-cycle* restricted networks, since a single gateway may serve a large number of LPWAN devices.

CHAPTER 5
ALTERNATIVE RECEIVER-FEEDBACK TECHNIQUES FOR SCHC

<<*Light makes visible what gets in the way.*>>

This chapter presents an evaluation of alternative Receiver-Feedback Techniques (RFTs) for fragmentation over LPWANs. To minimize and optimize the ACK size, we evaluate the RFT of SCHC, i.e., Compressed Bitmap (CB), and compare its performance with 5 alternative RFTs that we proposed. These new RFTs include new ways of encoding the receiver-feedback. All RFTs were evaluated regarding the ACK size, number of L2 frames required, and ToA, using an uncompressed bitmap as benchmark RFT. Results were obtained for all LoRaWAN world regions. Tests were performed using Sim-RFT, a channel error simulator that we developed. Sim-RFT provides different error rates and patterns. The RFTs description, performance evaluation, and guidance in selecting the optimal RFT have been published in a journal article [II].

5.1 Introduction

The main product of the IETF LPWAN WG is the specification of an adaptation layer framework, called SCHC [6,55]. The need for this solution is justified by the fact that prior efforts to support IPv6 over low-power wireless technologies, such as 6LoWPAN or 6Lo, yield a too high overhead in the light of the severe constraints of LPWAN technologies [1]. In order to overcome this issue, SCHC defines ultra-lightweight header compression, as well as LPWAN-tailored fragmentation mechanisms. This chapter focuses on aspects of the latter.

In order to adapt to the potentially diverse requirements of different LPWAN technologies and deployments, SCHC offers different fragmentation modes, namely: No-ACK, ACK-Always, and ACK-on-Error. The last two modes support ACKs and selective fragment retries. In both ACK-Always and ACK-on-Error, the receiver generates (upon fragment loss in the latter) a selective ACK after a group of fragments have been sent. That is, the ACK informs the sender about which fragments have been received or not from the considered group of fragments. The way in which such information is encoded is given by the RFT defined in SCHC, which is called CB. As per CB, the ACK payload carries a bitmap where the k-th bit of the bitmap indicates whether the k-th fragment has been received or not. In some cases, the bitmap may be compressed, which represents a performance optimization, compared with early versions of SCHC that made use of a simple Uncompressed Bitmap (UB). However, the performance of CB has not been evaluated, and, to the best of our knowledge, alternative RFTs have not been considered for SCHC.

In this chapter, we investigate the performance of CB, along with that of two alternative RFTs called List of Lost Fragments (LLF) and List of Deltas (LoD). LLF is a binary-encoded list of Fragment Numbers (FNs) that correspond to lost fragments. LoD is based on the differences (deltas) between the FNs of consecutive lost fragments. For efficiency, deltas are encoded with variable length formats using Self-Delimiting Numerical Values (SDNV) [58]. Regarding the latter, we investigate 4 different approaches which use base encoding format sizes of 2, 3, 4, and 5 bits. We also include UB as a benchmark in our study.

We evaluate the performance of the considered RFTs by means of extensive simulation. To this end, we developed Sim-RFT, an ad-hoc simulator that allows to analyze RFT performance for different fragmented packet sizes, and under different error rates and patterns. The main performance parameters evaluated are the ACK payload size, the number of L2 frames required to carry an ACK, and the ACK ToA. Our results show that CB only outperforms the alternative RFTs considered for short fragmented packet sizes or under high error rates.

The remainder of the chapter is organized as follows. In Section 5.2, we review existing work related to SCHC and fragmentation over LPWANs. We describe the RFTs considered in this chapter in Section 5.4. In Section 5.5, we present Sim-RFT, along with the configuration settings and error patterns used in the study. In Section 5.7, we evaluate the performance of the RFTs under a range of conditions, and discuss the obtained results.

5.2 Related work

In this section, we review research work that focused on SCHC and fragmentation over LPWANs, with a particular perspective on RFTs, where applicable. Some of them investigated both SCHC compression and fragmentation mechanisms [46–48, 55–57, 59, 60], while others focused only on fragmentation functionality [16, 61].

The authors in [55] provided an overview of SCHC. As a future work item, they proposed a reliable fragment delivery mechanism whereby a single ACK would report on the delivery success or failure of all the fragments that carry a large packet. When the number of fragments per packet is too high, it may be challenging to fit the receiver report in only one L2 frame. Therefore, the authors pointed out the need to consider alternative RFTs, instead of CB, specified in SCHC, which, under some conditions, may produce a too large ACK payload. However, the authors neither described nor evaluated alternative RFTs.

Suciu et al. evaluated the efficiency of fragmentation in dense LPWAN networks [16]. However, the authors did not consider receiver-feedback mechanisms in their study. Another work defined and evaluated the effect of an aggressive fragmentation strategy for LPWANs, *i.e.*, performing fragmentation even if the packet to be carried fits the L2 frame [61]. The study used UB to report fragment reception status in negative ACKs (NACKs). However, the authors neither studied the impact of error patterns on the NACK size, nor considered different RFTs.

Other studies analyzed the performance of IPv6 header compression [47,48,56] and/or fragmentation over LPWAN by using SCHC [46,57,59,60]. Abdelfadeel et al. [48,56] and Ayoub et al. [47] focused only on SCHC header compression. Moons et al. [46] compared the memory footprint of SCHC header compression and fragmentation with that of a 6LoWPAN-based solution. For SCHC fragmentation, they used UB as RFT. The authors in [57] compared the different SCHC fragmentation methods, assuming an ideal communication channel, in terms of channel occupancy, goodput, and delay. A mathematical model to calculate the ACK message overhead of ACK-on-Error mode, and how to optimally tune its most critical parameters, is presented in [59]. However, UB was assumed in both [57] and [59]. The authors in [60] evaluated SCHC header compression and fragmentation when using an end-to-end CoAP broker to connect LoRaWAN devices by using a publish/subscribe scheme. However, the No-ACK fragmentation mode was used, therefore no RFT was studied in this work.

Based on our literature analysis, and to the best of our knowledge, previous work neither evaluates the performance of RFTs different from UB (not even CB, which is the one used in SCHC), nor the impact of different packet sizes, error rates, and error patterns on RFT performance.

5.3 Reliable fragmentation over LPWAN

In this chapter, we consider an efficient configuration of reliable fragmentation where a single ACK provides feedback on the delivery success or failure of the whole set of fragments that transport a packet. If the ACK size cannot fit one L2 frame, each additional L2 frame required includes an ACK header as well.

Fig. 5.1 presents an example of the transmission of a packet that requires 10 fragments to be carried. Two fragments (with FNs 1 and 6) are lost, whereas the other fragments are correctly received. At the receiver, once the last fragment (which is signaled by a dedicated FN value) is received, an ACK carrying a payload produced by the RFT in use is assembled and transmitted. In the following section, the example shown in Fig. 5.1 will be used to illustrate the behavior of each RFT studied in this chapter.

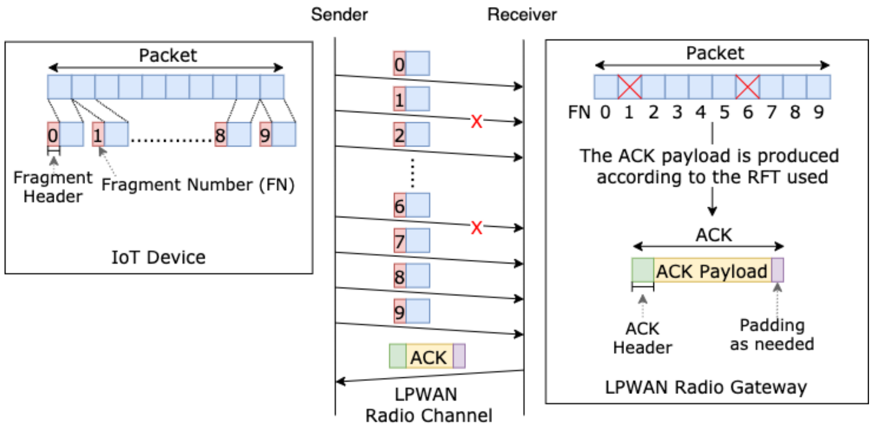


Figure 5.1: Example of the transmission of a 10-fragment packet with errors.

5.4 Receiver-Feedback Techniques for LPWAN Fragmentation

In this section, we describe the four RFTs considered in our performance evaluation, namely: UB, CB, LLF, and LoD.

UB was the initially considered RFT for SCHC. In this chapter, UB is used as a benchmark. CB, an improved version of UB, is the RFT standardized in SCHC. We introduce LLF and LoD as RFTs with potential to offer good performance in some scenarios. Each considered RFT follows a different approach and is thus expected to perform differently, depending on conditions such as error rates, error distribution, and packet size.

5.4.1 Uncompressed Bitmap (UB)

UB was introduced in early stages of the design of SCHC. This RFT is based on representing the sequence of received fragments by means of a sequence of bits, called a bitmap. Each bit in the bitmap corresponds to a fragment of the packet, where the k-th bit is set to 1 or 0 when the k-th fragment has been received or not, respectively. The leftmost bit of the bitmap corresponds to the first fragment of the packet.

Following the example presented in Fig. 5.1, when UB is used, the receiver generates the bitmap, sets to one the bitmap bits that correspond to successfully received fragments, and sets to zero the remaining bits (in the example, the ones corresponding to FNs 1 and 6). The resulting bitmap has a size of 10 bits, since the packet size is 10 fragments. Finally, the ACK is assembled by prepending the ACK header to the bitmap, as shown in Fig. 5.2. If needed, padding bits are appended at the end of the bitmap.

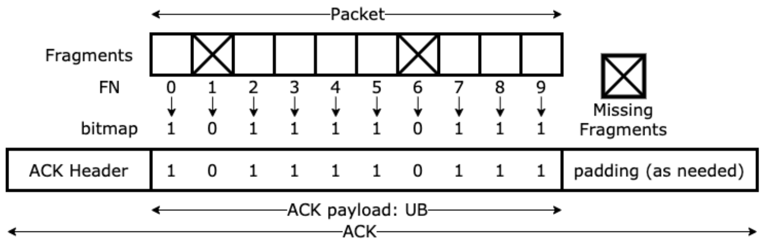


Figure 5.2: Example of ACK payload for UB, and ACK format, for the transmission of a 10-fragment packet where two fragments (with FNs 1 and 6) are lost.

5.4.2 Compressed Bitmap (CB)

CB is an RFT designed to reduce the size of the bitmap produced by UB, when possible. To this end, a receiver operates as follows. Firstly, a bitmap is built as described in Section 5.4.1. Then, in order to compress the bitmap, the receiver analyzes each bitmap bit from right to left. All contiguous bitmap bits set to 1 are removed. The receiver will stop this procedure when a 0 is found or when it reaches the leftmost bitmap bit. The result of this operation is a compressed version of the bitmap. After the ACK header is prepended to the compressed bitmap, the size of the latter may need to be adjusted, depending on the minimum data unit size supported by the underlying LPWAN technology. For example, if that technology is byte-oriented, the minimum number of bitmap bits with value 1 are restored on the right, so that the ACK header plus the ACK payload have a size multiple of an integer number of bytes. In some cases, padding may be needed as well. The sender can reconstruct the original bitmap from the (potentially) compressed bitmap received in the ACK message, as the sender knows the number of fragments sent to carry a given packet.

Following the example presented in Fig. 5.1, Fig. 5.3 illustrates how the size of the bitmap obtained in Fig. 5.2 is reduced from 10 bits (with values 1011110111) to 8 bits (*i.e.*, 10111101), assuming a 1-byte ACK header and a byte-oriented underlying LPWAN technology.

Note that the compression degree that can be achieved with CB depends on which is the last lost fragment carrying data from a fragmented packet. For example, an error in the transmission of the last fragment will not allow compressing the corresponding bitmap when using CB.

5.4.3 List of Lost Fragments (LLF)

We define LLF as an alternative RFT that produces the sequence of binary-encoded FNs of the lost fragments (if any) that carry a packet. In contrast with UB, which has a fixed length for a given packet size, LLF produces a variable length ACK payload, which is roughly proportional to the Frame Error Rates (FER).

Fig. 5.4 shows the ACK payload that corresponds to the example presented in Fig. 5.1, where the fragments with FNs 1 and 6 are lost, when LLF is used. The ACK payload comprises these two FN values, converted to binary code with a size of 7 bits per FN, thus leading to a 14-bit

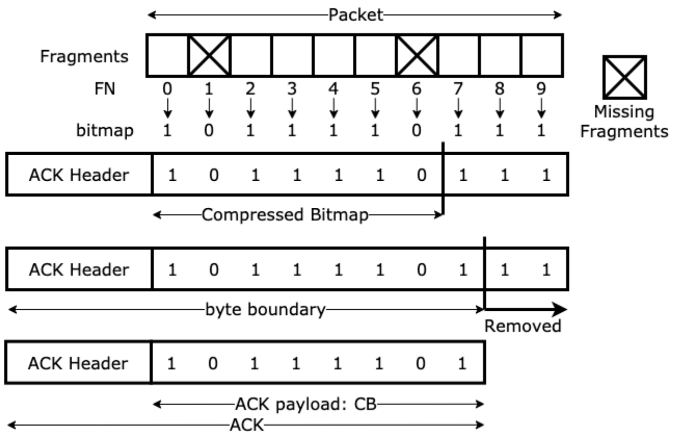


Figure 5.3: Example of a bitmap before compression and the corresponding compressed bitmap. The rightmost sequence of consecutive bitmap bits set to 1 is removed to obtain the compressed bitmap. The example assumes a byte-oriented underlying L2 LPWAN technology. In this example, padding is not needed.

ACK payload. Note that the 7-bit encoding per FN allows to identify fragments numbered in a range from 0 to 127. Considering a 10-byte fragment payload (which is typical in many LPWAN scenarios), the FN range will allow to unambiguously identify each fragment of a 1280-byte packet, thus allowing compliance with the IPv6 MTU requirement [2].

In general, an LLF implementation can be built with a simple concatenation of the FNs of missing fragments in a string. For example, in an MCU that supports C programming language, it can be done by using the “strcpy” function, which is part of the string.h standard library of C. This will represent a code size increase in the order of just a few bytes. Since SCHC includes the calculation of a RCS based on a CRC of 32 bits, which presents significantly more computational complexity than the considered RFTs, an LLF implementation would not add a significant amount of code footprint.

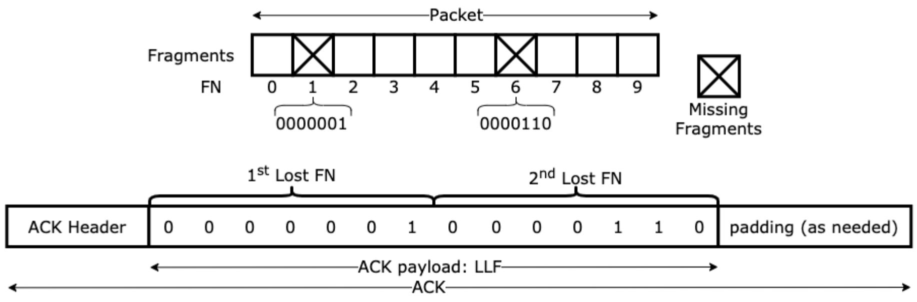


Figure 5.4: With LLF, the FNs of lost fragments are converted to binary and appended to the SCHC ACK Header. Finally, padding is added as needed.

5.4.4 List of Deltas (LoD)

We define LoD as an RFT where the receiver reports the differences (hereinafter, deltas) between the FNs of any two consecutive lost fragments. Instead of encoding absolute fragment numbers as in LLF, LoD exploits the smaller expected size of binary encoded deltas. In LoD encoding, the first lost fragment absolute FN is encoded as a reference. Subsequent encoded values are the deltas between consecutive lost fragments' FNs.

Following the example presented in Fig. 5.1, when LoD is used, the FN of the first lost fragment (*i.e.*, 1) is encoded as reference value. The next lost fragment corresponds to FN = 6, so a delta of 5 is then encoded. Note that the ACK payload produced by LoD is not only sensitive to the FER but also to the fragment loss distribution.

In order to optimize the LoD encoding, the number of bits to encode each delta needs to be variable, allowing to represent smaller deltas with a lower number of bits. To this end, we use SDNV encoding [58]. This technique allows a simple way of representing non-negative integers efficiently and with variable length. We next describe SDNV in detail.

SDNV represents a number by means of one or more elementary data units, which we refer to as bases. A base is a fixed-length set of bits used to fully or partially encode a number. The most significant bit of a base is a control bit reserved to determine whether that base is the last one for representing a number (in that case, the control bit is set to 0). The remaining bits in a base are data bits, *i.e.*, they are used to encode actual values. Therefore, in each base used, there is a 1-bit overhead. If a number cannot be encoded by using just one base, then additional bases are added as needed. In order to encode a number by using SDNV, the following steps are followed:

1. The number to be encoded is converted to binary.
2. The binary-encoded number bits are encapsulated, from left to right, in as many bases as needed, using the available data bits in each base. If needed, the base with the most significant binary number bit is padded with zeros on the left in order to fill in all data bits of that base.
3. The control bit of each base is appropriately set.

The SDNV standard uses a base of one byte thus, there are 7 data bits available in each base. For the sake of efficiency, in the evaluation carried out in this chapter, we consider several smaller base sizes. We use the notations SDNV-x and LoD-x to denote the usage of SDNV with a base size of x bits. Fig. 5.5 shows two examples of how two numbers in decimal (10 and 123) are encoded in SDNV-3 and SDNV-5, as per the steps provided above.

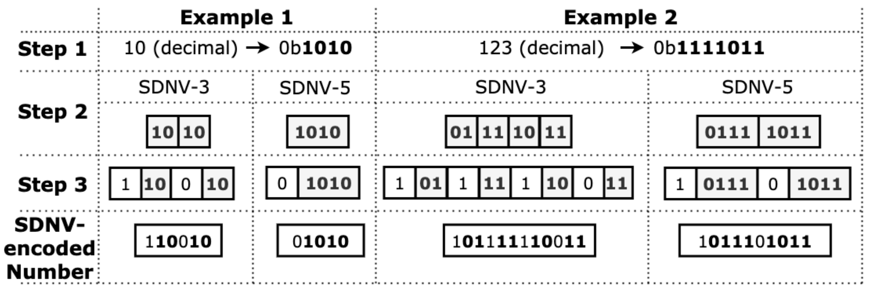


Figure 5.5: Example of different SDNV-encoded numbers. Data bits are represented in bold font.

Following the example presented in Fig. 5.1, where fragments 1 and 6 are lost, Fig. 5.6 illustrates the LoD encoding for different base sizes. In this example, LoD-4 and LoD-2 produce the shortest-sized ACK payload.

Note that SDNV encoding is based on a simple algorithm that requires around 12/13 lines of code for decoding/encoding operations, respectively, when using python and standard libraries. This will represent a code footprint increase in the order of tens of bytes, that depends on the embedded microcontroller, compiler and programming language used. Considering that SCHC

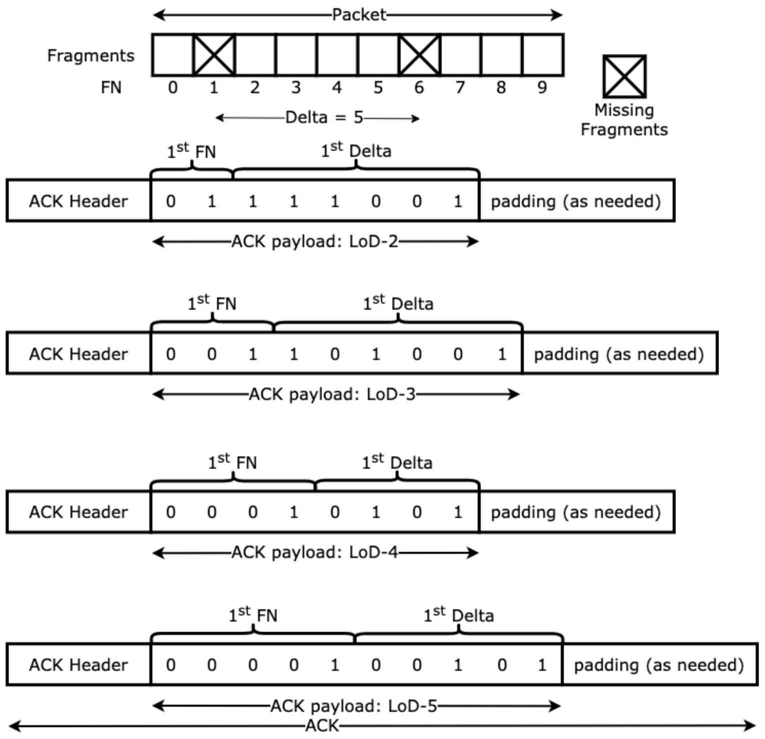


Figure 5.6: Examples of ACKs, using LoD, and for the fragment losses shown in Fig. 5.1 (*i.e.*, fragments with FN=1 and FN=6), for different SDNV bases. The decimal numbers to be encoded are 1 (first FN) and 5 (first delta). All considered SDNV base sizes require one base to encode the FN of the first lost fragment. To encode the delta, SDNV-4 and SDNV-5 only require 1 base, whereas SDNV-2 and SDNV-3 need 3 and 2 bases, respectively.

involves several operations (*e.g.*, including a 32-bit CRC for integrity checks, as mentioned earlier), LoD would not require a significant amount of additional program storage for an embedded microcontroller.

5.5 Simulation environment

In this section, we present Sim-RFT, the simulation environment that we use in this chapter to compare the performance of the different RFTs introduced in the previous section. We describe how Sim-RFT works and detail the main fragmentation-related parameters assumed in our evaluations, along with the characteristics of the error patterns considered.

5.5.1 Main features

Sim-RFT is an ad-hoc tool that we have developed to simulate fragmented packet transmission and reception over a lossy channel. After each simulated fragmented packet transmission, Sim-RFT creates the corresponding ACK, for each RFT.

Sim-RFT provides three main performance parameters from each obtained ACK: i) ACK size, ii) number of L2 frames required to carry the ACK, and iii) ToA for each ACK. These performance parameters take into account the main L2 frame characteristics of the underlying LPWAN technology considered: L2 frame header size, and L2 MTU.

Our study focuses on the fragmented packet first transmission attempt, as it will create the largest ACK payload size for all evaluated RFTs, except for UB, which always yields the same ACK payload size for all transmission attempts. As a side-contribution of this work, we offer Sim-RFT publicly [62].

5.5.2 Settings

In order to maximize the applicability of our evaluation results, in this chapter we configure Sim-RFT to use 3 different L2 MTU values: 11 bytes, 51 bytes and 242 bytes, for both the uplink and the downlink. Table 5.1 shows the LoRaWAN L2 MTU values for all LoRaWAN channel plans. The 11-byte L2 MTU corresponds to the maximum frame payload size of LoRaWAN US915 Data Rate 0 (DR0) and AU915 DR0. It is also similar to the 12-byte uplink, 8-byte downlink L2 MTU of Sigfox [4, 20], and the 19-byte L2 MTU of LoRaWAN AS923 DR0. The 51-byte L2 MTU corresponds to DR0 for the following LoRaWAN regional bands: EU868, CN779, EU433, CN470, KR920, IN865, and RU864. The 242-byte L2 MTU corresponds to the maximum one in LoRaWAN for the highest DR of all regions (except AS923), and it is similar to the 250-byte L2 MTU of AS923 with its maximum DR (see Table 5.1). Therefore, conclusions from the evaluation will be useful when considering fragmentation over LoRaWAN in all available regions or countries where it is defined, and also over Sigfox. Note that, in a LoRaWAN frame, only an integer number of bytes can be carried. Accordingly, Sim-RFT will add padding bits if required. Section 5.6 shows the LoRaWAN full set of L2 configuration parameters used in the simulations. Fragment header and ACK header sizes of 1 byte are assumed.

Table 5.1: LoRaWAN MTU Values

Channel Plan	Country/ Region	L2 MTU (bytes)	
		Minimum DR	Maximum DR
EU863-870	Europe Middle East Africa	51	242
US902-928	America	11	242
CN779-787	China	51	242
EU433	Europe Middle East Africa	51	242
AU915-928	Australia	11	242
CN470-510	China	51	242
AS923	Asia	19	250
KR920-923	South Korea	51	242
IN865-867	India	51	242
RU864-870	Russia	51	242

Sim-RFT does not support native L2 retransmission mechanisms (*e.g.*, LoRaWAN confirmed data messages [3]), as it is not required by SCHC (*e.g.*, when used over LoRaWAN [9]), and also because they can have a negative impact on uplink throughput [63] and cause considerable network performance degradation [64].

5.5.3 Error patterns and rates

In order to evaluate the performance of the considered RFTs in the presence of errors, two different error distributions are supported in Sim-RFT: a uniform error distribution [65–94], and a burst error distribution [71–87]. These error distributions cover a comprehensive set of characteristics of LoRaWAN networks such as frame loss over distance [68,70,71,74,78,88–90], different uses cases [68, 72,91–94], mobile or stationary devices [70,83,86,87], network capacity [91], and collisions [70,90]. Frame loss burstiness may be due to channel effects [71,73], mobility [71,72], limited coverage [75, 77,78,81], or opportunistic coverage [79–81]. Event-driven communication may also lead to burst errors, as many devices may try to communicate at the same time during a relatively long time interval [82–85]. Table 5.2 summarizes the types of error distributions identified in LoRaWAN literature.

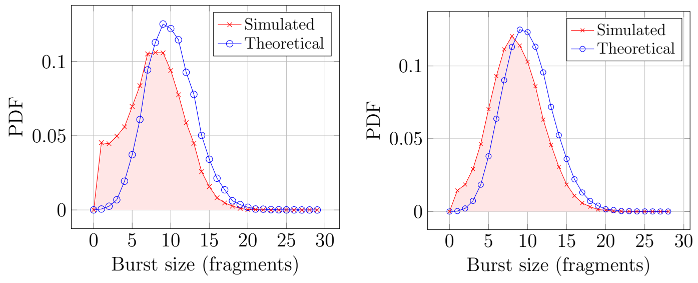
Table 5.2: LoRaWAN Error Distributions in Literature

References	Error distribution	
	Uniform	Burst
[65–70]	Yes	No
[71–94]	Yes	Yes

The uniform error distribution is modeled by a Bernoulli process, with a fragment error probability equal to the FER. The burst error distribution is modeled by a discrete Markov chain composed of two states [71]: a good state, where there are no fragment errors, and a burst state, where there is a burst of λ_L consecutive fragment errors. The probability of transition from a good state to the burst state is referred to as Burst Occurrence Probability (BOP). λ_L is modeled as a random variable that follows a Poisson distribution (see Fig. 5.7). Once in the burst state, several fragments are lost, and then the chain transitions back to the good state. Each fragmented packet transmission starts in the good state. We consider that transmitting different fragmented packets corresponds to independent events. Therefore, if a burst length λ_L is larger than the remaining number of fragments to be transmitted, the resulting burst length (λ_{RL}) will be smaller, hence the burst will be truncated ($\lambda_{RL} \leq \lambda_L$). As the fragmented packet size increases, the probability that a burst will be truncated decreases and larger burst lengths are more likely (see Figs. 5.7a and 5.7b).

In order to capture the characteristics of a wide range of LoRaWAN scenarios, as reported in prior work, in this chapter we consider FER values of 1%, 10% and 20%. $FER = 1\%$ corresponds to good channel conditions, with sporadic fragment errors [65, 68, 87]. A FER of up to 10% is expected in industrial deployments [66, 67], and it can also be found under certain LoRaWAN configurations for static devices [70] and mobile ones [76]. FER up to 20% was found in adverse environments [69], with link distance being the primary cause of losses [78, 87].

On the other hand, we consider BOP values of 1% and 2%, with an average burst size (λ) of 10 fragments (see Fig. 5.7), which captures burst error characteristics found in the literature. Burst error lengths between 2 and 30 frames have been reported, with the range between 2 and 7 frames corresponding to the most likely burst error length [71–75]. By modeling λ_L with a Poisson distribution, with an average burst size of 10 fragments, and considering burst truncation, the resulting burst length (λ_{RL}) distribution concentrates 40% of burst sizes between 2 and 7 fragments, with an actual average burst length of 7 fragments, while still providing burst sizes up to 30 fragments (see Fig. 5.7).



(a) Packet size: 22 fragments (b) Packet size: 80 fragments

Figure 5.7: Probability Density Function (PDF) of the theoretical vs simulated burst length ($\lambda = 10$).

5.6 LoRaWAN operation and settings

In this section, we describe the main characteristics and settings of LoRaWAN, emphasizing the ones that are most relevant to the evaluation carried out in this chapter. The section is divided into two parts, which focus on the LoRaWAN physical layer and on the LoRaWAN L2 layer, respectively.

5.6.1 LoRaWAN physical layer

LoRaWAN is based on LoRa as the physical layer. LoRa is a spread spectrum modulation scheme based on CSS technology [95].

In LoRa networks, the time duration of a frame transmission at a given SF and BW is called the ToA of a frame (ToA_{frame}). The available BW is of 125 kHz, 250 kHz or 500 kHz. The SF takes values from 7 to 12. The ToA_{frame} can be defined as the time required for the transmission of the preamble plus the payload of the LoRa frame [96], and can be obtained as follows:

$$ToA_{frame} = T_{preamble} + T_{payload}. \tag{5.1}$$

The preamble is a sequence of a programmable number of symbols for receiver synchronization. Its transmission time can be calculated as follows:

$$T_{preamble} = (n_{preamble} + 4.25) \cdot T_{sym}, \tag{5.2}$$

where $n_{preamble}$ is the aforementioned number of symbols. The symbol period (T_{sym}) depends on the channel BW and SF selected, and can be calculated as follows:

$$T_{sym} = \frac{2^{SF}}{BW}. \tag{5.3}$$

The transmission time of the payload can be calculated as:

$$T_{payload} = (payloadSymbNb) \cdot T_{sym}, \tag{5.4}$$

where *payloadSymbNb* is the number of symbols of the LoRa frame payload and header [96]. The *payloadSymbNb* can be calculated as follows:

$$payloadSymbNb = 8 + max\left(ceil\left(\frac{8PL-4SF+28+16-20H}{4(SF-2DE)}\right)(CR+4), 0\right), \tag{5.5}$$

where PL is the payload size in bytes, and other parameters (SF, H, DE and CR) can be found in Table 5.3.

Table 5.3 presents the LoRa and LoRaWAN configuration parameters used in the evaluation presented in Section 5.7.

Table 5.3: LoRa and LoRaWAN configuration parameters

LoRa Configuration Parameters			
Region/Country	US	EU	CN
Channel Plan	US902-928	EU863-870	CN779-787
Data Rate (DR)	0	0	5
Spreading Factor (SF)	10	12	7
Bandwidth (kHz)	125	125	125
Indicative physical bit rate (bit/sec)	980	250	5470
$n_{preamble}$		8 symbols	
Header enable (H)		Disable (0)	
Low Data Rate Optimization (DE)		Disable (0)	
Coding Rate (CR)		4/5	
CRC present		YES (1)	
LoRaWAN Configuration Parameters			
L2 control headers		13 bytes	
L2 MTU (N)	11 bytes	51 bytes	242 bytes

5.6.2 LoRaWAN L2 layer

LoRaWAN specification [3] defines a LoRaWAN L2 frame format. The LoRaWAN L2 frame is carried by the LoRa frame. The L2 LoRaWAN frame has a 13-byte control header. Table 5.3 presents the LoRaWAN configuration parameters used in the evaluation presented in Section 5.7.

To better understand the relation between the LoRaWAN L2 payload size and the ToA_{frame}, Fig. 5.8 shows the ToA_{frame} for the configurations of LoRa and LoRaWAN presented in Table 5.3, for different LoRaWAN L2 payload sizes. Note that ToA_{frame} shows a stepwise relationship with the LoRaWAN payload. This happens because of how LoRa physical layer determines the *payloadSymbNb* (see (5.5)).

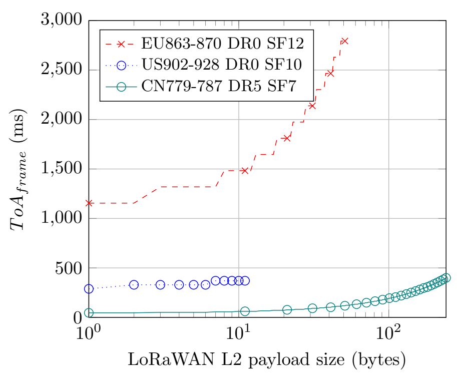


Figure 5.8: ToA_{frame} for different LoRaWAN L2 payload sizes obtained by using (5.1) and the parameters shown in Table 5.3.

5.6.2.1 LoRaWAN Device Class

LoRaWAN defines three classes of devices (Class A, Class B, and Class C), which are relevant to the energy consumption of a device and communication delay. However, LoRaWAN device class is not relevant for this chapter, since it is orthogonal to communication error characteristics.

5.7 Evaluation

In this section, we use Sim-RFT to investigate the performance of the RFTs presented in Section 5.4 for different FER values, different error distributions (uniform and burst), and for a range of packet sizes that require fragmentation, including the MTU required for IPv6, *i.e.*, 1280 bytes. Note that there exist applications that involve longer packet sizes. These include waveform captures, data logs, and large data packets using rich data types [97]. As introduced in Section 5.5.2, LoRaWAN is the underlying LPWAN technology. Regarding LoD, we consider base sizes of 2, 3, 4, and 5 bits.

The performance metrics are: i) ACK payload size, ii) number of L2 frames needed to carry an ACK, and iii) ACK ToA, hereinafter ToA, for each considered RFT. The ACK payload size is critical to LPWAN performance. On the one hand, the downlink channel of an LPWAN radio gateway is a bottleneck for the whole network. Note that most LPWAN traffic is sent in the uplink, part of that traffic requires downlink transmissions (*e.g.*, ACKs), and the number of LPWAN devices per radio gateway may be large. Furthermore, there exist spectrum access regulations that restrict the *duty-cycle* in some world regions and frequency bands (*e.g.*, LoRaWAN operates in Europe in the 868 MHz, which is limited to a maximum *duty-cycle* of 1%). Reducing the ACK size increases the number of IoT devices that can be supported per LPWAN radio gateway. On the other hand, the ACK size has a direct impact on the energy consumption of IoT devices. The number of L2 frames needed per ACK measures the ACK fragmentation overhead. If the ACK size exceeds the L2 frame maximum payload size, additional frame transmissions (including their corresponding L2 headers) are required, reducing efficiency. There may also be a negative impact on cost, as some operators charge by the number of downlink messages sent. Finally, ToA captures the channel occupancy over time due to ACK transmission, which is relevant to the scalability of the LPWAN. Each individual result provided has been obtained as the average over one million simulations.

5.7.1 ACK payload size analysis

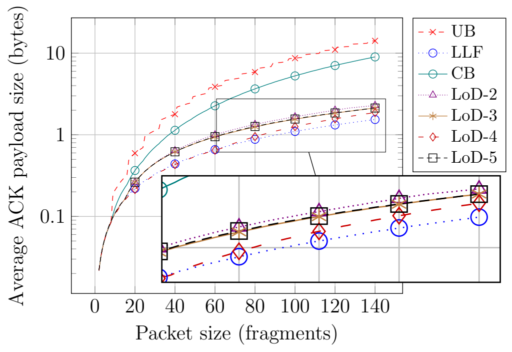
In this subsection, we evaluate the average ACK payload size for the different RFTs. In order to capture the full impact of ACK payload size when considering the characteristics of the underlying LPWAN technology, padding bits (if any) are added to the ACK payload sizes shown in the results. The subsection is organized into two subsections, which focus on the performance of the RFTs for the uniform, and for the burst error distributions, respectively.

5.7.1.1 Uniform losses

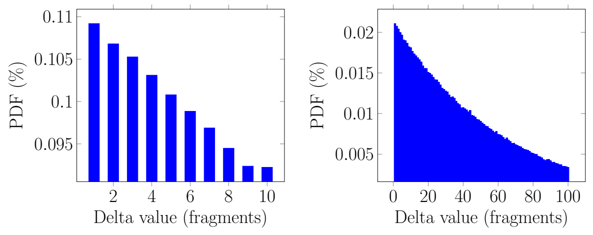
In this subsection, we evaluate the impact of a uniform fragment loss distribution on the ACK payload size for the evaluated RFTs, for FER values of 1%, 10% and 20%, uniform losses, and a range of packet sizes.

Fig. 5.9a presents the average ACK payload size for all considered RFTs, and for a FER of 1% and uniform losses. The sizes of the different ACK payloads produced by the different RFTs are very similar for small packet sizes (*i.e.*, packet sizes between 1 and 10 fragments). As packet size increases, the ACK payload size grows very rapidly for UB and CB. UB produces the greatest ACK payload size, which is linear with the number of fragments needed to carry a packet, and has a step-like behavior due to padding. CB offers better performance than UB, although its improvement is limited by losses, which reduce CB's compression gain.

LLF also has a linear behavior with packet size, since the number of fragments lost per packet is, in average, a fraction (approximately equal to the FER) of the total number of fragments required to transport the packet. Since a FER of 1% is low, LLF produces the smallest ACK payload size among the different RFTs evaluated, for all packet sizes considered.



(a) Average ACK payload size vs packet size



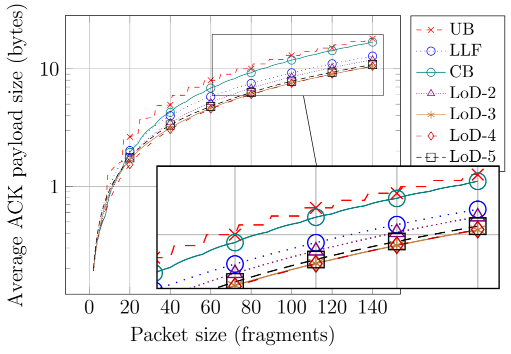
(b) Packet size: 10 fragments(c) Packet size: 100 fragments

Figure 5.9: Delta Probability Density Function (PDF), for $FER = 1\%$ and uniform losses.

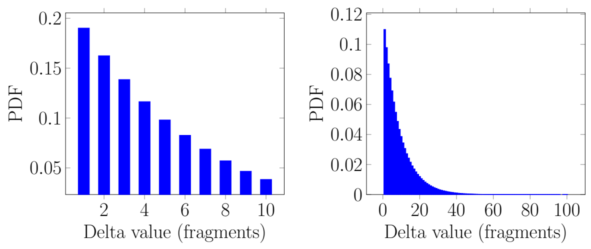
The considered LoD variants exhibit a similar behavior for small packet sizes. However, as packet size increases, LoD-2 tends to produce a greater ACK payload size due to the greater overhead of SDNV-2 when encoding large deltas. LoD-3 and LoD-5 yield a similar ACK payload size, with LoD-4 being the optimal LoD encoding for the conditions considered. LoD-4 performs similarly to LLF for a small packet size. However, as packet size increases, and since FER is low, deltas tend to increase, and therefore, LoD-4 produces a slightly greater ACK payload size than LLF.

In order to better understand the performance of the different LoD solutions, we analyzed the statistics of the deltas. Figs. 5.9b and 5.9c depict the delta PDF for packet sizes of 10 and 100 fragments, for $FER = 1\%$ and uniform losses, respectively. For both packet sizes, the delta PDF decreases steadily, from a delta value of 1, which is the most frequent delta value, up to the packet size (in number of fragments). As the probability of a delta value decreases, its encoded size increases. LoD-4 provides better performance due to its suitable trade-off between low encoding overhead and the relatively large deltas stemming from relatively infrequent errors.

Fig. 5.10a shows the average ACK payload size for $FER = 10\%$ and uniform losses. Similarly to the study for $FER = 1\%$ (Fig. 5.9a), for small packet size (of up to 10 fragments), all RFTs generate a similar ACK payload size. As packet size increases, UB/CB and LoD-3/LoD-4 yield the largest and the smallest ACK payload sizes, respectively. UB produces an ACK payload size independent of the error rate, whereas CB offers lower improvement than for $FER = 1\%$ due to the more frequent presence of losses at the end of packet transmission for $FER = 10\%$. LLF yields now a greater ACK payload size than the tested LoD schemes, as the ACK payload size is now 10 times greater than the one for a FER of 1%. Regarding the LoD schemes, LoD-2 still requires more bits to encode the deltas than the other considered LoDs, despite the fact that smaller deltas are more frequent for $FER = 10\%$ than for $FER = 1\%$ (see Figs. 5.9b and 5.9c, and Figs. 5.10b and 5.10c). On the other hand, LoD-3 is a more efficient encoding for $FER = 10\%$ than it was for $FER = 1\%$, leading to an ACK payload size similar to the LoD-4 one. LoD-5 uses 5 bits to encode small deltas, and therefore produces a slightly larger ACK payload size than LoD-3 and LoD-4.



(a) Average ACK payload size vs packet size

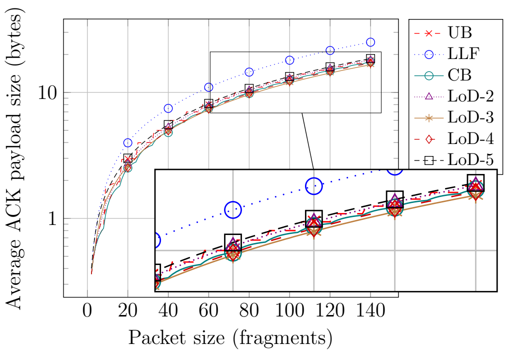


(b) Packet size: 10 fragments(c) Packet size: 100 fragments

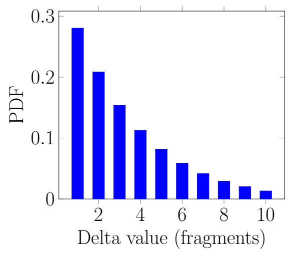
Figure 5.10: Delta Probability Density Function (PDF), for $FER = 10\%$ and uniform losses.

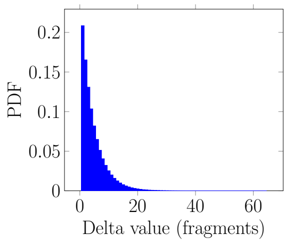
Fig. 5.11a shows the average ACK payload size for $FER = 20\%$ and uniform losses. In this case, LLF yields the largest ACK payload size among the considered RFTs, since the LLF ACK payload size is roughly proportional to the FER, which is very high at 20%. The rest of RFTs produce now very similar ACK payload sizes.

Regarding the LoD RFTs, LoD-3 yields the shortest ACK payload size, since deltas are smaller than for $FER = 10\%$ (see Figs. 5.11b and 5.11c) and they can be encoded more efficiently with a 3-bit base than with a 4-bit base. LoD-2 suffers from a too high overhead to encode deltas that are too large for the short 2-bit base, whereas the same deltas are too small for the larger 5-bit base in LoD-5. For $FER = 20\%$, UB and CB generate an ACK payload size that is in average very similar to the LoD-4 one. The UB ACK payload size remains independent of the error rate, whereas a high FER of 20% leads CB to perform similarly to UB. Therefore, UB and CB offer relatively good performance for high FER and uniform losses.



(a) Average ACK payload size vs packet size





(b) Packet size: 10 fragments(c) Packet size: 100 fragments

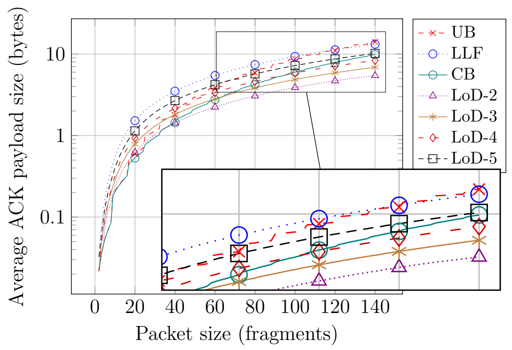
Figure 5.11: Delta Probability Density Function (PDF), for $FER = 20\%$ and uniform losses.

5.7.1.2 Burst losses

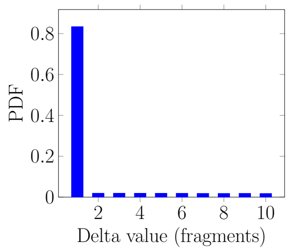
This subsection analyzes the performance of the considered RFTs for the burst error distribution, for BOP of 1% and 2%, and $\lambda = 10$ fragments *i.e.*, FER of 10% and 20%, respectively.

Fig. 5.12a depicts the average ACK payload size for $BOP = 1\%$ and $\lambda = 10$ fragments. For small packet sizes (*i.e.*, between 1 and 30 fragments), CB and LoD-2 produce the smallest ACK payload sizes, offering similar performance. For burst errors, CB improves the performance of UB to a greater extent than for uniform errors and the same FER (Fig. 5.10a). In a burst error distribution, it is more likely that the last loss will occur earlier in the packet transmission, thus allowing the compression advantages of CB to a greater extent. On the other hand, in a burst error distribution, there is a high probability of a delta value being equal to 1 (*e.g.*, 0.83 for 10-fragment packets and 0.90 for 100-fragment packets, see Figs. 5.12b and 5.12c, respectively), with other deltas corresponding to the distance between bursts. Since LoD-2 encodes the delta value of 1 with the lowest encoding overhead, LoD-2 offers good performance, outperforming the other RFTs for packet sizes greater than 40 fragments.

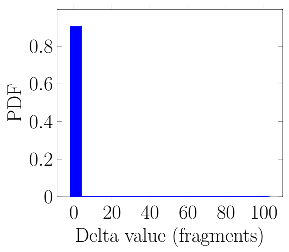
LoD-3 produces a smaller average ACK payload size than LoD-4. This is because LoD-3 requires less bits to encode the deltas (which are often equal to 1). For the same reason, LoD-5 performs worse than LoD-4. As expected, LLF exhibits a linear behavior with packet size and generates, for packet sizes up to ~ 120 fragments, the largest ACK payload size among all the evaluated RFTs. We have also evaluated the ACK payload size for $BOP = 2\%$ and $\lambda = 10$ fragments. The relative behavior of the different RFTs is qualitatively similar to the one obtained for $BOP = 1\%$, albeit for greater ACK payload size. Therefore, the results for $BOP = 2\%$ are not shown for the sake of brevity.



(a) Average ACK payload size vs packet size



(b) Packet size: 10 fragments



(c) Packet size: 100 fragments

Figure 5.12: Delta Probability Density Function (PDF), for $BOP = 1\%$.

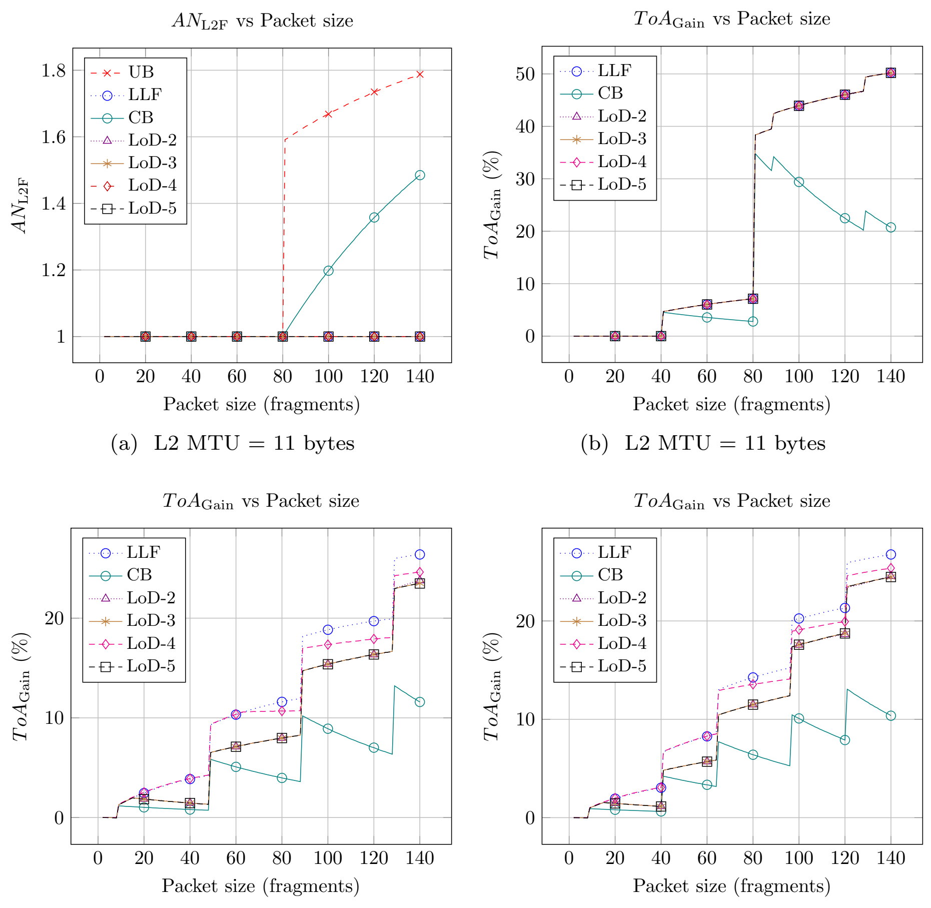
5.7.2 Average number of L2 frames and ToA gain

In this subsection, we evaluate two important performance metrics derived from the ACK payload size results: the average number of L2 frames needed to carry an ACK (denoted AN_{L2F}) and the ToA, for all packet sizes, L2 MTU values (*i.e.*,11 bytes, 51 bytes, and 242 bytes), and fragment loss scenarios considered in Section 5.7.1. For a given RFT, we represent the ToA in relative terms as the ToA Gain (ToA_{Gain}), which is obtained by dividing the UB ToA by the ToA of the considered RFT. ToA_{Gain} evaluates the benefits, if any, that can be obtained from using an RFT different from UB. ToA is calculated as described in Section 5.6.

5.7.2.1 Uniform losses

Fig. 5.13 shows the AN_{L2F} and the ToA_{Gain} results for $FER = 1\%$ and uniform losses, for the L2 MTU values considered. For a packet size smaller than 80 fragments and for an 11-byte L2 MTU, the AN_{L2F} is always one, as the ACK payload in this range fits the maximum payload size of one L2 frame (see Fig. 5.9a). For L2 MTU values of 51 and 242 bytes, the AN_{L2F} is always equal to 1. On the other hand, the ToA_{Gain} is negligible for all considered RFTs for packet sizes up to 40 fragments. This is due to the L2 header overhead, which is much greater than the ACK payload size differences for the considered RFTs, for short packet sizes. As the L2 MTU increases, and for the same range of packet sizes, the L2 overhead has lower impact and LoD-2 and LoD-3 present a ToA_{Gain} up to 4.8%. For greater packet sizes, all RFTs other than UB achieve a significant ToA improvement, for all L2 MTUs analyzed, since the differences in ACK payload size become more significant. The stepwise behavior of the ToA_{Gain} is due to the relation between the LoRaWAN frame size and its payload size (see further details in Appendix A.2).

Fig. 5.14 shows the AN_{L2F} and the ToA_{Gain} for $FER = 10\%$ and uniform losses for the



(a) L2 MTU = 11 bytes

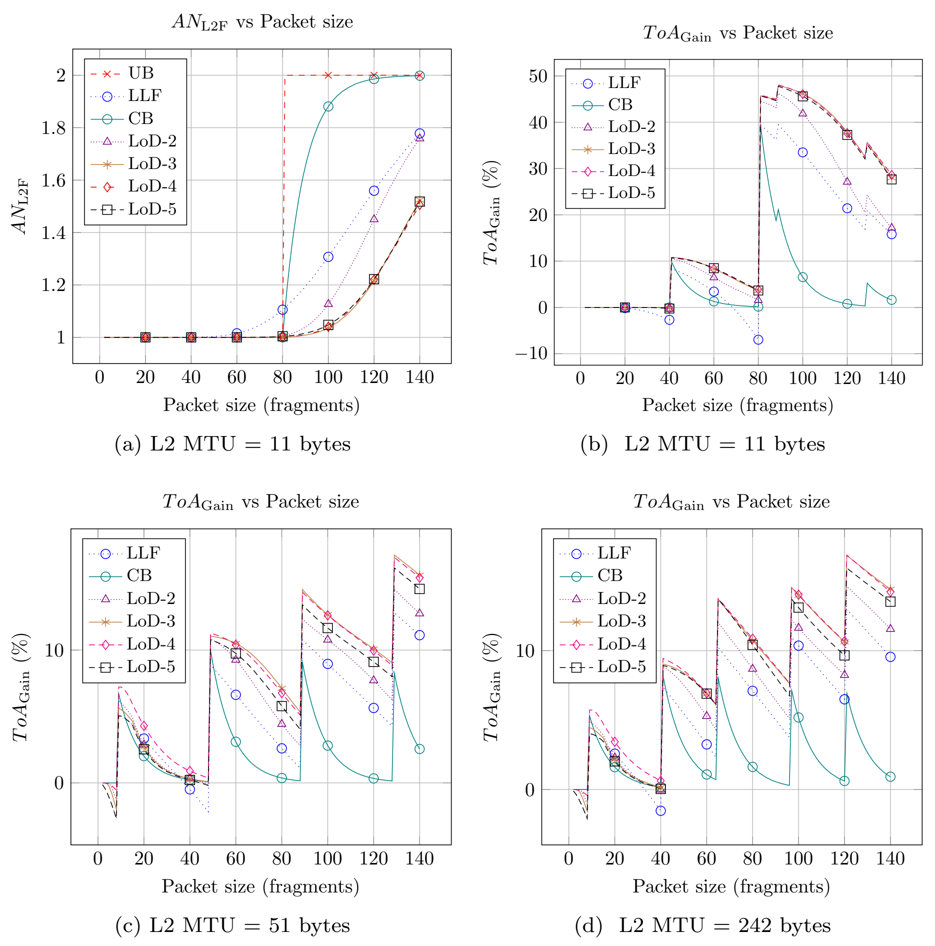
(b) L2 MTU = 11 bytes

(c) L2 MTU = 51 bytes

(d) L2 MTU = 242 bytes

Figure 5.13: Average number of L2 frames (a) and ToA_{Gain} (b,c,d) vs packet size for $FER = 1\%$ and uniform losses. In (b) results for LLF, and all LoD variants considered are overlapped.

considered L2 MTU values. For an 11-byte L2 MTU, the AN_{L2F} is also equal to one for packet sizes up to 40 fragments for LLF, and up to 80 fragments for all other RFTs. Since LLF depends strongly on the number of losses occurred during a packet transmission, its ACK payload size varies significantly, sometimes requiring two L2 frames to carry an ACK, even if its average ACK payload size is smaller than those of UB and CB (see Fig. 5.10a). For L2 MTU values of 51 and 242 bytes, the AN_{L2F} is always equal to 1. As a result, LLF exhibits even negative ToA_{Gain} for some values within the considered packet size range. For packet sizes greater than 80 fragments, UB and CB require two L2 frames to carry an ACK more often than the rest of RFTs, and the LoD variants offer the best performance. The frequent additional L2 frame penalizes UB and CB, introducing a significant ToA_{Gain} of up to $\sim 45\%$ for the rest of RFTs. This ToA_{Gain} decreases with packet size, with LoD variants, *i.e.*, LoD-3, LoD-4 and LoD-5, achieving a similar ToA_{Gain}, with values up to 27%. As the L2 MTU value increases, and for packet sizes between 2 and 8 fragments, UB and CB offer the best performance, since other RFTs yield a negative ToA_{Gain}. As packet size increases, LoD-3 and LoD-4 become optimal, with ToA_{Gain} of up to 16%. On the other hand, the ToA_{Gain} of CB is up to only 9%, since this RFT can provide a relatively low bitmap compression degree due to fragment losses occurring at the end of packet transmission with relatively high probability.



(a) L2 MTU = 11 bytes

(b) L2 MTU = 11 bytes

(c) L2 MTU = 51 bytes

(d) L2 MTU = 242 bytes

Figure 5.14: Average number of L2 frames (a) and ToA Gain (b,c,d) vs packet size for $FER = 10\%$ and uniform losses.

Fig. 5.15 illustrates the AN_{L2F} and the ToA_{Gain} for $FER = 20\%$ and uniform losses. For a small L2 MTU (*i.e.*, 11 bytes), results reflect how, for high FER, UB and CB generally offer good performance, compared with the rest of RFTs considered. LLF yields the largest AN_{L2F} and a negative ToA_{Gain} (down to -83% for a packet size of 80 fragments). Such AN_{L2F} increase happens because, in some cases, the ACK payload requires two L2 frames to be carried. As L2 MTU increases, LoD-3 and LoD-4 become the optimal RFTs, with a ToA_{Gain} of up to 9%. A seesaw ToA_{Gain} pattern arises because of the ToA stepwise behavior of UB, which makes UB yield better values than LoD-3, LoD-4, and the same ToA values as CB, for a short range of packets. For L2 MTU values of 51 and 242 bytes, AN_{L2F} is always equal to 1.

5.7.2.2 Burst losses

Fig. 5.16 depicts the AN_{L2F} and ToA_{Gain} for $BOP = 1\%$ and $\lambda = 10$ fragments for 11-byte, 51-byte, and 242-byte L2 MTUs, respectively. For a large range of packet sizes, LLF exhibits the largest AN_{L2F} for the 11-byte L2 MTU, and the smallest ToA_{Gain} for all L2 MTUs analyzed, due to its large ACK payload. Since fragment losses concentrate in bursts, LoD-2 benefits from its small overhead when encoding the highly frequent delta value of 1, and offers the best overall

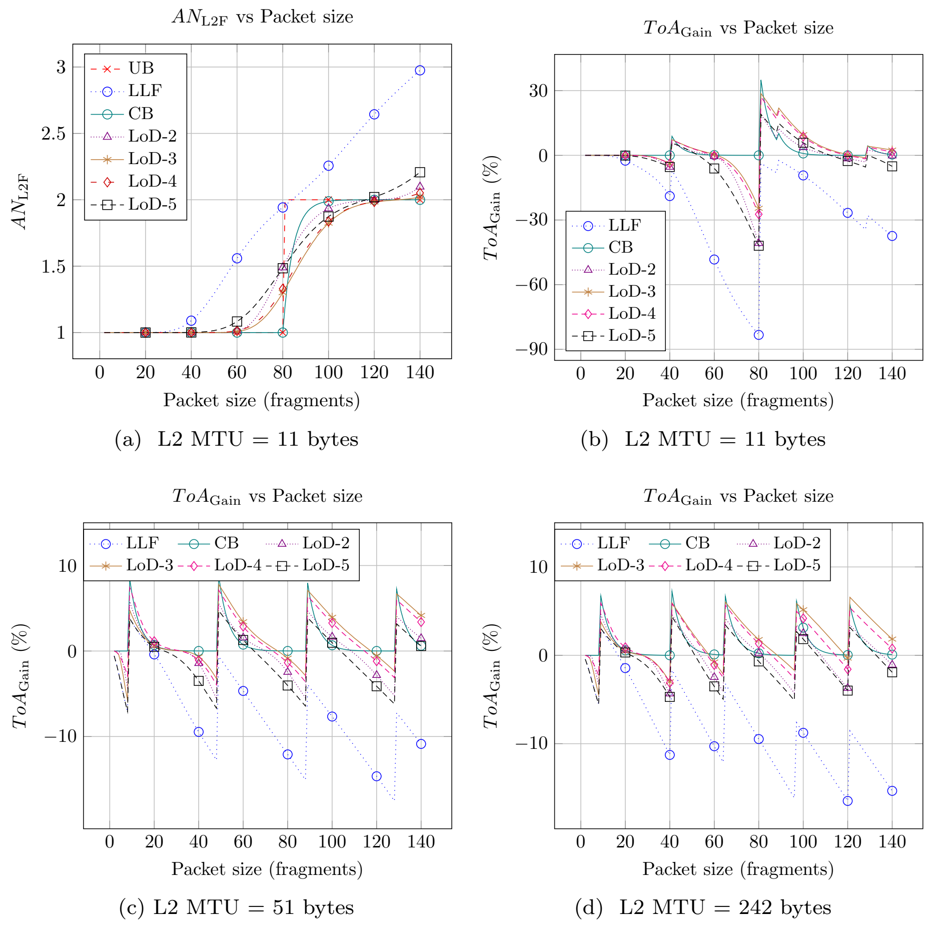
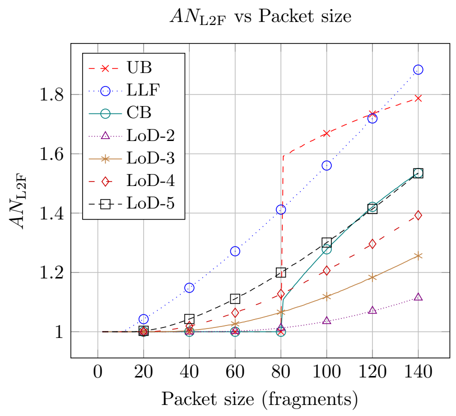
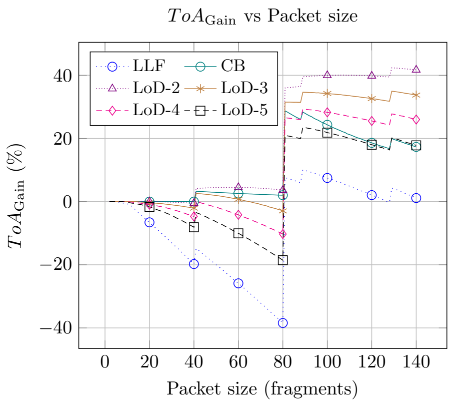


Figure 5.15: Average number of L2 frames (a) and ToA_{Gain} (b,c,d) vs packet size for $FER = 20\%$ and uniform losses.

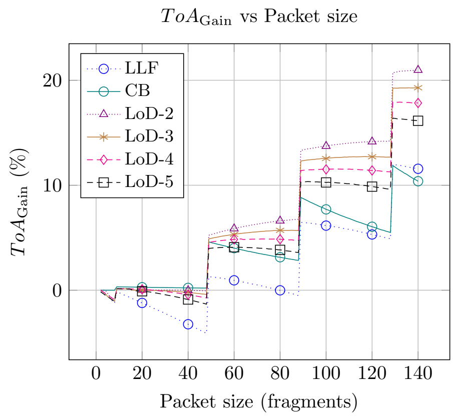
performance in terms of AN_{L2F} for the 11-byte L2 MTU, and ToA_{Gain} for all L2 MTUs analyzed. LoD-2 is followed closely by LoD-3 and LoD-4, as these LoD techniques have a larger overhead when encoding small deltas (see Fig. 5.12c). As the L2 MTU and packet size increases, UB is outperformed by all other RFTs considered (except LLF). The AN_{L2F} and ToA_{Gain} results for $BOP = 2\%$ are qualitatively similar to those for $BOP = 1\%$, with LoD-2 being the optimal RFT for a large range of packet sizes and for all L2 MTUs considered.



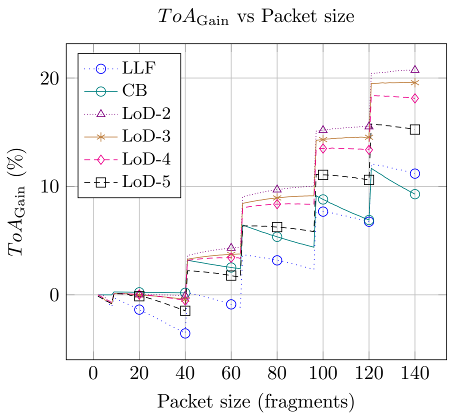
(a) L2 MTU = 11 bytes



(b) L2 MTU = 11 bytes



(c) L2 MTU = 51 bytes



(d) L2 MTU = 242 bytes

Figure 5.16: Average number of L2 frames (a) and ToA Gain (b,c,d) vs packet size for $BOP = 1\%$ and $\lambda = 10$.

CHAPTER 6
SCHC PACKET FRAGMENTATION: IMPLEMENTATION AND PERFORMANCE EVALUATION

<<Light unifies energy and matter.>>

Following the development of the SCHC framework, technology-specific versions of SCHC have been designed for specific LPWANs. In this chapter, the adaptation of SCHC over Sigfox is deeply studied, providing mathematical models for the number of SCHC messages and ACKs, and SCHC Packet transfer time, as well as an experimental evaluation of total number of uplink and downlink messages, and transfer time. We also provide the implementation (working code). Results are obtained covering all Sigfox world regions and validating SCHC over Sigfox application feasibility. As part of the results, an excess of ACK traffic was found, which offered optimization opportunities. Contributions of this thesis investigation led to the co-authorship of the SCHC over Sigfox LPWAN WG Internet Draft [VIII], and the development of a new message (along with a corresponding mechanism) called SCHC Compound ACK, which was adopted as an LPWAN WG Internet Draft [IX]. The SCHC Compound ACK message is presented in Chapter 9. We believe that the results provide useful insights for researchers, developers, implementers, and providers, with applicability to the application design, network planning, and resource management of IoT solutions. This investigation was performed in collaboration with the Sigfox company and the University of Chile. The contributions in this chapter have been published in a journal article [III] and in a conference paper [VII].

6.1 Introduction

Considering the diversity of LPWAN technologies and scenarios, SCHC was purposefully designed to offer generic, technology-independent functionality. To optimize the use of SCHC over a given LPWAN technology, a specific definition of the SCHC mechanism choices and parameter settings over that technology (called a SCHC Profile) is needed. As of the writing of this chapter, the IETF LPWAN WG has worked on profiling SCHC over flagship LPWAN technologies such as Sigfox [7], LoRaWAN [9], and NB-IoT [8]. Since C/D depends on static context rules to be defined by the network operator/administrator on a per-deployment basis, SCHC Profiles mainly focus on specifying how F/R is performed over each particular technology.

In this chapter, and for the first time to our best knowledge, we investigate the performance of SCHC over Sigfox. We focus on its most promising feature: a reliable, yet efficient F/R mode called ACK-on-Error, which is intended for uplink communication. We provide a theoretical analysis and an experimental evaluation of crucial performance metrics such as packet transfer time and number of Sigfox messages required when using ACK-on-Error. For the experimental evaluation, we implemented a real testbed and performed measurements in two different world regions corresponding to different Sigfox geographical zones, which are characterized by different Sigfox radio features. Among others, our results show that in some cases fragment losses may reduce the transfer time, compared with a scenario without losses. Moreover, very small packet size changes can significantly impact the packet transfer time. Also, we found that the number of uplink and downlink messages is not proportional to a Fragment Loss Rate (FLR) increase due to the fact that downlink messages are device-driven. As a side-contribution of this work, we published our SCHC over Sigfox implementation source code [98, 99].

The remainder of the chapter is organized as follows: Section 6.2 presents related work. Section 6.3 overviews SCHC over Sigfox, focusing on its uplink ACK-on-Error mode. Section 6.4 presents a theoretical analysis of the transfer time, and the total number of uplink and downlink messages required to carry a fragmented generic data packet over Sigfox. Section 6.5 describes the environment used in our experimental evaluation and presents the results and discussion.

6.2 Related work

Recently, SCHC has attracted the attention of academia and industry as it enables IPv6 connectivity over LPWAN networks [6–9, 46, 47, 57, 60, 100–104].

The definition of specific SCHC Profiles (see Section 2.3.7) is progressing rapidly after the publication of the generic SCHC specification, RFC 8724 [6]. The first profile, SCHC over LoRaWAN has become RFC 9011 [9]. Similar paths are being followed by profile definitions of SCHC over Sigfox [7] and SCHC over NB-IoT [8]. At the same time, there is an increased interest in implementing SCHC and SCHC profiles in simulation, experimental, and industrial settings. The Open-SCHC initiative [101] has been evolving together with the IETF's standardization track, with an open-source Python implementation of SCHC that also allows rapid testing via a simulation tool.

To compare SCHC performances to previous proposals of the IETF for different technologies, Ayoub et al. implemented a generic version of SCHC in the ns-3 simulator, focusing on the compression/decompression function [47]. In [102, 103], the performance of SCHC over LoRaWAN is modeled and validated with an experimental testbed limited to the ACK-on-Error mode for uplink traffic. Toutain et al. [104] provided an early implementation in Python of the compression/decompression SCHC function. The code was tested in Pycom devices using LoRa, although the authors mentioned it could be easily adapted for Sigfox. In [57], the generic SCHC specifications where simulated using Open-SCHC, providing performance results of SCHC fragmentation, without fragment losses, while using LoRaWAN. Sanchez-Gomez et al. provided an evaluation of the SCHC profile over LoRaWAN using a real testbed that considers regular IPv6 and CoAP traffic [60]. The evaluation established the benefits of using SCHC in terms of delay and packet delivery ratio, and assessed the need for resources in the constrained devices (*i.e.*, computing and memory requirements). To test SCHC in multimodal LPWAN solutions, Moons et al. implemented a generic library for constrained devices using the OSS-7 operating system and the Click modular router for packet processing and IPv6 forwarding [46].

The work mentioned above provides valuable insights into SCHC performance at both generic and profile levels, but mainly focused on the LoRaWAN technology. In this work, we evaluate the performance of SCHC over Sigfox, both analytically and experimentally. The experimental evaluation is based on the first SCHC implementation that addresses the specifics of the Sigfox technology in a scenario of packet fragmentation and reassembly.

6.3 SCHC over Sigfox Profile

The SCHC framework defers to each LPWAN technology profile the proper choice and configuration of the F/R modes to be used (*e.g.*, RuleID field size, N size, M size, padding bits), so that SCHC is suitably adapted to the characteristics of each radio technology. In this section, we describe how SCHC F/R is used over Sigfox.

6.3.1 SCHC over Sigfox overview

In Sigfox, as in other LPWAN technologies, uplink traffic is typically dominant. The SCHC over Sigfox specification [7] provides two modes for uplink fragmentation: No-ACK and ACK-on-Error. No-ACK mode is intended for the transmission of short, non-critical SCHC Packets, as it does not provide reliability. In contrast, ACK-on-Error defines two variants, offering reliability for short and long SCHC Packets, respectively. ACK-on-Error also reduces the number of ACKs when compared with the other reliable SCHC fragmentation mode (i.e. ACK-Always) [59, 100]. This is suitable considering that in Sigfox, the downlink message rate is very limited (see Section 2.2.1.2). This chapter focuses on the ACK-on-Error mode, as it provides reliability and efficiency for a very wide range of SCHC Packet sizes.

6.3.2 Architecture

The SCHC over Sigfox architecture under study is composed of three main elements (see Fig. 6.1): the SCHC F/R Sender (hereinafter, the sender), the Sigfox Network, and the SCHC F/R Receiver (hereinafter, the receiver). The sender and the receiver are hosted by the device and the application server, respectively.

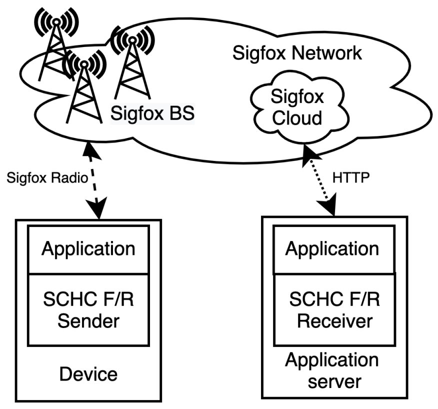


Figure 6.1: SCHC over Sigfox architecture.

6.3.3 ACK-on-Error mode over Sigfox

In this section we present the two variants of ACK-on-Error mode over Sigfox. These variants use a single-byte and a two-byte fragment header size, respectively.

6.3.3.1 Single-byte-header ACK-on-Error

SCHC over Sigfox recommends an 8-bit fragment header for SCHC Packets of a size up to 300 bytes. The fragment header (see Fig. 2.24) is composed of a 3-bit RuleID, a 2-bit W number size ($M = 2$), and a 3-bit FCN size ($N = 3$). Neither RCS and DTag are used. WINDOW_SIZE is equal to 7 tiles. The tile size is fixed to 11 bytes (*i.e.*, one tile per SCHC Fragment). The SCHC ACK header includes a 3-bit RuleID, a 2-bit W and it may include a 7-bit bitmap (when there are SCHC Fragment losses), for a total of 5 or 13 bits, depending on whether the ACK reports success or failure, respectively (see Fig. 2.25). In Sigfox, the downlink payload size must always be 64 bits therefore, padding bits must be added.

6.3.3.2 Two-byte-header ACK-on-Error

Using the two-byte SCHC Fragment header supports the fragmentation and reassembly of SCHC Packets with a size up to 2250 bytes. This is achieved by adding 1 bit to the W field and 2 bits to the FCN, compared with the single-byte SCHC header. The 16 SCHC Fragment header bits are organized as an 8-bit RuleID, a 3-bit W size ($M = 3$), and a 5-bit FCN size ($N = 5$). RCS and DTag are not used. WINDOW_SIZE is equal to 31 tiles. Compared to the single-byte SCHC header, the W field size now allows for up to 2 times more windows per SCHC Packet and the FCN field size allows up to 4 times more fragments per window. However, the tile size is now fixed to 10 bytes, one byte less. The SCHC ACK size is now of 11 or 43 bits, depending on whether SCHC ACK reports success or failure, respectively (see Fig. 2.25). Padding bits are added to complete the required downlink frame payload size of 64 bits.

The two-byte SCHC Fragment header may be used to comply with the IPv6 MTU of 1280 bytes, and may also be useful for applications that require long packet sizes, such as smart meters (*e.g.*, gas, water, etc.), waveform captures, data logs, and large data packets using rich data types [97,105–107].

6.3.3.3 Operation specifics

In the SCHC over Sigfox Profile, in an intermediate window (*i.e.*, a window that is not the last one), an All-0 SCHC Fragment signals the end of the window. The All-0 is sent by using a B-procedure, in order to open a downlink opportunity (see Section 2.2.1.2). The receiver may use this downlink opportunity to send a SCHC ACK if there are fragment losses in current or previous windows, and if the network assesses that radio conditions are favorable. After resending all lost fragments or if the SCHC ACK is not received after the Retransmission Timer expires, the sender continues with the transmission of SCHC Fragments of the next window.

At the end of the last window (*i.e.*, after the All-1 SCHC Fragment), the sender always expects a SCHC ACK therefore, it opens a downlink opportunity. The All-1 SCHC Fragment is sent using a B-procedure (see Section 2.2.1.2), and is also used to request a SCHC ACK after retransmission of lost SCHC Fragments.

The SCHC over Sigfox draft specification recommends setting MAX_ACK_REQUESTS to 5. Both the Retransmission Timer and the Inactivity Timer are application-dependent and need to be consistent with spectrum access regulations (e.g. *duty-cycle* constraints). RCS is not always recommended because Sigfox performs an integrity check for each uplink frame delivered by the Sigfox Cloud to the receiver (see Section 2.2.1.2). The DTag is not used as different RuleIDs can be used to interleave different SCHC Packets sent simultaneously.

Finally, another Sigfox-specific feature is the use of the Sigfox Sequence Number to keep track of SCHC Fragment transmissions and to identify missing fragments.

6.3.3.4 Single- vs two-byte-header trade-off

SCHC over Sigfox provides different configurations of ACK-on-Error mode to serve different use cases and SCHC Packet sizes. On the one hand, the single-byte ACK-on-Error mode requires less overhead for small SCHC Packet sizes and a smaller window size (with a smaller bitmap), which increases the downlink opportunities, *i.e.*, the number of times the receiver can provide feedback to the sender and recover from eventual losses. On the other hand, the two-byte ACK-on-Error mode requires a larger overhead with lesser downlink opportunities, which is traded for a faster transfer time for large SCHC Packet sizes.

6.4 Theoretical analysis

In this section we present a theoretical performance analysis of the uplink SCHC Packet transfer over Sigfox by using the ACK-on-Error mode. First, we obtain the total delay of a Sigfox message transmission by using the U-procedure and the B-procedure. Based on the results, we calculate the total SCHC Packet transfer time under the conditions considered. Finally, we derive the number of uplink (hereinafter, UL) and downlink (hereinafter, DL) messages required to perform a SCHC Packet transfer. This number is critical since message rates in Sigfox are limited, as aforementioned (see Section 2.2.1.2).

We assume that no fragment losses occur. Therefore, our theoretical model provides a lower bound on the SCHC Packet transfer time, as well as on the number of UL and DL messages required to transfer a SCHC Packet. Note that, in Section 6.5, we experimentally evaluate the impact of fragment losses on performance.

6.4.1 Sigfox message transmission U/B procedures delay

We now derive the Sigfox message total delay for the U-procedure and the B-procedure.

6.4.1.1 U-procedure delay

The U-procedure (see Fig. 2.6) total delay ($T_{U-total}$) can be obtained as follows:

$$T_{U-total}(sec) = 3 \cdot T_{Tx} + 2 \cdot T_{Wait_{Tx}} + T_{Cool}. \tag{6.1}$$

The duration of the transmission state (T_{Tx}) depends on the Sigfox frame length and the Bitrate (BR). Recall that each Sigfox RC defines an UL BR of either 100 bit/s or 600 bit/s (see Section 2.2.1.2) and the Sigfox protocol overhead per UL data unit is 96 bits (see Fig. 2.7). Then, T_{Tx} can be calculated as:

$$T_{Tx} = \frac{96 + L_{MAUTH} + L_{UL}}{BR}, \tag{6.2}$$

where L_{MAUTH} depends on the UL payload size (L_{UL}) and it can be obtained from Table 2.5.

6.4.1.2 B-procedure delay

The B-procedure total delay ($T_{B-Total-DL}$) can be obtained as follows:

$$\begin{aligned} T_{B-total-DL}(sec) = {} & 3 \cdot T_{Tx} + 2 \cdot T_{Wait_{Tx}} + T_{Wait_{Rx}} \\ & + T_{Rx} + T_{Conf} + T_{Cool}, \end{aligned} \tag{6.3}$$

where T_{Tx} is derived from (6.2). Note that this calculation assumes that a DL frame is actually received after the three UL frame transmissions (see Fig. 2.8). However, in some cases, a DL frame might not be received during a B-procedure. This event can occur when a DL message is actually not transmitted or when such message is not successfully delivered to the device by the radio link.

The B-procedure total delay when no DL is received ($T_{B-Total-no-DL}$) can be derived from (6.3), with $T_{Rx} = T_{Rx_{MAX}}$, and $T_{Conf} = 0$. Indeed, once $T_{Rx_{MAX}}$ time has passed, the radio proceeds to cooldown. Note that $T_{B-Total-no-DL} \geq T_{B-Total-DL}$ as $T_{Rx_{MAX}} \geq T_{Rx}$. This means that when a DL frame is received, the B-procedure delay may be smaller than when a DL frame is not received.

6.4.2 SCHC Packet transfer time: Analysis

To obtain the SCHC Packet transfer time (T_{SCHC}), we first calculate the transmission time of each SCHC Fragment type and of the SCHC ACK. Then we determine the number of SCHC Fragments and SCHC ACKs required to transfer the SCHC Packet.

As explained in Section 6.3, a SCHC Packet is carried by different types of SCHC Fragments, such as the Regular SCHC Fragment, including the All-0, and the All-1 SCHC Fragment. All SCHC Fragments, except for the All-1, have a fixed tile size. We next determine the transfer delay of each type of SCHC Fragment.

A Regular SCHC Fragment that is not the All-0 is transmitted using the Sigfox U-procedure (see Section 6.4.1.1), therefore its transmission time (T_{U-proc}) is given by (6.1), where T_{Tx} is given by (6.2), with $L_{UL} = 12$ bytes.

The transmissions of the All-0 and All-1 SCHC Fragments use the Sigfox B-procedure. When there is no fragment loss, the All-0 transmission time (T_{All-0}) is equal to $T_{B-Total-no-DL}$, and it can be calculated using (6.3), with $L_{UL} = 12$ bytes.

The All-1 SCHC Fragment carries the last fragment of the SCHC Packet and its size may be smaller than the Regular SCHC Fragment (which has a size of 12 bytes). The All-1 SCHC Fragment size L_{All-1} can be calculated as:

$$L_{All-1} = L_{SCHC} - (\left\lceil \frac{L_{SCHC}}{L_{UL_{MAX}} - L_H} \right\rceil - 1) \cdot (L_{UL_{MAX}} - L_H), \tag{6.4}$$

where L_H is the size of the SCHC Fragment header, $L_{UL_{MAX}}$ is the maximum UL payload size and L_{SCHC} is the size of the SCHC Packet. In UL ACK-on-Error mode, L_H can be either 1 or 2 bytes. The All-1 SCHC Fragment transmission time (T_{All-1}) can be calculated using (6.2)-(6.4).

The number of SCHC Fragments using the U-procedure required in a SCHC Packet transfer (N_{U-proc}) can be obtained as follows:

$$N_{U-proc} = \left\lceil \frac{L_{SCHC}}{L_{UL_{MAX}} - L_H} \right\rceil - N_{All-0} - 1. \tag{6.5}$$

The number of All-0 SCHC Fragments used in a fragmented SCHC Packet transmission (N_{All-0}) can be obtained as follows:

$$\begin{aligned} N_{All-0} &= N_{Windows} - 1 \\ &= \frac{L_{SCHC}}{WINDOW\_SIZE \cdot t} - 1, \end{aligned} \tag{6.6}$$

where $N_{Windows}$ is the number of windows required to transmit the SCHC Packet. The number of All-1 SCHC Fragments (N_{All-1}), when there is no loss, is 1 for all SCHC Packet sizes.

Once the total number of SCHC Fragments and their transmission times have been calculated, T_{SCHC} can be obtained as follows:

$$T_{SCHC} = N_{U-proc} \cdot T_{U-proc} + N_{All-0} \cdot T_{All-0} + N_{All-1} \cdot T_{All-1}. \tag{6.7}$$

In Sigfox RCs where *duty-cycle* restrictions apply, the device must keep the transmission radio off for a certain amount of time ($T_{Tx_{OFF}}$), after UL transmissions. This time can be calculated as follows:

$$T_{Tx_{OFF}} = \frac{3 \cdot T_{Tx}}{DC} - 3 \cdot T_{Tx}. \tag{6.8}$$

where DC denotes the *duty-cycle* enforced (*e.g.*, $DC = 0.01$ corresponds to a *duty-cycle* of 1%).

The total time off for a SCHC Packet transfer (T_{OFF}) can be determined as follows:

$$T_{OFF} = (N_{U-proc} + N_{All-0} + N_{All-1}) \cdot T_{Tx_{OFF}}, \tag{6.9}$$

where $T_{Tx_{OFF}}$ can be approximated to 600 s [1].

Finally, the total SCHC Packet transfer time, including all inactive intervals due to enforcing *duty-cycle* regulations ($T_{SCHC_{DC}}$), can be obtained as:

$$T_{SCHC_{DC}} = T_{SCHC} + T_{OFF}. \tag{6.10}$$

6.4.3 SCHC Packet transfer time: Results

To obtain the SCHC Packet transfer time, first the U-Procedure and B-Procedure states were measured by using an Agilent N6750A power analyzer on a Pycom LoPy4 development board [108], as shown in Fig. 6.2.

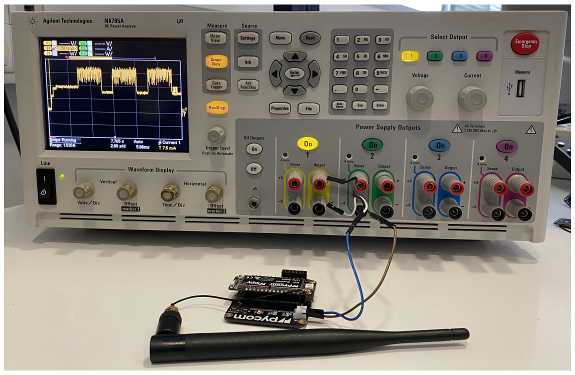


Figure 6.2: Experimental setup for SCHC Packet transfer time measurements.

Table 6.1 shows the measured duration of each U-procedure state for Sigfox RC1 and RC4. Conversely, Table 6.2 shows the measured B-procedure transmission states duration. Note that T_{Rx} is variable within a range of values. From our experiments in RC1, the mean reception time $T_{Rx_{Mean}}$ is 14.5 s. Recall that, when no DL frame is received, T_{Rx} takes its maximum value, $T_{Rx_{MAX}}$.

Table 6.1: U-Procedure Transmission States and Their Corresponding Durations

State number	State notation	Duration (ms)	
		RC1 (100 bit/s)	RC4 (600 bit/s)
1	T_{TX}	[1120, 2080]	[186.67, 346.67]
2	$T_{Wait_{Tx}}$	1000	500
3	T_{Cool}	1000	1000

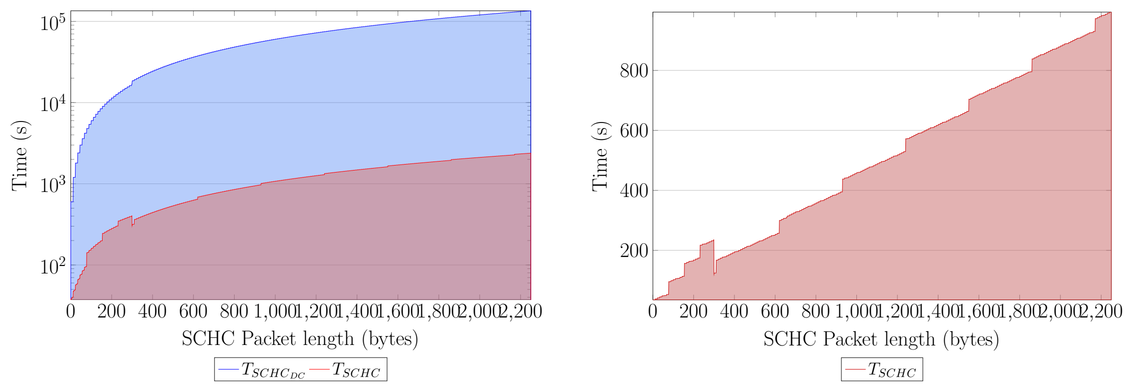
Figs. 6.3a and 6.3b show the SCHC Packet transfer time (T_{SCHC}) for SCHC Packet sizes from 0 to 2250 bytes, using the values from Tables 6.1 and 6.2, with $T_{Rx_{Mean}} = 14.5\ s$, and using (6.7) and (6.9), for RC1, and a *duty-cycle* of 1%, and RC4, respectively. SCHC Packet transfer time tends to increase linearly with the SCHC Packet size. The stepwise behavior of T_{SCHC} as SCHC Packet size increases happens because every 77 or 310 bytes, an additional window is needed (with

[1]see `https://build.sigfox.com/study` (accessed on 19/01/2023).

Table 6.2: B-Procedure Fragment Transmission States and Their Corresponding Durations

State number	State notation	Duration (ms)	
		RC1 (100 bit/s)	RC4 (600 bit/s)
1	T_{TX}	[1120, 2080]	[186.67, 346.67]
2	$T_{Wait_{Tx}}$	500	500
3	$T_{Wait_{Rx}}$	15556	15556
4	T_{Rx}	[387,25000]	[387,25000]
5	T_{Conf}	1799	1799
6	T_{Cool}	1000	1000

the corresponding wait time after sending an All-0 message) for single-byte or two-byte SCHC Fragment headers, respectively. Therefore, a small change in SCHC Packet size may lead to a sudden increase in T_{SCHC}. However, in RC4, a slight SCHC Packet size increase, from a SCHC Packet size slightly below 300 bytes, leads to a T_{SCHC} decrease by a factor of ~ 2 (see Fig. 6.3b). This sudden discontinuity at 300 bytes occurs because, at that size, the UL ACK-on-Error mode changes from a single-byte to a two-byte SCHC Fragment header. The stepwise behavior is also reflected in $T_{SCHC_{DC}}$ (see Fig. 6.3a). However, the number of fragments transmitted, and their subsequent $T_{Tx_{OFF}}$, become more dominant than the number of windows regarding their impact on $T_{Tx_{OFF}}$.



(a) RC1 (b) RC4

Figure 6.3: Theoretical SCHC Packet transfer time for (a) RC1 and (b) RC4. In (a), the *duty-cycle* is of 1% and the total time off (T_{OFF}) required to comply with *duty-cycle* regulations is depicted in shaded blue.

6.4.4 Number of messages (UL and DL)

The number of Sigfox UL messages (N_{UL}) required to transmit a SCHC Packet using ACK-on-Error mode can be obtained as follows:

$$N_{UL} = N_{U-proc} + N_{All-0} + N_{All-1}. \tag{6.11}$$

In absence of fragment losses, N_{UL} can be calculated as:

$$N_{UL} = \left\lceil \frac{L_{SCHC}}{L_{UL_{MAX}} - L_H} \right\rceil . \tag{6.12}$$

In the considered conditions, the number of DL messages (N_{DL}) is equal to 1 for all SCHC Packet sizes. This is because the ACK-on-Error mode, when there are no fragment losses, only requires one SCHC ACK at the end of the SCHC Packet to confirm the correct reception of all SCHC Fragments.

Table 6.3 shows the number of UL and DL messages required to transfer SCHC Packets of different sizes when using the UL ACK-on-Error mode over Sigfox with no fragment losses. The results given in Figs. 6.3a and 6.3b and Table 6.3 will be used as benchmarks for the experimental results given in Section 6.5.

Table 6.3: Number of UL and DL Messages in a No Loss Scenario

SCHC Packet size (bytes)	N_{UL}	N_{DL}
0	1	1
11	1	1
22	2	1
77	7	1
90	9	1
150	14	1
231	21	1
233	22	1
512	52	1
1280	128	1
2250	225	1

6.5 Experimental evaluation

In this section, we present our experimental performance evaluation of the UL SCHC Packet transfer over Sigfox using the ACK-on-Error mode. First, we present the experimental environment. Then, we measure the SCHC Packet transfer time and the number of UL and DL messages required for SCHC Packet transfer.

6.5.1 Experimental environment

In order to study the performance of UL ACK-on-Error mode over Sigfox, we implemented an experimental scenario consisting of the main elements of the SCHC over Sigfox architecture (presented in Section 6.3.2). We developed the sender on a Pycom LoPy4 hardware platform. The LoPy4 uses the low-cost, low-power ESP32 system on a chip microcontroller [2] from Espressif, which is widely used in IoT applications. Furthermore, this platform is compatible with the Pycom SiPy development board [109], which has the Sigfox Verified certification.

The receiver was developed to run as a Cloud Function in the Google Cloud Platform [110], where the latter serves as the application server. The Sigfox Network forwards the SCHC Fragments

[2]https://www.espressif.com/en/products/socs/esp32, accessed on 03/01/2023.

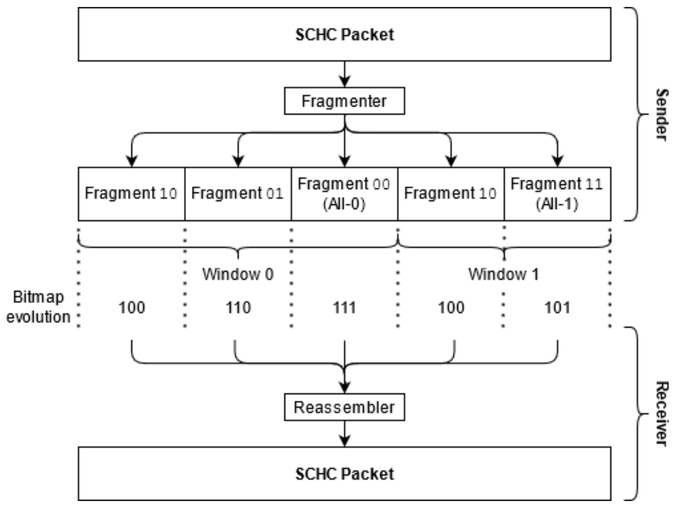


Figure 6.4: SCHC F/R mechanism diagram using `Fragmenter` and `Reassembler` objects.

received from the device to the application server using HTTP messages. Both sender and receiver share the same Python-based implementation, adapted to the requirements of each host. For example, Cloud Functions retain no memory between executions. Because of this, and due to its fast response time, Firebase Realtime Database [111] was used to save all transmission data (*e.g.*, SCHC Packets, SCHC Fragments, bitmaps, Sequence Numbers, etc).

The code implementation is class-based to ease modifications and extensions and it corresponds to an updated version of the one presented in [112]. A `Profile` class is defined and is extended in the `Sigfox(Profile)` subclass, which sets all the parameters defined in the SCHC over Sigfox profile [7]. In the sender, an object named `Fragmenter` is instantiated to perform the fragmentation process. Each fragment obtained in the process belongs to the `Fragment` class. This class defines the header and payload format for every SCHC Fragment type, particularly specifying the Rule ID, the window number, and the FCN. In the receiver, the `Reassembler` object recovers the original message and performs the reassembly process. Fig. 6.4 depicts the F/R process for an example of three fragments per window.

In our implementation, RuleIDs for single-byte or two-byte ACK-on-Error start with 0b1 or 0b01, respectively. Note that the remaining RuleIDs may be used for C/D.

To keep track of the received fragments, a bitmap is created at the receiver (see Fig. 6.4). This bitmap becomes part of the SCHC ACK Message that is sent back to the sender, if any fragment is not received. When the transmission is completed, the last executed instance of the receiver performs the reassembly process, reads the fragments from Firebase Realtime Database, and stores the SCHC Packet in the same service.

Moreover, at the application server, we developed a fragment loss emulator, which enables the artificial injection of fragment losses. At both sender and receiver, we developed a data tracker to facilitate the capture of SCHC Packet transfer metrics (*e.g.*, Sigfox procedure transmission time, total number of fragments sent by the sender and receiver, fragment losses, etc). The source code of the implementations for both sender and receiver are publicly available [98, 99].

6.5.2 Experiment details

The experiments were performed in Sigfox RC1 (in Barcelona, Spain) and RC4 (in Santiago, Chile). In both scenarios, the FLR was found to be approximately 0%. This allows the validation

Table 6.4: SCHC Packet Sizes Evaluated

SCHC Packet size (bytes)	Number of SCHC Fragments	$N_{Windows}$	L_H (bytes)
77	7	1	1
150	14	2	1
231	21	3	1
512	52	2	2

of the theoretical model, which was developed assuming no fragment losses. However, to test the performance of the SCHC over Sigfox ACK-on-Error mode in the presence of losses, fragment losses following a Bernoulli distribution error distribution were artificially introduced at the receiver with FLR values of 10% and 20%. We performed experiments of three types: i) with no losses, ii) with losses in the UL only, and iii) with losses both in the UL and the DL (UL/DL).

Table 6.4 shows the SCHC Packet sizes employed in the experiments, with the corresponding number of SCHC Fragments, $N_{Windows}$, and L_H. These SCHC Packet sizes allow analyzing the impact of the number of windows on the total SCHC Packet transfer time and on the number of UL and DL messages.

The Retransmission Timer and the Inactivity Timer were set to 60 s and 200 s, respectively. SCHC Packet transfer time results (see Section 6.5.4) were obtained from the average of 10 and 20 experiments for each combination of parameters (*i.e.*, SCHC Packet size, FLR, and RC).

6.5.3 SCHC Packet transfer time: Analysis validation

To validate the theoretical SCHC Packet transfer time results presented in Section 6.4, we compare such results with experimental results (Fig. 6.5). As it can be seen, experimental results of T_{SCHC} are very close to the theoretical ones, with gaps that do not exceed 3.38% in the worst case. For small SCHC Packet sizes (*i.e.*, with a small number of U-procedures), these gaps are due to the variability of T_{Rx} in the experimental values of $T_{B-total-DL}$, whereas the theoretical value of T_{Rx}, obtained as an average value, is slightly smaller than the experimental ones. For large SCHC Packet sizes (i.e, with a larger number of U-procedures), the SCHC Packet transfer time gaps are mainly due to the fact that the measured and experimental values of $T_{U-total}$ are slightly different. For a Regular SCHC Fragment that is not an All-0, $T_{U-total_{theoretical}} = 9.24\ s$, whereas $T_{U-total_{experimental}} = 9.4\ s$. The greater value of the latter is due to the processing time overhead of the Sigfox API of LoPy4 [113]. As N_{U-proc} increases, the $T_{U-total}$ gap becomes more dominant than the $T_{B-total-DL}$ gap, making the experimental SCHC Packet transfer time slightly greater (up to 1.30% for a 2250-byte SCHC Packet) than the theoretical one.

6.5.4 SCHC Packet transfer time: Experimental results

In the following, we present the experimental SCHC Packet transfer time results, T_{SCHC}, obtained in RC1 (see Fig. 6.6) and RC4 (see Fig. 6.7), respectively. Note that RC1, RC3, RC5, and RC6 have the same UL BR, *i.e.*, 100 bit/s, whereas both RC4 and RC2 also share the same 600 bit/s. Therefore, T_{SCHC} results for RC1 and RC4 can be mapped to all the other Sigfox RCs.

T_{SCHC} increases with the UL FLR and $N_{Windows}$, as a greater number of SCHC Fragment transmissions and retransmissions are needed. SCHC Packet sizes with equal $N_{Windows}$, *e.g.*, 150-byte and 512-byte, exhibit similar behavior with the FLR. The small T_{SCHC} decrease as DL FLR increases is due to fact that Regular SCHC Fragment losses will yield a smaller time. This happens

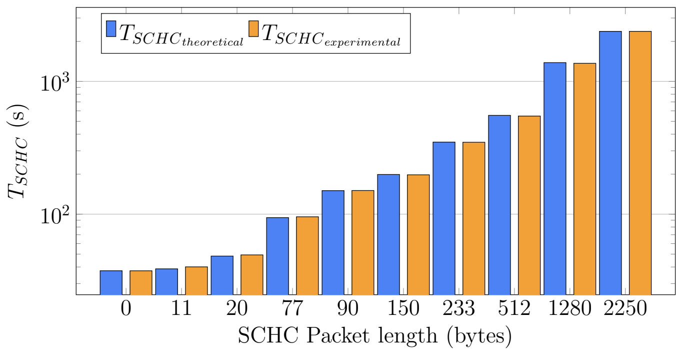


Figure 6.5: Theoretical and experimental SCHC Packet transfer time with 1% *duty-cycle*. Sigfox zone RC1.

because $T_{B-total-DL}$ is smaller when a SCHC ACK is received (see Section 6.4.3). T_{SCHC} and $T_{SCHC_{DC}}$ show similar behaviors with SCHC Packet size and FLR (see Fig. 6.6).

6.5.5 Average number of messages (UL and DL)

Fig. 6.8 shows the experimental results for N_{UL} in RC1 and RC4, for the different cases analyzed. Recall that N_{UL} comprises N_{U-proc}, N_{All-0}, and N_{All-1}. N_{U-proc} increases with UL FLR due to retransmissions. Moreover, N_{All-1} increases with both UL and DL FLR. This happens because Regular SCHC Fragment retransmission cycles trigger additional All-1 SCHC Fragments to request SCHC ACKs and losses in the DL (*i.e.*, in SCHC ACKs) trigger an UL message (*i.e.*, All-1) that requests again the lost SCHC ACK.

The N_{All-0} increase with UL and DL FLR is small, since this SCHC Fragment will only be requested again if lost. However, N_{All-0} increases linearly with $N_{Windows}$. SCHC Packet sizes with the same number of windows, *e.g.*, 150-byte and 512-byte, yield similar N_{All-0} values, independently of the SCHC Packet size.

Fig. 6.9 presents the experimental results for N_{DL} in RC1 and RC4, for all studied cases. N_{DL} increases with the UL and DL FLR. As UL FLR increases, more retransmissions are needed to successfully transmit all SCHC Fragments, increasing also the number of SCHC ACKs. Increasing

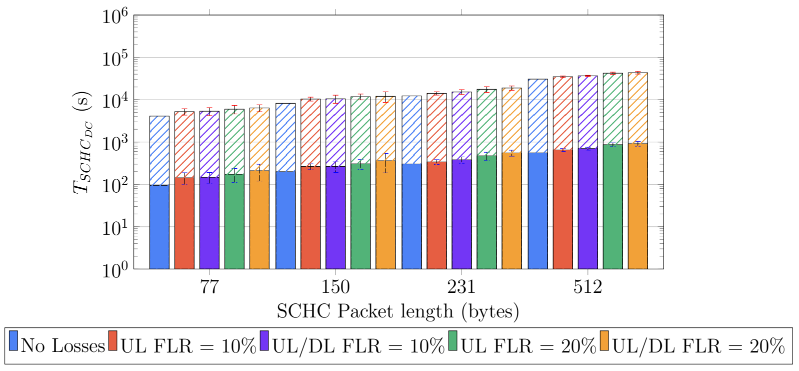


Figure 6.6: Experimental SCHC Packet transfer time: results obtained in RC1 (1% *duty-cycle*). The solid colored part of the bars represents T_{SCHC}, and the dashed part illustrates the time off (T_{OFF}) required to comply with *duty-cycle* regulations.

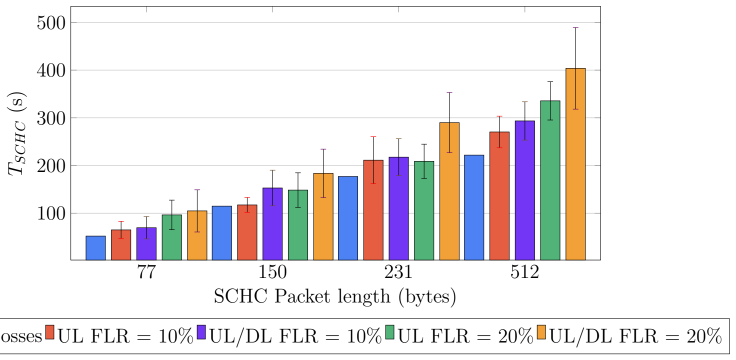


Figure 6.7: Experimental SCHC Packet transfer time: results obtained in RC4.

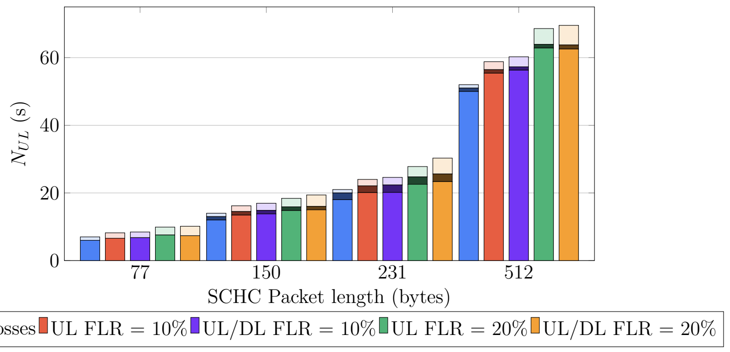


Figure 6.8: Number of UL messages: experimental results, Sigfox zones RC1 and RC4. The solid colored part, the darker colored part, and the lighter colored part represent the Regular SCHC Fragments, All-0 SCHC Fragments, and All-1 SCHC Fragments, respectively.

the DL FLR also increases the number of SCHC ACK retransmissions.

Another observation from Fig. 6.9 is that, in the presence of losses, N_{DL} increases with $N_{Windows}$ (which is also related to the SCHC Packet size). This is due to the high probability that the transmission of a SCHC Packet causes the loss of Regular SCHC Fragments, leading to a large number of SCHC ACKs.

We also noted that errors in the first windows (for SCHC Packet sizes greater than one window) have a greater impact on performance than errors in the last window, which increases the SCHC Packet transfer time and number of messages required. This is because the last window is unconditionally acknowledged, while fragment losses within intermediate windows generate additional SCHC ACKs. Therefore, the location of the SCHC Fragment losses directly affects SCHC Packet transfer performance.

One possible improvement to reduce N_{DL} may be reporting losses from several intermediate windows in a single extended ACK. In turn, N_{UL} would also decrease, as less All-1 SCHC Fragments would be required.

6.5.6 Results applicability and implementation considerations

The results provided in this section are useful for application design, since they allow to verify the feasibility of running a given application using SCHC over Sigfox considering SCHC Packet size, FLR, and the limitations in message rate and *duty-cycle* (see Section 2.2.1.2). For example, an

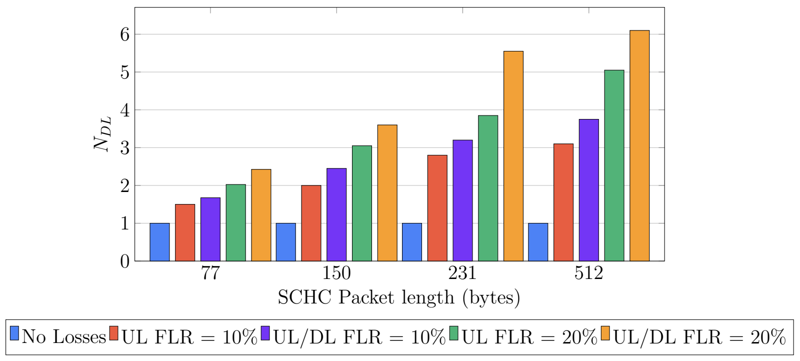


Figure 6.9: Number of DL messages, experimental results: Sigfox zones RC1 and RC4.

application that requires sending a 77-byte SCHC Packet every 120 minutes may be feasible in RC1, and with *duty-cycle* restrictions enforced, since $T_{SCHC_{DC}}$ would be 89 minutes (see Fig. 6.6), for UL/DL FLR of up to 10%. As another example, sending a SCHC Packet of 231 bytes for a UL FLR up to 20% requires slightly less than 4 DL messages (see Fig. 6.9). If only 4 DL messages per day are allowed, sending such a SCHC Packet per day will be possible. However, an additional non-zero DL FLR may lead N_{DL} to exceed the limit of 4 DL messages.

CHAPTER 7 ENERGY CONSUMPTION MODEL OF SCHC PACKET FRAGMENTATION

<<*Light generates matter to become visible.*>>

Most LPWAN devices are battery-powered, therefore it becomes fundamental to evaluate the energy performance of SCHC fragmented transmission. In this chapter, we propose an energy consumption model of SCHC Packet transfers over Sigfox, considering all states involved in preparing, fragmenting, sending and receiving. We evaluate two strategies which comply with *duty-cycle* restrictions, using deep and light sleep modes. Moreover, results for average current and energy consumption are obtained for single and periodic SCHC packet transfers and, for the latter, we derive the device lifetime depending on packet size and transfer period. The results from this chapter have been published in a journal article [IV].

7.1 Introduction

The SCHC F/R process is performed at the expense of contributing header and message overhead to packet transmission. Considering the energy constraints of IoT devices (many of which are not mains-powered), it is fundamental to evaluate the energy performance of SCHC fragmented packet transmission. However, to the best of our knowledge, there is no previous work in the literature on this topic [16, 57, 59–61, 105, 114–116]. In this chapter, we model and evaluate the energy performance of SCHC Packet fragmentation over Sigfox, a flagship LPWAN technology that supports a severely constrained maximum payload size (*i.e.*, 12 bytes) for IoT device packet transmission. Among others, our results quantify how the lifetime of a battery-operated device performing periodic packet transfers over Sigfox increases with the idle period between transfers and decreases with packet size. For example, assuming a battery capacity of 2000 mAh, and a period of 5 days, the device lifetime increases from 168 days (for a 2250-byte packet) to 1464 days (for a 77-byte packet). We also evaluate the impact on performance of different fragment transmission strategies, as well as device hardware features.

The rest of the chapter is organized as follows: Section 7.2 presents related work. Section 7.3 provides our current consumption model of SCHC Packet fragmentation over Sigfox. In Section 7.4, the model is used to evaluate the energy performance of SCHC Packet fragmentation over Sigfox.

7.2 Related work

This section reviews related work. First, we focus on literature regarding the energy performance of Sigfox. Secondly, we evaluation studies related to SCHC F/R.

7.2.1 Sigfox energy performance

The energy performance of the Sigfox technology has been a topic of interest for the research community. Martinez et al. [117] provided a general energy consumption model for IoT devices, including Sigfox devices, but did not provide information regarding the considered Sigfox module, nor device lifetime. In [20], authors provided an analytical model that characterizes Sigfox in terms of device current consumption, device lifetime, and energy cost of data delivery. Their results show that, using a MKRFOX1200 development kit with an ATA8520 Sigfox module, with a battery of 2400 mAh, a theoretical device lifetime of 1.5 years is possible when sending one message every 10 min at 100 bit/s. In [118], authors evaluated theoretically the energy consumption of different LPWANs for over-the-air updates. Results show that full firmware updates consume a significant amount of energy, especially for low bit rate technologies such as Sigfox. The datasheet values of an Onsemi AXSF Sigfox module were considered in their evaluation.

Hernandez et. al. [119] performed extensive energy consumption measurements; their results show that IoT sensors using Sigfox technology can be autonomous during remarkably long periods of time, with a lifetime of up to 4 years, when sending a message every 60 min at 100 bit/s and operating on a 1000 mAh battery. The evaluation was based on a Telit LE51-868/DIP Sigfox module. Morin et. al. [120] compared different wireless technologies, such as Sigfox, LoRaWAN, and BLE in terms of device lifetime. It was found that a Sigfox device, running on two AAA batteries (of 1250 mAh at 1.5 V each) can achieve a lifetime of 25 years when sending 10 bytes per day at 100 bit/s. Ogawa et al. [121] estimated the energy cost of Sigfox transmissions using the TD1207R/08R module.

An IoT solution for art conservation using Sigfox was presented in [122] and showed that if a device with a 1700 mAh battery is sending one sample per hour, it is possible to achieve a lifetime of 1.5 years, using the Telit LE51-868S Sigfox module. Moreover, the authors indicate that aggregation strategies (*i.e.*, sending more than one sample per transmission) can improve energy consumption and extend the node lifetime to 5 years. Authors in [123] concluded that, in cases where extremely long range is required, Sigfox has better device battery lifetime for small daily throughputs than other LPWAN technologies. Their theoretical evaluation was based on the AX-Sigfox module. Similar results are presented in [124]. In [125], Lykov et al. studied Sigfox using the AX-SIP-SFEU radio module. The authors found that, for a battery capacity of 2000 mAh, a payload size increase will reduce the device lifetime by up to 18 days, and that a daily message rate increase (up to 140 messages/day) can reduce the device lifetime down to 209 days.

Table 7.1 presents a summary of published work that evaluates Sigfox energy performance. None of these studies considered the energy consumption of packet fragmentation over Sigfox.

7.2.2 SCHC F/R

Authors in [16] showed that SCHC Packet fragmentation can increase reliability, with a trade-off in terms of energy consumption and goodput. The same authors analyzed in [61] the use of SCHC Packet fragmentation to reduce network congestion and increase network capacity. An overview and a simple evaluation showing the header and message overhead of SCHC F/R is presented in [100]. Other performance metrics, such as total channel occupancy, goodput and total delay were studied in [57] over an ideal channel. Optimal configuration values for SCHC F/R over LoRaWAN and Sigfox were provided in [59]. SCHC RFTs and alternative RFTs were presented and evaluated in [105] over LoRaWAN, as part of a reliable fragmentation method. Results show that alternative RFTs may be optimal depending on the error rate and pattern, providing greater efficiency.

Since the publication of the base SCHC specification [6], the IETF LPWAN WG has been developing SCHC Profiles, which provide configurations of SCHC F/R functionality tailored to specific LPWAN technologies such as Sigfox [7], LoRaWAN [9], and NB-IoT [8]. Sanchez-Gomez et al. [60] presented an evaluation of the LoRaWAN SCHC Profile in a real testbed. SCHC F/R

Table 7.1: List of references that evaluate Sigfox energy performance.

Reference	Sigfox Module	Battery Capacity (mAh)	Sending Period	Lifetime (Years)	Packet Fragmentation
[117]	Not specified	Not specified	Not specified	Not specified	No
[20]	ATA8520 Sigfox 2400	2400	10 min	1.5 (at 600 bit/s) 2.5 (at 100 bit/s)	No
[118]	Onsemi AXSF	2400	Not specified	Not specified	No
[119]	Telit LE51-868/DIP	1000	60 min	4 (at 600 bit/s)	No
[120]	TD1202	1500×2	24 h	25 (at 100 bit/s)	No
[121]	TD1207R/08R	Not specified	Not specified	Not specified	No
[122]	Telit LE51-868S	1700	60 min	1.5 (at 100 bit/s)	No
[123]	AX-Sigfox	1500	10 min	1 (at 100 bit/s)	No
[124]	Not specified	Not specified	Not specified	Not specified	No
[125]	AX-SIP-SFEU	2000	10 min	0.57 (at 100 bit/s)	No

provided benefits in terms packet delivery ratio, with a processing time overhead below 8 ms and a memory usage of only 609 bytes. Santa et al. [114] used SCHC to support IPv6 over LoRaWAN and NB-IoT for personal mobility vehicles. NB-IoT showed lower latency and low fragment error rate; however, it consumed more power than LoRaWAN. In [115], SCHC fragmentation over Sigfox was overviewed and evaluated theoretically and empirically by using a LoPy4 module, in terms of transfer time and number of Sigfox uplink and downlink messages. Muñoz et al. [116] evaluated SCHC over LoRaWAN and obtained a model to determine channel occupancy efficiency based on LoRaWAN and SCHC configuration parameters.

Table 7.2 summarizes the work related to SCHC F/R performance evaluation, along with the performance parameters and the methods used. Together, these studies provide important insights into SCHC F/R. However, they neither provide a detailed model of, nor evaluate, the current consumption or the energy performance of SCHC Packet transfer.

To the best of our knowledge, no previous work provides a current or energy consumption model nor evaluates the energy performance of fragmented packet transfer with SCHC.

7.3 Modeling SCHC F/R over Sigfox current consumption

In this section, we present models of crucial energy performance parameters of SCHC F/R over Sigfox, *i.e.*, device current consumption, device lifetime, and energy cost. We assume a Sigfox device that sends SCHC Packets to the Sigfox network. Such behavior may correspond to an IoT device sending sensor readings.

We first introduce the experimental setup used to perform current consumption measurements on a real device. Second, we identify the different states of a device that performs reliable SCHC Packet transfer by using ACK-on-Error over Sigfox, to obtain their corresponding current and energy consumption profiles. Finally, we model the current and energy consumption of fragmented SCHC Packet transfers considering single and periodic transfers. For the latter, we also model the lifetime of a battery-operated device.

Table 7.2: List of references related to SCHC F/R performance evaluation.

Reference	Performance Parameters	Method	Energy Performance Evaluation
[16]	Overhead, throughput, goodput, end to end delay	Simulation	No
[61]	Goodput, application capacity, efficiency, header overhead	Simulation	No
[100]	Header compression, number of fragments, number of ACKs	Theoretical	No
[57]	Channel occupancy, goodput, total delay	Simulation	No
[59]	ACK message overhead, ACK bit overhead with and without L2 headers	Theoretical	No
[105]	Error rates and patterns	Simulation	No
[60]	Packet delivery ratio, goodput per ToA	Experimental	No
[114]	Network delay, SNR, power consumption	Experimental	No
[115]	Transfer time, number of uplink and downlink messages	Theoretical, Experimental	No
[116]	Channel occupancy efficiency	Theoretical, Experimental	No

7.3.1 Experimental setup

Our models are derived from current consumption measurements on a real Sigfox device: a Pycom LoPy4 development board [108]. Fig. 7.1 shows the experimental setup, which includes an Agilent N6750A power analyzer and the Sigfox device. The experiments were carried out in an indoor environment in the city of Castelldefels, in Spain. The Sigfox coverage in the scenario is near-ideal, with negligible frame loss rate.

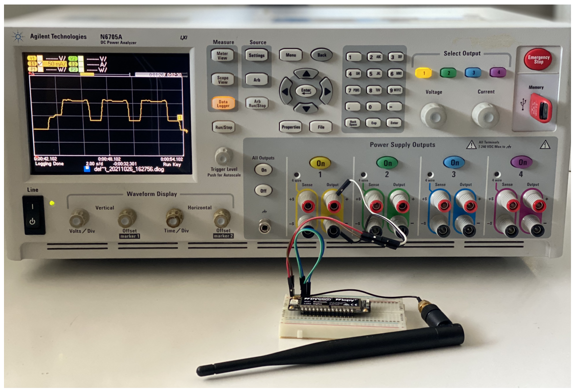


Figure 7.1: Experimental setup with the Sigfox device and the power analyzer used.

The LoPy4 module is based on the Espresiff ESP32 MCU. The latter includes a Wi-Fi and a Bluetooth interface, along with a Sigfox Semtech SX1276 radio module. Note that the LoPy4 module also has a built-in RGB LED. In our measurements, the LoPy4 board was programmed to enable the Sigfox radio interface and shut down other radio modules and peripherals (including the Wi-Fi and BLE interfaces and the RGB LED) on boot.

The LoPy4 has a voltage regulator, which supports input voltages between 3.5 V and 5.5 V. The output voltage of the regulator is 3.3 V. In all measurements performed, the supplied voltage is 3.5 V. The Sigfox radio of the LoPy4 board is configured for RC1. Accordingly, the UL data rate is 100 bps, and the DL data rate is 600 bps. The transmit power is +14 dBm. The receiver sensitivity is -126 dBm. The SCHC over Sigfox implementation used in our evaluation is based on the one presented in [115], which is publicly available.

7.3.2 SCHC Packet transfer states

In order to comply with the *duty-cycle* constraints in RC1, SCHC Fragments may be sent by using different approaches. In our model, we consider two possible options: (i) sending one SCHC Fragment per cycle of 10 min (and sleeping otherwise), and (ii) sending up to 6 SCHC Fragments back to back per cycle of 60 min (and sleeping otherwise).

Let N_{pC} denote the number of SCHC Fragments that are sent back-to-back per cycle, where $1 \leq N_{pC} \leq 6$. Let N_C denote the number of cycles required to complete a SCHC Packet transfer.

Each cycle comprises several states (see Fig. 7.2). Initially, the device is sleeping (Sleep state), and then, the device wakes up (Wake-up state). If a new SCHC Packet needs to be sent, the device enters the Fragmenter state, where SCHC Packet fragmentation is performed. In this state, the device creates the SCHC Fragments from the SCHC Packet, which includes selecting the appropriate RuleID (according to the SCHC Packet size) and the corresponding FCN and W values for each SCHC Fragment.

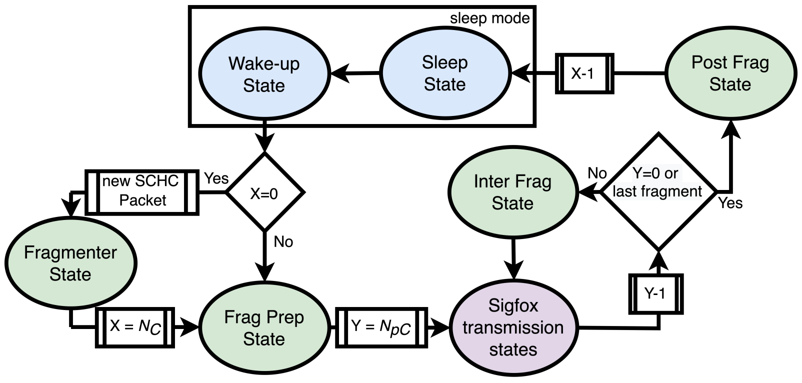


Figure 7.2: Fragmented SCHC Packet transfer state diagram. Sleep and Wake-up states, SCHC Fragmentation-related states, and Sigfox transmission states are depicted in blue, green, and purple, respectively. X and Y variables correspond to the number of cycles (N_C) and to the number of SCHC Fragments per cycle (N_{pC}), respectively.

After the Fragmenter state or after the Wake-up state if the device continues sending an already fragmented SCHC Packet, the device reaches the Frag Prep state, where it prepares the next SCHC Fragment to be sent and selects the Sigfox transmission procedure to be used for this SCHC Fragment. The prepared SCHC Fragment is then sent accordingly (the device is in the Sigfox transmission state). When more than one SCHC Fragment is sent per cycle $N_{pC} \geq 2$), the device enters the Inter Frag state to prepare the next SCHC Fragment to be transmitted or to process a SCHC ACK (when available).

Finally, after sending all the SCHC Fragments of a cycle, or after sending the last SCHC Fragment of a SCHC Packet, the device reaches the Post Frag state. In this state, the device processes a SCHC ACK (when available) and returns to the Sleep State. Sleep state time will depend on N_{pC}. Next, we characterize the device current consumption in the states involved in each cycle.

7.3.3 Current consumption profile

In this section, we present the current consumption profile of all the states involved in a fragmented SCHC Packet transfer, which have been introduced in Section 7.3.2. These current consumption profiles are obtained by using the experimental setup shown in Section 7.3.1. All individual results provided correspond to the average of 10 experiments. For a given scenario and set of configuration parameters, we found negligible differences among the individual results.

7.3.3.1 Sleep and Wake-up states current consumption profile

Most Sigfox devices are battery-powered. Therefore, to improve battery lifetime, they must remain in Sleep state most of the time, and only wake up for communication. The LoPy4 supports two sleep modes: the light sleep mode and the deep sleep mode. In the light sleep mode, most peripherals and CPU are clock-gated, and voltage consumption is reduced, which allows for a reduced wake-up time. In the deep sleep mode, the CPU and all peripherals are stopped, which reduces the current consumption to the minimum but increases wake-up time.

The Wake-up state current consumption and duration depends on the sleep mode used. Tables 1 and 2 present the Wake-up state duration and current consumption, and the Sleep state current consumption, for the light sleep mode and the deep sleep mode, respectively.

As shown in Tables 7.3 and 7.4, there is a large difference between Wake-up state and Sleep state time and current consumption for light and deep sleep modes. The light sleep mode has a shorter Wake-up state time but a greater sleep current. The corresponding average energy consumption is illustrated in Fig. 7.3. For short sleep periods, light sleep is more efficient energywise, as the Wake-up state time is shorter. For long sleep intervals, deep sleep becomes more efficient, since the longer Wake-up state duration is compensated by the ultralow deep sleep current consumption in the Sleep state.

Table 7.3: Wake-up and Sleep states characterization for the light sleep mode.

States	Duration Notation	Duration (ms)	Average current consumption notation	Average Current Consumption (mA)	Average Energy Consumption (mJ)
Wake-up	$T_{Wake-up}$	20	$I_{Wake-up}$	42	2.94
Sleep	T_{Sleep}	-	I_{Sleep}	2.07	-

7.3.3.2 SCHC Fragmentation states current consumption profile

Table 7.5 presents the SCHC fragmentation states duration and their corresponding current consumption.

In contrast with the durations of the Frag Prep, Inter Frag, and Post Frag states, which are constant, the Fragmenter state duration is proportional to the SCHC Packet size. Fig. 7.4

Table 7.4: Wake-up and Sleep states characterization for the deep sleep mode.

States	Duration Notation	Duration (ms)	Average current consumption notation	Average Current Consumption (mA)	Average Energy Consumption (mJ)
Wake-up	$T_{Wake-up}$	2770	$I_{Wake-up}$	52.4	508.02
Sleep	T_{Sleep}	-	I_{Sleep}	0.02	-

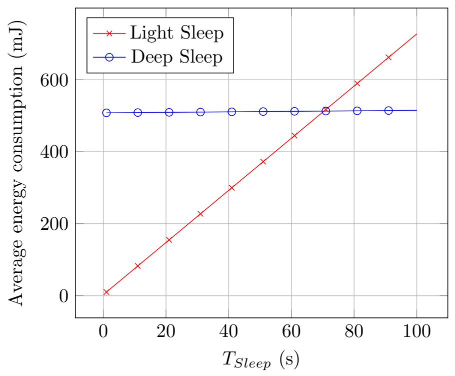


Figure 7.3: Average energy consumption of Wake-up and Sleep states for different T_{Sleep} values, and for light and deep sleep mode.

Table 7.5: SCHC Fragmentation states.

States	Duration Notation	Duration (ms)	Average current consumption notation	Average Current Consumption (mA)
Fragmenter	T_{Frag}	see Fig. 7.4	I_{Frag}	55.3
Frag Prep	T_{Prep}	23.26	I_{Prep}	55.3
Inter Frag	T_{Inter}	19.07	I_{Inter}	55.3
Post Frag	T_{Post}	28.74	I_{Post}	55.3

illustrates the Fragmenter state duration, as a function of the SCHC Packet size, for SCHC Packet sizes between 1 and 2250 bytes. For small SCHC Packet sizes, the impact of the fragmentation process on time is negligible. However, as SCHC Packet size increases, the Fragmenter state duration becomes more significant (up to 3.54 s for a SCHC Packet size of 2250 bytes). Note that the Fragmenter state is only present once in each SCHC Packet transfer.

7.3.3.3 U-procedure current consumption profile

In the Frag Prep state or in the Inter Frag state, the following SCHC Fragment is prepared to be transmitted by using one of the two Sigfox procedures (*i.e.*, U-procedure or B-procedure), depending on the SCHC Fragment type (*i.e.*, Regular (not All-0), All-0 or All-1). If the SCHC Fragment is a Regular (not All-0) SCHC Fragment, the U-procedure is selected. Fig. 7.5 shows the U-procedure current consumption profile, as measured on the LoPy4.

The U-procedure comprises three substates: Transmission (Substate 1), Wait next transmission (Substate 2), and Cooldown (Substate 3). Table 7.6 presents the duration and current consumption

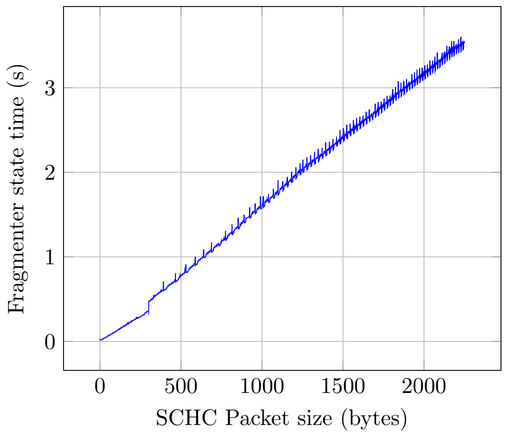


Figure 7.4: Fragmenter state time as a function of the SCHC Packet size.

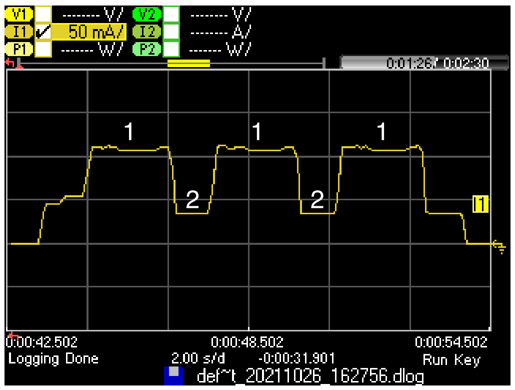


Figure 7.5: Current consumption profile of a LoPy4 device performing a U-procedure. In this measurement, the Sigfox UL frame payload size is 12 bytes, equivalent to a Regular (not All-0) SCHC Fragment carrying one tile.

of these substates along with their notations. Substate 1 is repeated three times, as the UL frame is sent by using three different frequencies. Substate 2 is present twice, between two consecutive transmissions. After sending the UL frame, the Sigfox radio module enters Substate 3 before transiting to the Inter Frag or Post Frag states, or before handling other processes.

Table 7.6: U-procedure substates and their corresponding duration and current consumption values.

Substate	Duration Notation	Duration (ms)	Average current consumption notation	Average Current Consumption (mA)
1. Transmission	T_{Tx}	[1120,2080]	I_{Tx}	112.9
2. Wait next transmission	T_{Wait_Tx}	1000	I_{Wait_Tx}	34.02
3. Cooldown	T_{Cool}	1000	I_{Cool}	33.98

The transmission current measured value, denoted I_{Tx}, is greater than the one presented in the LoPy4 datasheet [108], as it involves the MCU in addition to the Sigfox radio module, and the input voltage is different (our experiments are performed using 3.5 V, whereas datasheet values are provided for 5 V). Let I_{U-proc} denote the average current consumption of a U-Proc. Using the notation of Table 7.6, I_{U-proc} can be calculated as follows:

$$I_{U-proc}(mA) = \frac{3 \cdot I_{Tx} \cdot T_{Tx} + 2 \cdot I_{Wait\_Tx} \cdot T_{Wait\_Tx} + I_{Cool} \cdot T_{Cool}}{T_{U-proc}}, \tag{7.1}$$

where T_{U-proc} denotes the total $U-procedure$ duration and can be calculated as follows:

$$T_{U-proc}(s) = 3 \cdot T_{Tx} + 2 \cdot T_{Wait\_Tx} + T_{Cool} \tag{7.2}$$

7.3.3.4 B-procedure current consumption profile

All-0 and All-1 SCHC Fragments need to open a DL reception window to offer the SCHC receiver the opportunity to transmit a SCHC ACK. To this end, the transmission of such fragments is performed by using a B-procedure. Fig. 7.6 shows the B-procedure current consumption profile, as measured on the LoPy4 module. The B-procedure comprises six substates: Transmission (substate 1), Wait next transmission (substate 2), Wait for reception (substate 4), Reception (substate 5), Confirmation (substate 6), and Cooldown (substate 3). Table 7.7 presents the measured duration and current consumption for each substate of the B-procedure.

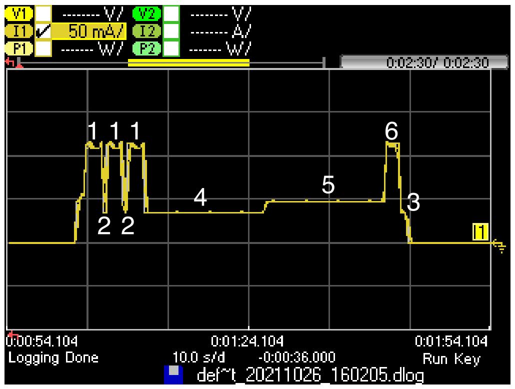


Figure 7.6: Current consumption profile of a LoPy4 in a B-procedure, with an UL frame payload of 12 bytes, equivalent to an All-0 SCHC Fragment or an All-1 SCHC Fragment carrying one tile. A DL frame is received by the device, which subsequently sends a confirmation control frame.

In a similar way to a U-procedure, the UL frame in a B-procedure is transmitted by using 3 different frequency channels; therefore, substate 1 is present three times, and the substate 2 is present twice, between transmissions. After the third transmission for the UL frame, the Sigfox module waits for a fixed duration time interval in substate 4 and opens the reception window in substate 5. The duration of substate 5 depends on when the DL frame is received. After receiving the DL frame, the confirmation control frame is sent in substate 6. In substate 3, the device radio cools down before allowing the MCU to perform other operations. In case the Sigfox network and/or application does not send any DL frame to the device, or the device does not receive it, substate 6 is not present, and substate 5 duration is the maximum one (*i.e.*, T_{RXMAX}, which is equal to 25 s in RC1).

The average current consumption of a B-procedure when a DL frame is received by the device, denoted $I_{B-proc-DL}$, can be obtained as follows:

Table 7.7: B-procedure substates and their corresponding duration and current consumption values.

Substate	Duration Notation	Duration (ms)	Average current consumption notation	Average Current Consumption (mA)
1. Transmission	T_{Tx}	[1120,2080]	I_{Tx}	112.9
2. Wait next transmission	T_{Wait_Tx}	1000	I_{Wait_Tx}	34.02
4. Wait for reception	$TWaitRx$	15556	$IWait_{R}x$	34.14
5. Reception	T_{Rx}	[387,25000] *	I_{Rx}	45.94
6. Confirmation	T_{Conf}	1799	I_{Conf}	114.95
3. Cooldown	T_{Cool}	1000	I_{Cool}	33.98

* The value obtained in measurements and used in the evaluation is 15550 ms.

$$I_{B-proc-DL}(mA) = \frac{I_{Tx} \cdot T_{Tx} + 2 \cdot I_{Wait\_Tx} \cdot T_{Wait\_Tx} + I_{Wait\_Rx} \cdot T_{Wait\_Rx} + I_{Conf} \cdot T_{Conf} + I_{Cool} \cdot T_{Cool}}{T_{B-proc-DL}} \tag{7.3}$$

where $T_{B-proc-DL}$ can be calculated as follows:

$$T_{B-proc-DL}(s) = 3 \cdot T_{Tx} + 2 \cdot T_{Wait\_Tx} + T_{Wait\_Rx} + T_{Conf} + T_{Cool}. \tag{7.4}$$

The average current consumption of a B-procedure when a DL frame is not received by the device, denoted $I_{B-proc-NO-DL}$, can be obtained as follows:

$$I_{B-proc-NO-DL}(mA) = \frac{3 \cdot I_{Tx} \cdot T_{Tx} + 2 \cdot I_{Wait\_Tx} \cdot T_{Wait\_Tx} + I_{Wait\_Rx} \cdot T_{Wait\_Rx} + I_{Cool} \cdot T_{Cool}}{T_{B-proc-NO-DL}}, \tag{7.5}$$

where $T_{B-proc-NO-DL}$ can be obtained as follows:

$$T_{B-proc-NO-DL}(s) = 3 \cdot T_{Tx} + 2 \cdot T_{Wait_{T}x} + T_{Wait_{R}x} + T_{Cool}. \tag{7.6}$$

7.3.4 SCHC Packet transfer current and energy consumption model

In this subsection, we model the SCHC Packet transfer current and energy consumption over Sigfox. To this end, we first calculate the number of U-procedure and B-procedure required to transfer a SCHC Packet. Then, we derive a current and energy consumption model of SCHC Packet transfer in two cases: (i) single and (ii) periodic SCHC Packet transfers.

7.3.4.1 Number of U-procedure and B-procedure

The number of U-procedure (N_{U-proc}) required to transfer a SCHC Packet of size L_{SCHC} can be obtained as:

$$N_{U-proc} = \left\lceil \frac{L_{SCHC}}{L_{UL-L_{H}eader}} - \frac{L_{SCHC}}{(WINDOW\_SIZE * t)} \right\rceil, \tag{7.7}$$

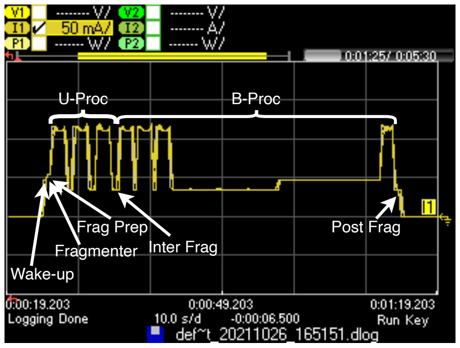
where L_{UL} is the maximum Sigfox UL frame payload size of 12 bytes, and L_{Header} is the size of the SCHC Fragment header.

The number of B-procedure with a DL frame ($N_{B-proc-DL}$) required to transfer a SCHC Packet without fragment losses is equal to 1. Under such conditions, the number of B-procedure with no DL ($N_{B-proc-NO-DL}$) can be obtained as follows:

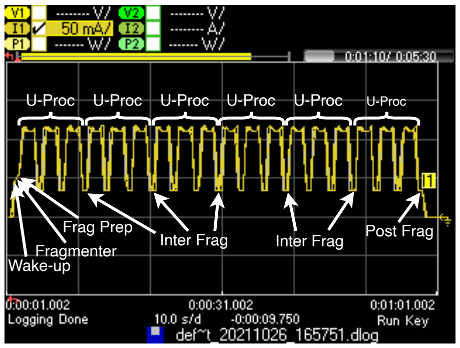
$$N_{B-proc-NO-DL} = \left\lceil \frac{L_{SCHC}}{WINDOW_{S}IZE \cdot t} - 1 \right\rceil. \tag{7.8}$$

7.3.4.2 Single SCHC Packet transfer model

The number of cycles required to transfer a single SCHC Packet (NC) depends on the fragment sending strategy, *i.e.*, on the N_{pC} value. Fig. 7.7 illustrates the current consumption of (a) a 22-byte SCHC Packet transfer for $N_C = 1$ and $N_{pC} = 2$ and (b) the first transfer cycle of a 77-byte SCHC Packet for $N_C = 2$ and $N_{pC} = 6$. The figure shows that the number of Inter Frag states increases with N_{pC}.



(a)



(b)

Figure 7.7: Current consumption of two SCHC Packet transfer examples: (a) a 22-byte SCHC Packet is sent completely and a SCHC ACK is received; (b) six SCHC Fragments are sent back-to-back before the device returns to the Sleep state in the first transfer cycle of a 77-byte SCHC Packet.

Note that N_C is related to the number of Wake-up and Frag Prep and Post Frag states required to complete the SCHC Packet transfer. By sending up to 6 SCHC Fragments back-to-back (*i.e.*, $N_{pC} \leq 6$), the number of Wake-up, Frag Prep, and Post Frag states is minimized, when compared to $N_{pC} = 1$.

Once the sending strategy is selected, *i.e.*, the N_{pC} value is chosen, N_C can be calculated as follows:

$$N_C = \left\lceil \frac{N_{U-proc} + N_{B-proc-NO-DL} + N_{B-proc-DL}}{N_{pC}} \right\rceil . \tag{7.9}$$

The number of Wake-up, Frag Prep, and Post Frag states (denoted $N_{Wake-up}$, N_{Prep}, and N_{Post}) is equal to N_C, as each time the device transmits one or several back-to-back SCHC Fragments, it must wake up, prepare the next SCHC Fragment, and then do the SCHC Fragment post processing in the Post Frag state before returning to the Sleep state. We define the SCHC Packet active time (T_{act}) as the time the device is not in the Sleep state. T_{act} can be obtained as follows:

$$\begin{aligned} T_{act}(s) = & T_{Frag} + N_C \cdot (T_{wake-up} + T_{Prep} + T_{Post} + T_{Inter} \cdot (N_{pC} - 1)) + \\ & N_{U-proc} \cdot T_{U-proc} + N_{B-proc-NO-DL} \cdot T_{B-proc-NO-DL} + T_{B-proc-DL} \end{aligned} \tag{7.10}$$

The SCHC Packet active time current consumption (I_{act}) can be calculated as follows:

$$\begin{aligned} I_{act}(mA) = & \frac{1}{T_{act}} (I_{Frag} \cdot T_{Frag} + N_C \cdot (I_{wake-up} \cdot T_{wake-up} + I_{Prep} \cdot T_{Prep} + \\ & I_{Post} \cdot T_{Post} + I_{Inter} \cdot T_{Inter} \cdot (N_{pC} - 1)) + N_{U-proc} \cdot I_{U-proc} \cdot T_{U-proc} + \\ & N_{B-proc-NO-DL} \cdot I_{B-proc-NO-DL} \cdot T_{B-proc-NO-DL} + \\ & I_{B-proc-DL} \cdot T_{B-proc-DL}). \end{aligned} \tag{7.11}$$

As explained in Section 2.2.1.2, in RC1, a transmission procedure can only be started, at least, every 600 s, denoted $T_{perProc}$. Therefore, T_{act} is only a small fraction of the total SCHC Packet transfer time (T_{SCHC}). This latter can be obtained as follows:

$$T_{SCHC} = (N_{U-proc} + N_{B-proc-DL} + N_{B-proc-NO-DL}) \cdot T_{perProc}. \tag{7.12}$$

Note that T_{SCHC} is independent of N_C, since the same total wait time has to be enforced, regardless of whether up to 6 messages are sent back-to-back per cycle ($N_{pC} \leq 6$) or one message is sent per cycle ($N_{pC} = 1$). The effect of N_C are reflected in T_{act}. Therefore, the amount of time that the device is required to be in the Sleep state to comply with *duty-cycle* restrictions for the transfer of a SCHC Packet, denoted T_{Sleep}, can be calculated as follows:

$$T_{Sleep} = T_{SCHC} - T_{act}. \tag{7.13}$$

Finally, the average current consumption of a SCHC Packet transfer over Sigfox (I_{SCHC}) can be calculated as follows:

$$I_{SCHC} = \frac{I_{act} \cdot T_{act} + T_{Sleep} \cdot I_{Sleep}}{T_{SCHC}}. \tag{7.14}$$

In addition, the average energy consumed in a SCHC Packet transfer can be determined as:

$$E_{SCHC} = I_{SCHC} \cdot V \cdot T_{SCHC}, \tag{7.15}$$

where V denotes the voltage supplied to the Sigfox device.

7.3.5 Periodic SCHC Packet transfer energy performance metrics

This subsection presents the metrics used to evaluate the energy performance of SCHC over Sigfox, for a device that transfers a SCHC Packet periodically. These metrics are (i) the average current consumption, (ii) the SCHC Packet transfer energy cost, and (iii) the device lifetime. We assume that the device starts a SCHC Packet transfer (by sending the first fragment) every time period T_p. Note that the minimum possible T_p value, denoted T_{p_min}, should be equal to T_{SCHC}. After a SCHC Packet transfer, the device will wait in the Sleep state for T_{Wait} until T_p time has elapse since the start of the previous SCHC Packet transfer. T_p can be calculated as follows:

$$T_p = T_{SCHC} + T_{Wait}. \tag{7.16}$$

During the wait period between SCHC Packet transfers, the device is in the Sleep state, consuming a current of I_{Sleep}. Otherwise, the device transfers a SCHC Packet, with an average current consumption of I_{SCHC}. In consequence, the average current consumption of periodic SCHC Packet transfers (I_p) can be obtained as follows:

$$I_p = \frac{I_{SCHC} \cdot T_{SCHC} + I_{Sleep} \cdot T_{Wait}}{T_p}. \tag{7.17}$$

The energy consumed by a device performing periodic SCHC Packet transfers over an interval of duration T_p can be obtained as follows:

$$E_p = I_p \cdot T_p \cdot V. \tag{7.18}$$

Sigfox devices are commonly battery-operated, and therefore, device lifetime calculation is crucial to the performance of SCHC Packet transfer over Sigfox. In order to calculate the device lifetime, the battery capacity must also be taken into consideration. Let C_p denote the battery capacity (typically expressed in mAh). The device lifetime, LT, can be calculated as follows:

$$LT = \frac{C_p}{I_p}. \tag{7.19}$$

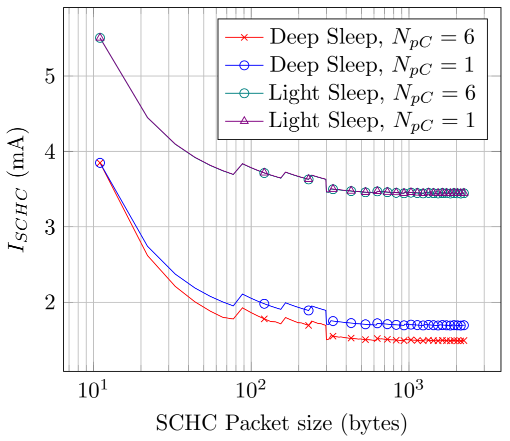


Figure 7.8: Average current consumption of a SCHC Packet transfer over Sigfox.

7.4 Evaluation

In this section, we evaluate energy-related performance parameters for single and periodic SCHC Packet transfers over Sigfox. First, we present the SCHC Packet current consumption and energy cost, for light and deep sleep modes, for different sending strategies. Then, we evaluate periodic SCHC Packet transfers in terms of current consumption, energy cost and device lifetime.

7.4.1 SCHC Packet current and energy consumption

Fig. 7.8 depicts I_{SCHC} for SCHC Packet sizes between 11 and 2250 bytes, for deep sleep and light sleep, and for N_{pC} values equal to 1 and 6. I_{SCHC} values are obtained by using (7.14). As SCHC Packet size increases, T_{Sleep} increases as well due to *duty-cycle* restrictions. In consequence, I_{SCHC} decreases since the device remains in sleep mode for a greater percentage of time (with a sleep current of 40 µA for deep sleep and 42 mA for light sleep).

Note that for small SCHC Packet sizes, the sleep time versus active time ratio increases rapidly with SCHC Packet size. Such ratio is only 14 for an 11-byte SCHC Packet, while it increases to 52 for a 350-byte SCHC Packet. As the SCHC Packet size increases beyond 350 bytes, the same ratio tends asymptotically to a value of 54. Therefore, I_{SCHC} becomes stable between 1.36 mA and 1.32 mA, for the deep sleep mode, and equal to 3.44 mA for the light sleep mode. The I_{SCHC} stepwise behavior for small SCHC Packet sizes is due to the additional windows needed to perform the SCHC Packet transfer (which increases current consumption due to the corresponding additional B-procedure). The larger step with a SCHC Packet size of 300 bytes is due to the change from a 1-byte to a 2-byte SCHC header at that value.

Fig. 7.9 illustrates the energy consumed by a device to perform a SCHC Packet transfer, for SCHC Packet sizes between 11 and 2250 bytes. The depicted values are obtained by using (7.15). The energy consumption increases linearly with SCHC Packet size. Despite the fact that the average current consumption of a SCHC Packet transfer is relatively constant for SCHC Packet sizes beyond 350 bytes, the increase of SCHC Packet transfer duration with SCHC Packet size is reflected as a SCHC Packet transfer energy consumption increase.

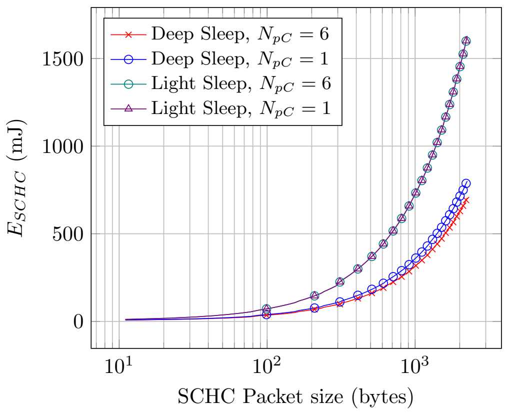


Figure 7.9: Energy consumed by a device to perform a SCHC Packet transfer over Sigfox.

7.4.2 Periodic SCHC Packet transfer energy performance

Table 7.8 presents the SCHC Packet sizes used for the periodic SCHC Packet transfer energy performance evaluation, along with the corresponding values of N_{U-proc}, $N_{B-proc-NO-DL}$, and the number of windows required for a single SCHC Packet transfer. The considered SCHC Packet sizes allow to test different values for N_{U-proc}, $N_{B-proc-NO-DL}$, as well as number of windows, for single-byte and two-byte SCHC header sizes. Table 7.8 also provides the T_{p_min} value for each SCHC Packet size, and the number of SCHC Packets per day that can be transferred with a SCHC Packet sending period equal to T_{p_min}.

Table 7.8: SCHC Packet sizes used in the energy performance evaluation.

SCHC Packet Size (Bytes)	N_{U-proc}	$N_{B-proc-NO-DL}$	$N_{B-proc-DL}$	Number of Windows	T_{p_min} (Minutes)	SCHC Packets per Day with T_{p_min}
77	6	0	1	1	70	20
154	12	1	1	2	140	10
275	21	3	1	4	250	5
510	49	1	1	2	510	2
2250	217	7	1	8	2250	0.64 *

* Requires more than one day for a packet transfer.

Fig. 7.10 illustrates the average current consumption of a device that performs periodic SCHC Packet transfers, I_p, for different SCHC Packet sizes and N_{pC} values of 1 and 6. We only consider the deep sleep mode since it is more energy-efficient than the light sleep mode. The depicted values are obtained by using (7.17). Note that all curves do not start at the same T_p value since the minimum T_p (T_{p_min}) value is equal to T_{SCHC} and depends on the SCHC Packet size (see Table 7.8). As T_p increases, I_p decreases for all SCHC Packet sizes, since T_{Sleep} increases, reducing the average current consumption. As shown in Fig. 7.10, for a given SCHC Packet size, $N_{pC} = 1$ consumes a higher of current than $N_{pC} = 6$ since with the latter, the number of Wake-up, Frag Prep, and Post Frag states (and thus, their contribution to current consumption) is reduced. As the SCHC Packet size increases, the average current consumption gap between the considered N_{pC} values increase.

Fig. 7.11 illustrates the energy consumption of a SCHC Packet transfer over a period T_p for different SCHC Packet sizes, N_{pC} values of 1 and 6, and with the deep sleep mode. The depicted values are obtained by using (7.18). This performance parameter increases linearly with T_p. This

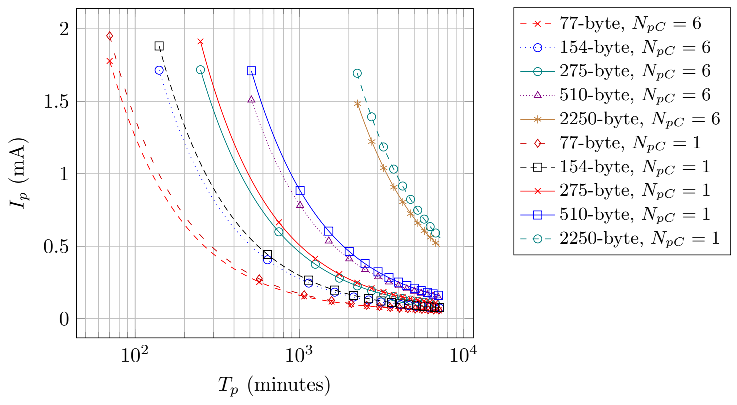


Figure 7.10: Average current consumption of a device performing periodic SCHC Packet transfers over Sigfox, for different T_p, N_{pC}, and SCHC Packet size values.

increase is small, since as T_p increases, the device remains in sleep mode for a greater amount of time, which increases energy consumption, albeit to a small extent.

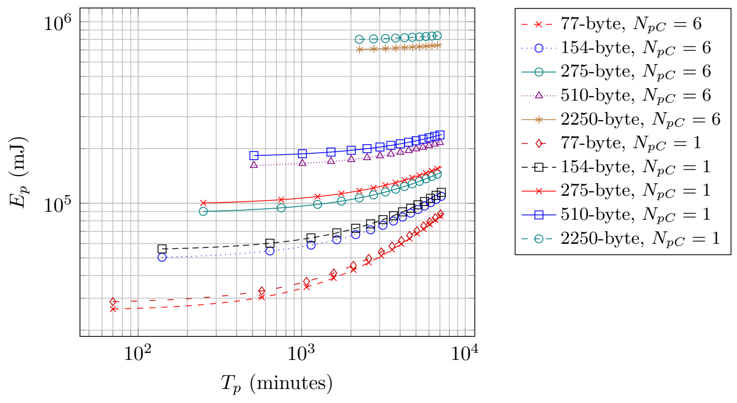


Figure 7.11: Energy consumption of a SCHC Packet transfer over a period T_p, for different T_p, N_{pC}, and SCHC Packet size values.

Finally, Fig. 7.12 shows the results obtained by using (7.19) regarding the lifetime of a device that performs periodic SCHC Packet transfers for different SCHC Packet sizes, for N_{pC} values of 1 and 6, and for the deep sleep mode. The battery capacity is 2000 mAh. Recall that $T_{p\_min} = T_{SCHC}$. T_{p_min} ranges from 70 min for a 77-byte SCHC Packet to 2250 min for a 2250-byte SCHC Packet (see Table 7.8).

For the corresponding T_{p_min} and $N_{pC} = 1$, the device lifetime yields the smallest values, *i.e.*, 42 days for a 77-byte SCHC Packet size, and 49 days for a 2250-byte SCHC Packet size. Note that there are large differences between the Tp_min value for specific SCHC Packet sizes, which in turn increase device lifetime, as the device spends more time in Sleep mode and is involved in a lower number of Fragmenter states. Indeed, for a fixed T_p value, as SCHC Packet size increases, more U-procedures and B-procedures are required, with the corresponding energy consumption increase and device lifetime decrease.

On the other hand, device lifetime increases asymptotically with T_p. For a T_p value of 5 days and for $N_{pC} = 6$, and for a 77-byte SCHC Packet size, the device lifetime is 1464 days (*i.e.*, more

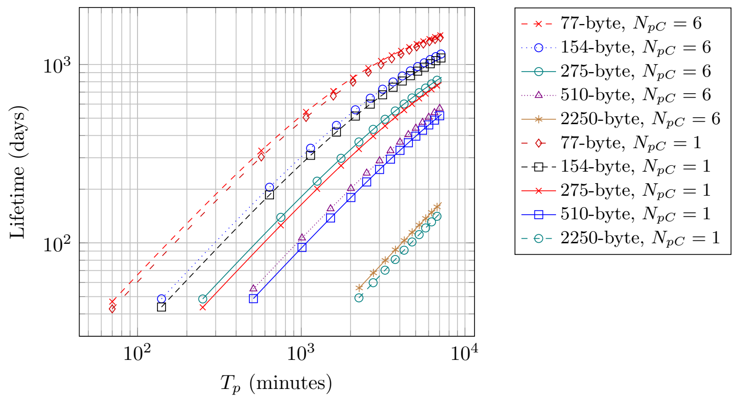


Figure 7.12: Lifetime of a battery-operated device performing periodic SCHC Packet transfers over Sigfox, for different T_p, N_{pC}, and SCHC Packet size values.

than 4 years). For the same T_p and N_{pC} values, and for a 2250-byte SCHC Packet size, the device lifetime is 168 days. The device lifetime gaps for $N_{pC} = 1$ and $N_{pC} = 6$ decrease with SCHC Packet size, due to the consequent increase of sleep time during the SCHC Packet transfer, reducing the impact of the time spent in the Wake-up, Frag Prep, and Post Frag states. For the 77-byte SCHC Packet size, such difference ranges from 4 days ($T_{p\_min} = 70$ min) to 42 days ($T_p = 5$ days), whereas for the 2250-byte SCHC Packet size, the differences range from 6 days ($T_{p\_min} = 2250$ min) to 19 days ($T_p = 5$ days).

CHAPTER 8 IPV6 OVER CROSS-TECHNOLOGY COMMUNICATIONS WITH WAKE-UP RADIO USING SCHC

<<Light responds.>>

In this chapter, we present the design, implementation, and evaluation of an adaptation layer to provide IPv6 support on low rate channels over WuR Systems, by using Cross-Technology Communications with Wake-up Radio (WuR-CTC) and by leveraging the IETF SCHC framework. SCHC was built for LPWAN technologies, however, its unique features like its optimized header compression and its low overhead reliable F/R modes are applicable beyond the LPWAN landscape. WuR-CTC provides direct data exchange for devices with incompatible network technologies, such as IEEE 802.15.4 and IEEE 802.11 devices. By designing, implementing, and evaluating the SCHC over WuR-CTC Profile, we contribute to the IPv6 support over LRLPWN, providing a SCHC Profile definition for devices that support WuR-CTC. Our solution enables full Internet protocol stack interoperability between devices of different communication technologies, without the need for a gateway, contributing to the advancement of wireless convergence.

8.1 Introduction

The IoT is built with the idea that all devices and applications communicate with one another and with the Internet. According to published studies, the number of connected IoT devices as of 2022 is 14.4 billion [126]. Several wireless (and wired) technologies are playing a key role as IoT communication enablers. More specifically, two flagship wireless communication technologies, *i.e.*, IEEE 802.11 and IEEE 802.15.4, are outspread in the IoT ecosystem. On the one hand, IEEE 802.11 is being crucial to provide Internet connectivity to IoT devices[1], with an already deployed infrastructure of 628 million public APs [36]. On the other hand, IEEE 802.15.4 is widely used in smart infrastructures, such as smart homes, buildings, and factories. As a result, many scenarios comprise both IEEE 802.11 and IEEE 802.15.4 devices. For example, in smart home and industry deployments, coexisting IEEE 802.11 and IEEE 802.15.4 networks are common [127].

Despite the fact that IEEE 802.11 and IEEE 802.15.4 can share the same frequency band, they are not interoperable out-of-the-box, and a gateway is needed to interconnect them (see Fig. 8.1). This approach presents a number of drawbacks: i) the gateway may be a single point of failure, ii) signals are retransmitted by the gateway, thus reducing spectral efficiency and increasing energy consumption, and iii) the additional cost of the gateway device itself [128, 129].

To provide direct communication (*i.e.*, without a gateway) between IEEE 802.11 and 802.15.4 devices, research has focused on CTC. One approach, based on signal emulation, has offered better

[1]`https://www.wi-fi.org/discover-wi-fi/internet-of-things`, (accessed on 07/12/2022).

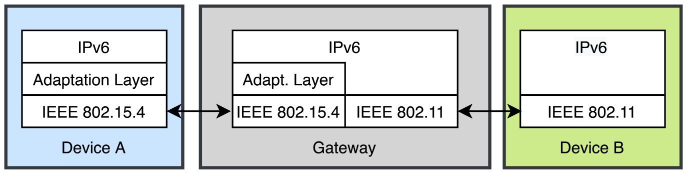


Figure 8.1: Gateway-based communication between Device A (IEEE 802.15.4 radio interface) and Device B (IEEE 802.11 radio interface).

results in terms of throughput and channel efficiency compared to other CTC methods [129], while providing bidirectional communication. Signal emulation CTC is based on reproducing the signals of one wireless technology with the transmitter of another wireless technology. Authors in [5, 129] proposed a signal emulation CTC method for direct communication between IEEE 802.11 and IEEE 802.15.4 devices called WuR-CTC. This method exploits a WuR channel. In [5], the WuR channel was originally designed to wake up devices being in sleep mode by using an external secondary low-power radio. In WuR-CTC [129], data transfer is performed via the WuR channel itself. To accomplish this, the sender device main radio encodes data using OOK. The secondary radio performs decoding at the receiver device.

WuR-CTC provides Physical Layer and Link Layer functionality with a maximum frame payload size, also known as MTU, of 89 bytes. In order to provide full protocol stack interoperability for WuR-CTC devices, it is necessary to support IPv6 over WuR-CTC. IPv6 provides natural Internet connectivity, a vast address space, and it allows to leverage IP-based protocols, security, and tools (see Fig. 8.2). However, the WuR-CTC MTU is smaller than the MTU required by IPv6 for its underlying layer, *i.e.*, 1280 bytes [2]. Therefore, IPv6 cannot run as is atop WuR-CTC.

In this chapter, and for the first time to our best knowledge, we present the design, implementation, and evaluation of an adaptation layer that enables IPv6 over WuR-CTC. Our solution leverages the header compression and fragmentation framework, called SCHC, which has been recently standardized by the IETF [6]. The header compression reduces communication overhead (decreasing both latency and energy consumption), whereas the fragmentation allows to transfer IPv6 packets larger than the WuR-CTC MTU. SCHC has been designed with a primary focus on LPWAN scenarios. However, in this chapter, we adapt SCHC to the WuR-CTC environment to allow IPv6 support, enabling full protocol stack interoperability between an IEEE 802.15.4 device and an IEEE 802.11 device, without a gateway [129].

We have implemented and evaluated our IPv6 over WuR-CTC solution in terms of IPv6 packet compression ratio, error rate, total number of frames and overhead required, transfer time, and throughput. Our results show that, by using SCHC, a 1280-byte IPv6 packet is transferred over WuR-CTC in an average of 444 ms (with average throughput and overhead of 23.4 kbps, and 204.9 bytes, respectively), whereas a 127-byte IPv6 packet is transferred, on average of only 69 ms

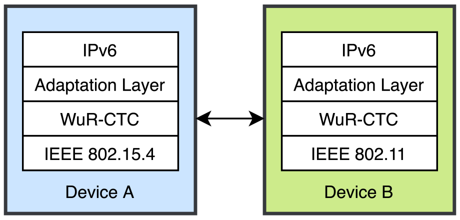


Figure 8.2: WuR-CTC-based communication between Device A (an IEEE 802.15.4 device) and Device B (an IEEE 802.11 device).

(with a throughput of 15.96 kbps, and 26.1 bytes of overhead). Accordingly, the designed solution even supports real-time interactions between IEEE 802.15.4 and IEEE 802.11 devices, without a gateway, in smart environments where there is a human in the loop (*e.g.*, the user presses a light control button and the reaction is visible on a lightbulb fastly for human standards).

The rest of the chapter is organized as follows. Section 8.2 presents related work. Section 8.3 overviews the main features of WuR-CTC. Sections 8.4 and 8.5 present our design and evaluation, respectively, of a SCHC-based adaptation layer to enable IPv6 over WuR-CTC.

8.2 Related work

This section is divided in two parts. The first one focuses on related work in the area of CTC. The second one overviews the literature in the field of SCHC.

8.2.1 Cross-Technology Communication (CTC)

CTC has gained the attention of academia, industry, and standards development organizations, as it enables communication between non-interoperable devices or networks without the need to use a gateway [5, 129–139]. In this subsection, we review literature that focuses specifically on CTC between IEEE 802.11 and IEEE 802.15.4 devices.

In [130], authors used energy patterns in the air to provide CTC. Additional hardware is required to perform energy sensing from other nodes and to wake up the receiver device when needed. This solution does not support bidirectional communication.

In [131], the authors presented FreeBee, a CTC system that allows communication by embedding symbols into the timing of beacon frames. FreeBee requires no hardware modifications to the communicating devices. However, the authors propose the use of WuR receivers to detect specific Wi-Fi APs to significantly reduce standby energy consumption. Li et al. implement a bidirectional, high-throughput CTC solution via physical layer emulation, called WEBee [132]. This method allows to manipulate an IEEE 802.11 packet payload, requiring no additional hardware. The same authors presented also a long range CTC solution called LongBee [133]. Their results show that LongBee allows to double the range of other CTC solutions.

In [134], the authors enabled concurrent communication with a solution called B^2W^2, obtaining a throughput greater than the one achieved with FreeBee [131]. Channel state information is used to allow communication in [135]. StripComm [136] uses an interference-resisting encoding, which is specially designed for coexistence environments. C-Morse uses packet and beacon timing to create patterns that are captured by sensing the RSSI [137]. In [138], the authors proposed the use of energy patterns with the existing data packets, a technique called CTC via data packets (DCTC). DCTC enhances throughput compared to other CTC techniques.

In [139], the authors presented NetCTC. This solution uses ACKs to provide feedback and reliability. This is accomplished by an emulation-based CTC using WEBee [132], with confirmation messages sent only by IEEE 802.15.4 devices. NetCTC requires access to raw signal samples, which is not available in most IEEE 802.15.4 devices.

In [5], authors proposed a new WuR system, where legacy Orthogonal Frequency Division Modulation (OFDM)-based IEEE 802.11 devices can send wake-up signals. The same authors proposed a CTC system which leveraged a WuR channel to transfer data between devices, creating a Link Layer that contains a Medium Access Control (MAC) sublayer and a Logical Link Control (LLC) sublayer [129]. Additional low-power hardware is required to detect WuR signals and enable bidirectional communication between IEEE 802.11 and IEEE 802.15.4 devices.

All the solutions described above provide valuable developments in the field of CTC for IoT. However, to the best of our knowledge, none of them provide IPv6 support over CTC.

8.2.2 SCHC

SCHC was specially developed for LPWAN technologies, therefore, most of its literature is related with LoRaWAN, Sigfox, or NB-IoT technologies. However, SCHC functionality is increasingly used in other scenarios [39, 40], including the ones where WuR-CTC is used.

A tutorial-style overview of SCHC is presented in [100]. The performance of SCHC C/D functionality has been compared to other IETF standards in [47], finding that SCHC offers performance advantages. Moons et al. [46] also compared SCHC C/D with other IETF standards showing that SCHC has lower overhead (20 times smaller when compared with 6LoWPAN), as well as smaller footprint and memory. Authors in [56] presented an improvement for SCHC C/D based on dynamic context. Ayoub et al. [140] used SCHC C/D to improve the mobility of devices for roaming in LoRaWAN. Multimodal communication using SCHC C/D was tested in [141], where energy performance was evaluated for LoRaWAN devices. The authors indicate that higher header compression rates may be achieved depending on how homogeneous the traffic is. In [142], another multimodal deployment was tested using SCHC C/D. Internet access was provided to LoRaWAN devices using SCHC in [143], where the authors also evaluated end-to-end security. In [144], authors presented a device discovery and context registration for SCHC and showed that SCHC can aignificantly reduce energy consumption.

The performance of SCHC F/R functionality was studied in [16], showing that it increases reliability, with a direct impact on goodput and energy consumption. In [61], the authors proposed the use of SCHC F/R to increase network capacity and reduce congestion. A mathematical model of the SCHC ACK volume and optimal configuration values when using SCHC over LoRaWAN and Sigfox is presented in [59]. Authors in [57] provided an evaluation of all F/R modes over an ideal channel, in terms of goodput, channel occupancy, and total delay. Uplink transmission using SCHC over LoRaWAN is evaluated in [116]. Also using SCHC over LoRaWAN, authors in [60] showed the benefits of SCHC for IPv6 and CoAP traffic in terms of delay and packet delivery ratio. In [114], Santa et al. provided personal mobility vehicles with IPv6 support using SCHC over LoRaWAN and NB-IoT. SCHC over Sigfox was implemented and evaluated in terms of packet transfer time and number of exchanged messages [115]. Also using SCHC over Sigfox in [145], the energy performance of SCHC F/R functionality was modeled and evaluated in terms of device lifetime.

While the reviewed work provides significant insights into the use of SCHC over LPWAN (or other) technologies, none of them focuses on using SCHC to support IPv6 over CTC technologies.

8.3 Cross Technology Communication with WuR

As mentioned in previous sections, one type of CTC that stands out, due to its unique characteristics, is WuR-CTC [129]. As shown in Fig. 8.3, WuR-CTC enables communication between an IEEE 802.15.4 device (Device A) and an IEEE 802.11 device (Device B) by using an additional element called the WuRx. The WuRx waits for an incoming WuS. The WuS is sent by the main radio of the other device, acting as WuTx. Once the WuS is detected by the WuRx, the latter sends data to the WuR-CTC Controller, which may reply by using the main device radio, the WuTx, enabling bidirectional communication. WuR-CTC comprises a Physical Layer (see Section 8.3.1) and a Link Layer (see Section 8.3.2).

8.3.1 WuR-CTC Physical Layer overview

The Physical Layer of WuR-CTC uses a single transmission rate of 250 kbps with OOK symbols. This rate and encoding are supported by the WuTx of both IEEE 802.15.4 and IEEE 802.11 main radios.

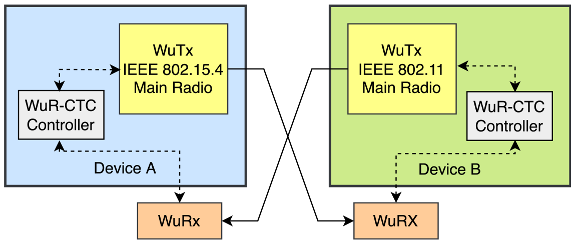


Figure 8.3: WuR-CTC communication between Device A (with an IEEE 802.15.4 main radio) and Device B (with an IEEE 802.11 main radio). The main radio of a device works as the WuTx.

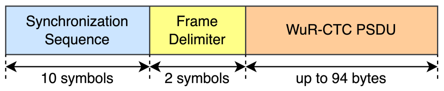


Figure 8.4: WuR-CTC PPDU frame format.

At this layer, a data unit called Physical Protocol Data Unit (PPDU) is defined by WuR-CTC. Fig. 8.4 shows the PPDU frame format. The PPDU includes a preamble sequence, a frame delimiter, and the Physical Service Data Unit (PSDU). The preamble sequence and the frame delimiter are headers intended for the synchronization process, which allows the WuRx to correctly receive the PSDU. Two encapsulation formats are required to allow a WuRx to decode the PPDU regardless of the WuTx technology.

For the IEEE 802.11 WuTx, the WuR-CTC PPDU is encapsulated in the IEEE 802.11g frame MAC Service Data Unit (MSDU) [129]. The bandwidth used by the IEEE 802.11g WuR-CTC PPDU is 20 MHz, with the PPDU encoded with OOK symbols at one bit per symbol, yielding a data rate of 250 kbps. The result is a standard-compliant IEEE 802.11g OFDM symbol. The IEEE 802.15.4 WuTx is configured to generate an ideal square OOK symbol at 250 kbps [129], with a bandwidth of 1 MHz. The WuR-CTC PSDU size of 94 bytes is given by the number of bits per OFDM symbol and the IEEE 802.11g encapsulation format, which provides the most limiting scenario of both IEEE 802.11g and IEEE 802.15.4 technologies [129].

8.3.2 WuR-CTC Link Layer overview

The WuR-CTC Link Layer comprises a MAC sublayer, and an LLC sublayer. The MAC sublayer provides medium access sharing and addressing. At the MAC sublayer, 10-bit unicast addresses are used. Each main radio technology, *i.e.*, IEEE 802.11 or IEEE 802.15.4, defines a specific medium access sharing mechanism. The IEEE 802.11 WuTx uses a standards-compliant transmitter, therefore CSMA/CA is used as defined by IEEE 802.11. The IEEE 802.15.4 WuTx CSMA/CA is tuned to be as close as possible to the one in IEEE 802.11 for mixed b/g networks, to reduce inter-network interference.

The LLC sublayer allows a sender to know whether the last frame sent was received or not, by using ACK frames, along with a sender transmission timer. ACK frames may carry data as payload. The LLC also provides a receiver transmission timer that allows the receiver device to enter into sleep mode upon timer expiration, when no more frames are received. The LLC protocol in WuR-CTC defines the control frames to wake up devices or to enable sleep mode.

Fig. 8.5 shows the WuR-CTC MAC Protocol Data Unit (MPDU), which is composed of the following fields: a 10-bit receiver address to identify the destination WuR-CTC node, a 10-bit sender address to identify the source WuR-CTC node, a 3-bit type field which indicates the frame type, a 1-bit Sequence Number (SN) that is used to match the corresponding ACK, an 8-bit length field that contains the length of the MAC Protocol Service Unit (MPSU), the MPSU (which ranges from 0 to 89 bytes), and an 8-bit CRC. Note that the MTU of WuR-CTC is only 89 bytes, which is smaller than the MTU required for IPv6 (*i.e.*, 1280 bytes).

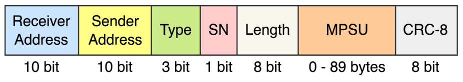


Figure 8.5: WuR-CTC MPDU frame format.

8.3.2.1 WuR-CTC frame types

The frame type field contains three flags. Ordered from the most significant bit to the least significant bit, these flags are the Data Flag, the ACK Flag, and the WuS Flag. By combining the flags, WuR-CTC defines 5 frame types as shown in Table 8.1, where two frame types are reserved for future use.

Table 8.1: WUR-CTC frame types

Frame Type	Active Flags (Data, ACK, WuS)	Description
DATA	100	This frame type must carry a payload with a non-null length. Used to transmit arbitrary data.
ACK	010	Acknowledge previous frame. SN must be the same as the one in the acknowledged frame.
WAKE	001	Wake up or sleep other stations. Payload data length must be 1 byte. Payload of 0xFF indicates the device should be woken up. Payload of 0x00 indicates the device must return to sleep.
DATA + ACK	110	Includes data and acknowledges the previous frame
WAKE + DATA	101	Includes data for a station in sleep mode, which returns to sleep mode after reception

The two remaining frame types are reserved for future use.

All frame types, except the ACK frame, are replied to with an ACK. Only one frame can be sent at a time, as no other frame transmission is allowed until the corresponding ACK is received or until the expiration of the sender transmission timer.

8.3.3 WuR-CTC Message Exchange Overview

In WuR-CTC, a series of frame exchanges between the sender and the receiver is required to begin and end communication. This process is described next, and is illustrated in Fig. 8.6, based on two nodes with addresses 0x001 and 0x002.

The sender starts the WuR-CTC communication by sending a WAKE frame with the aim to wake up the receiver device. Once the receiver is awake, it confirms its current state to the sender by transmitting an ACK frame. Once the ACK frame is received by the sender, it is responded to by an ACK frame to complete a 3-way handshake. Then, one or more DATA frames can be transferred following a stop-and-wait pattern. Each ACK frame may piggyback data from the receiver as well. Finally, once the last DATA frame is acknowledged, a SLEEP frame is sent, after which both nodes respond with an ACK frame and can return to sleep. Sending the SLEEP frame is not mandatory since the receiver will return to sleep automatically after a timeout; however, this is recommended in order to reduce the energy consumption of the receiver.

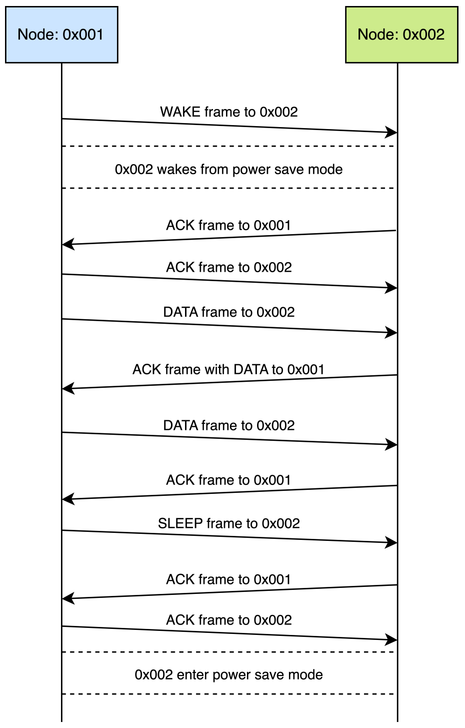


Figure 8.6: Communication between two nodes using WuR-CTC.

8.4 SCHC over WuR-CTC Profile design

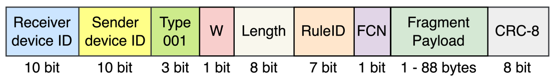
In this section, we present our design of a SCHC over WuR-CTC Profile, with the aim to optimally enable IPv6 over WuR-CTC. This section comprises: i) the SCHC over WuR-CTC header compression, ii) how SCHC F/R is adapted over WuR-CTC by reusing the WuR-CTC Link Layer, iii) SCHC over WuR-CTC message adaptation, and iv) SCHC over WuR-CTC message exchange.

8.4.1 SCHC over WuR-CTC header compression

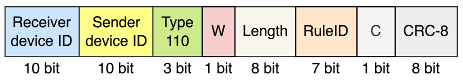
The SCHC over WuR-CTC Profile uses SCHC C/D as described in Section 2.3.3. The degree of compression that may be achieved depends on traffic predictability and on the design of the C/D Rules in order to compress protocol header fields. In our experimental evaluation (see Section 8.5), we are able to compress a 40-byte IPv6 packet header down to an 8-bit RuleID, with no compression residue, by using an *equal* matching operator with a *not-sent* action. If the resulting SCHC Packet size is greater than the WuR-CTC MTU (of 89 bytes), SCHC F/R is used.

8.4.2 SCHC over WuR-CTC Fragmentation and Reassembly

We now describe the design of our SCHC F/R adaptation to WuR-CTC. In order to efficiently use SCHC over WuR-CTC, we selected the ACK-Always SCHC F/R mode. In this mode, after each window of tiles, an ACK message is unconditionally sent. This matches the WuR-CTC behavior, whereby an ACK frame is always sent after each received DATA frame. To optimally adapt ACK-Always over WuR-CTC, we set the window size to one tile and use one tile per SCHC Fragment payload. A SCHC Fragment is carried in a DATA frame. After each SCHC Fragment, a SCHC



(a) SCHC over WuR-CTC fragment format.



(b) SCHC over WuR-CTC ACK format.

Figure 8.7: SCHC over WuR-CTC ACK message format.

ACK will be sent (carried by a WuR-CTC ACK frame).

Moreover, the ACK-Always F/R mode supports retransmission of missing SCHC Fragments. A SCHC Fragment retransmission is performed when the RT expires, if no corresponding SCHC ACK message has been received.

The SCHC over WuR Fragment and ACK message formats use the 1-bit WuR-CTC MPDU SN field as the SCHC F/R window number (W). Since there is only one tile per window, each tile can be numbered using a 1-bit FCN with value 0 for windows that are not the last one of the SCHC Packet. The last FCN will have a value of 1, which signals the last tile (and SCHC Fragment) of the fragmented SCHC Packet transfer (see Section 2.3.4).

The 3-way handshake provided by WuR-CTC is required before starting a SCHC Packet transfer, since the receiver node must be woken up. At the end of the SCHC Packet transfer, the sender node will send the SLEEP frame and will wait for the corresponding ACK frame before closing the transmission.

As described in [129], each WuR-CTC frame integrity is ensured with a CRC-8. Because the integrity of each SCHC Fragment is protected, the RCS used by default in SCHC is not needed in this case. The DTag is also not needed since SCHC Packet interleaving is not needed for SCHC over WuR-CTC. As explained in Section 8.3.2, WuR-CTC provides a sender timer and a receiver timer. SCHC over WuR-CTC uses both timers as RT and Inactivity Timer, respectively.

8.4.3 SCHC message adaptation over WuR-CTC

Figs. 8.7a and 8.7b show the adaptation of SCHC Fragment and SCHC ACK formats, when used over WuR-CTC DATA and ACK frames, respectively.

The SCHC over WuR-CTC message format extends the WuR-CTC Link Layer frame format by adding a 7-bit RuleID and a 1-bit FCN to the MSDU.

The tile size is equal to the WuR-CTC MTU (*i.e.*, 89 bytes) minus the SCHC Fragment header size (*i.e.*, 1 byte). Therefore, the tile size is 88 bytes for all SCHC over WuR-CTC fragments that are not the last one. The tile size of the last fragment can be smaller. The SCHC over WuR-CTC fragment format is valid for WuR-CTC Frame Type 001, *i.e.*, the DATA frame.

The SCHC over WuR-CTC ACK message format extends the WuR-CTC ACK frame format by adding a 7-bit RuleID and a 1-bit C field to the MPSU. The SCHC ACK bitmap is not used. In fact, no other SCHC Fragment can be sent before a sender receives the SCHC ACK message with the same W and there is only one tile per window; therefore, each SCHC ACK message corresponds to a unique tile. The SCHC over WuR-CTC ACK message format is valid for WuR-CTC Frame Type 010, *i.e.*, the ACK Frame.

8.4.4 SCHC over WuR-CTC message exchange

Fig. 8.8 shows the message exchange between two nodes that support SCHC over WuR-CTC. The first node (with address 0x001) wakes up the other node (with address 0x002) by using a WAKE frame. Node 0x002 awakens and replies with an ACK frame, which is responded to by node 0x001 with an ACK frame to complete the three-way handshake of WuR-CTC. Now that both nodes are awake, the SCHC Packet transfer may begin. Node 0x001 sends a SCHC over WuR-CTC fragment with FCN=0 and W=0. A SCHC over WuR-CTC ACK is sent by node 0x002 to confirm the correct reception of the corresponding fragment. This process continues until the last fragment of the SCHC Packet is sent. If any fragment is lost, it is retransmitted upon RT expiration. The last SCHC over WuR CTC Fragment has the FCN set to 1 to signal the end of the SCHC Packet transfer. Node 0x002 confirms the reception of the complete SCHC Packet by setting the C bit to 1 in the SCHC over WuR-CTC ACK. Finally, node 0x001 sends a SLEEP frame to node 0x002, which confirms reception with an ACK frame that in turn is confirmed by an ACK frame sent by node 0x001. Finally, both nodes can return to sleep mode and turn off their main radios.

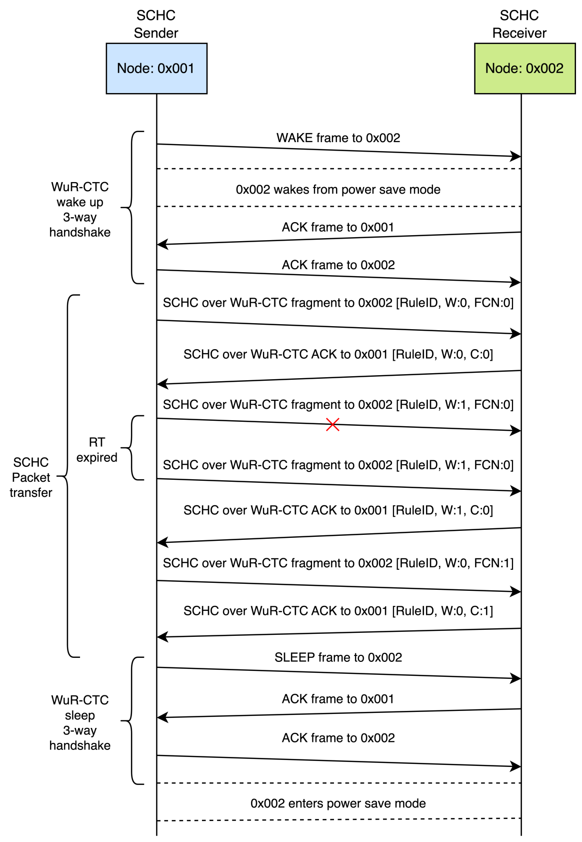

SCHC Sender
Node: 0x001
SCHC Receiver
Node: 0x002
WuR-CTC wake up 3-way handshake
WAKE frame to 0x002
0x002 wakes from power save mode
ACK frame to 0x001
ACK frame to 0x002
SCHC Packet transfer
SCHC over WuR-CTC fragment to 0x002 [RuleID, W:0, FCN:0]
SCHC over WuR-CTC ACK to 0x001 [RuleID, W:0, C:0]
SCHC over WuR-CTC fragment to 0x002 [RuleID, W:1, FCN:0]
RT expired
SCHC over WuR-CTC fragment to 0x002 [RuleID, W:1, FCN:0]
SCHC over WuR-CTC ACK to 0x001 [RuleID, W:1, C:0]
SCHC over WuR-CTC fragment to 0x002 [RuleID, W:0, FCN:1]
SCHC over WuR-CTC ACK to 0x001 [RuleID, W:0, C:1]
WuR-CTC sleep 3-way handshake
SLEEP frame to 0x002
ACK frame to 0x001
ACK frame to 0x002
0x002 enters power save mode

Figure 8.8: SCHC over WuR-CTC message exchange between Node 0x001 and Node 0x002, with one retransmission.

8.5 Evaluation of SCHC over WuR-CTC Profile

This section presents the SCHC over WuR-CTC Profile implementation and evaluation on a testbed. First, the SCHC over WuR-CTC implementation and testbed are described. Second, the experimental methodology is presented, followed by the performance metrics used to evaluate the SCHC over WuR-CTC Profile. Finally, the evaluation results are provided and discussed.

8.5.1 SCHC over WuR-CTC testbed implementation

The SCHC over WuR-CTC testbed consists of an IEEE 802.15.4 device and an IEEE 802.11 device. Both devices implement the WuR-CTC controller, *i.e.*, the WuTx, and are provided with WuRx additional hardware (see Fig. 8.9). On top of the WuR-CTC Controller, the SCHC layer offers C/D and F/R functionality and manages the WuR-CTC Link Layer by communicating with the WuR-CTC Controller.

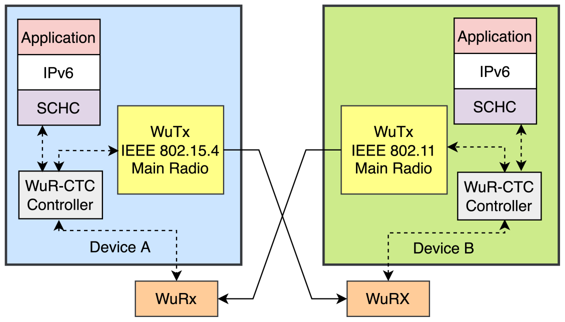


Figure 8.9: SCHC over WuR-CTC implementation architecture and main WuR-CTC blocks. The WuRx hardware is external to the device. The application generates a payload, *e.g.*, a smart thermostat measurement, which is carried by an IPv6 packet. The latter is compressed/decompressed and fragmented/reassembled by SCHC.

Fig. 8.10 shows the testbed set up to evaluate the SCHC over WuR-CTC Profile. The WuTx of the IEEE 802.15.4 and IEEE 802.11 devices are implemented by means of the EFR32MG Mighty Gecko and ESP32 Sparkfun development kits, respectively. Both devices also implement the WuR-CTC Controller. The EFR32MG Mighty Gecko is selected as its reconfigurable radio can provide OOK symbols at a rate of 250 kbps while being IEEE 802.15.4 standard compliant.

The WuRx additional hardware consists of two main elements: a RF Front-End and a Baseband module. The RF Front-End amplifies, demodulates, and normalizes the incoming signal. The Baseband module processes incoming signals into a binary stream and parses them according to the WuR-CTC protocol. The Baseband module is implemented on a STM32 Nucleo L053R8. The RF Front-End is implemented by using a Band-Pass Filter (GPIO Labs BPF), a Low Noise Amplifier (GPIO Labs 40 dB LNA), an envelope detector (LTC5508), and an operation amplifier (CA3140), as presented in [129].

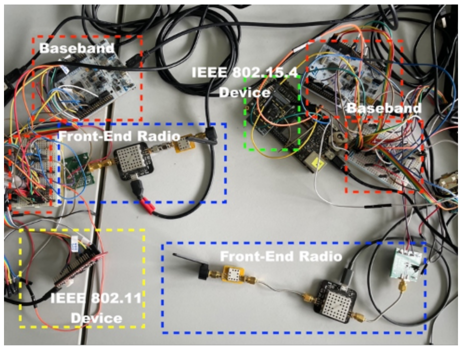


Figure 8.10: SCHC over WuR-CTC testbed implementation. The Front-End Radio and the Baseband modules are framed in dashed blue and red lines, respectively. The IEEE 802.11 Device is framed in dashed yellow lines. The IEEE 802.15.4 Device is framed in dashed green lines.

8.5.2 Experimental methodology

The SCHC over WuR-CTC Profile testbed is placed in an indoor environment (a lab in our university), where the two devices are separated by 15 cm and the WuRxs of both devices are calibrated to reduce propagation losses. The evaluation emulates an IEEE 802.15.4 smart thermostat sending measurements, carried by IPv6 packets, to an IEEE 802.11 central controller without the use of a gateway. We assume that the application data generation is predictable (*e.g.*, one measurement is sent every 5 minutes between a sender and a receiver that are known a priori), and that endpoint source and destination addresses are previously configured in a Rule. Accordingly, the SCHC C/D sublayer can compress the 40-byte IPv6 header down to an 8-bit RuleID, with no compression residue. With regard to SCHC over WuR-CTC Profile configuration parameters, two values (30 ms and 1000 ms) are tested for the RT. The 30-ms RT value is expected to allow faster IPv6 packet transfer. The 1000-ms RT value may be suitable when error bursts are present, as it spreads retransmissions over a greater time interval. The maximum number of SCHC over WuR-CTC fragment message transmission attempts is set to 5. The Inactivity timer is set to 2 seconds, allowing the receiver to be awaken for enough time to receive SCHC over WuR-CTC fragment message retransmissions in case of losses.

For each performance metric (see Section 8.5.3), the IPv6 packet sizes presented in Table 8.2 are evaluated. These values range from a 127-byte IPv6 packet -that requires a single SCHC over WuR-CTC fragment transmission, and thus is used as benchmark-, up to a 1280-byte IPv6 packet, which is the IPv6 MTU. Each experiment is performed 20 times for each IPv6 packet size, and for each RT value. A test is considered valid if the 3-way handshake of WuR-CTC is correctly performed (note that an error rate of 3.2% was found for the WuR mechanism in [129]). The SLEEP frame is only sent once per test, as the receiver device will enter the sleep mode, after the Inactivity timer expires, if no SLEEP frame is received.

Table 8.2: IPv6 Packet sizes evaluated

IPv6 packet size (bytes)	SCHC Packet size (bytes)	Number of SCHC over WuR-CTC fragments
127	88	1
215	176	2
303	264	3
479	440	5
743	704	8
1280	1241	15

8.5.3 Performance metrics

In the evaluation, we focus on the performance of the SCHC over WuR-CTC Profile when transferring an IPv6 packet. The performance metrics we use are: IPv6 packet compression ratio, error rate, total number of frames and overhead, transfer time, and throughput.

The compression ratio measures the relation between the IPv6 packet size and the SCHC Packet size. It can be calculated as follows:

$$\text{compression ratio} = \frac{\text{IPv6 packet size}}{\text{SCHC Packet Size}}. \tag{8.1}$$

The error rate is defined as the relation between the number of frames received (N_{FR}) and the number of frames sent (N_{FS}). It can be obtained as follows:

$$\text{error rate} = \left(1 - \frac{N_{FR}}{N_{FS}}\right) \cdot 100. \tag{8.2}$$

The total number of frames in an IPv6 packet transfer comprises: i) the number of WAKE, SLEEP and ACK frames, ii) the number of SCHC over WuR-CTC fragment messages (carried in DATA frames), and iii) the number of SCHC over WuR-CTC ACK messages (carried in ACK frames).

We define the overhead as the number of additional bits that are added in the transmission of the IPv6 packet by the SCHC over WuR-CTC Profile including the WAKE, SLEEP, and ACK frames, and the SCHC over WuR-CTC fragment and ACK message headers.

The IPv6 packet transfer time is defined as the time elapsed between the transmission of the WAKE frame that starts the transfer, and the transmission of the last ACK received after the SLEEP frame, as measured by the IEEE 802.15.4 device. Finally, we define the throughput as the IPv6 packet size divided by its corresponding transfer time.

8.5.4 Evaluation results

Fig. 8.11 shows the compression ratio obtained using (8.1) when applying SCHC compression to the IPv6 packet header, for the IPv6 packet sizes of Table 8.2. For small IPv6 packet sizes, the compression ratio is the highest, with a value of 1.44 for a 127-byte size. As the IPv6 packet size increases, the compression ratio decreases rapidly, since the payload size becomes the dominant component of the IPv6 packet size. For a 1280-byte IPv6 packet size, the compression ratio is only 1.03.

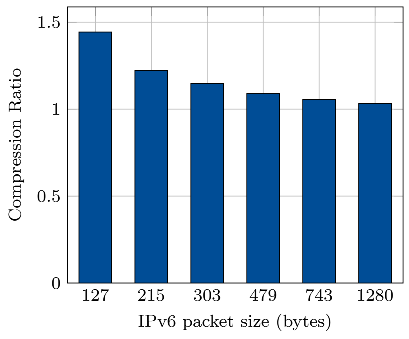


Figure 8.11: Compression ratio by using SCHC header compression as in our implementation.

Fig. 8.12 presents the error rate obtained in each test for each considered IPv6 packet size. The testbed is calibrated to reduce the error rate, as the experimental setup is sensitive to interference that may be present in the 2.4 GHz ISM band. The error rate values are similar for all IPv6 packet sizes, between 4.3% and 6.1% for the 30-ms RT experiments, and between 3.6% and 5.4% for the 1000-ms RT experiments. The overall error rate for a 1000-ms RT value is slightly smaller. This happens because greater RT values spread retransmissions over a larger time interval, which contributes to a better overcoming of burst interference.

Fig. 8.13a illustrates the average number of frames required for an IPv6 packet transfer. The number of WAKE frames is equal to 1, since tests are considered valid when the 3-way handshake is completed. Similarly, the number of SLEEP frames is equal to 1, as there are no retransmissions for this frame type.

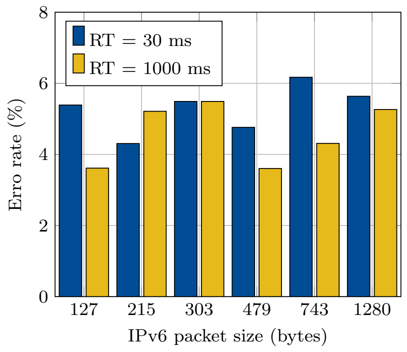


Figure 8.12: Error rate as a function of IPv6 packet size, for 30-ms and 1000-ms RT.

Recall that the WAKE frame requires 2 ACK frames for the 3-way handshake, which is equal to the number of ACK frames required by the SLEEP frame (see Fig. 8.8). Also in Fig. 8.13, the number of SCHC over WuR-CTC fragments and ACKs increases with the IPv6 Packet size and error rate, as more SCHC over WuR-CTC fragments are required or retransmitted, respectively. Differences in the average number of SCHC over WuR-CTC fragments sent, for the two RT values, are very small because the corresponding error rates are similar (see Fig. 8.12).

Fig. 8.13b shows the overhead incurred in the IPv6 packet transfers using SCHC over WuR-CTC. In each IPv6 packet transfer, a 6-byte WAKE frame is sent, followed by two ACK frames (one from the receiver and one from the sender), each ACK frame with a size of 5 bytes. Moreover, for each SCHC over WuR-CTC fragment, a 5-byte header is added by the WuR-CTC Link Layer and 1 byte is added by the SCHC F/R sublayer. The SCHC over WuR-CTC ACK size is 6 bytes.

The WAKE or SLEEP frame overhead remains constant for all the IPv6 packet sizes evaluated. The total ACK overhead has a greater relative impact on the smaller IPv6 packet sizes. Also in Fig. 8.13b, as the IPv6 packet size increases, the overhead increases, reflecting the overhead of each SCHC over WuR-CTC fragment, and the corresponding ACK. The total ACK overhead is larger than the overhead of the other frames combined, due to the 3-way handshake (for the WAKE and SLEEP frames) and the fact that every SCHC over WuR-CTC fragment is acknowledged. As the overhead is directly related to the number of transmitted frames, overhead differences for the considered RT values are also very small.

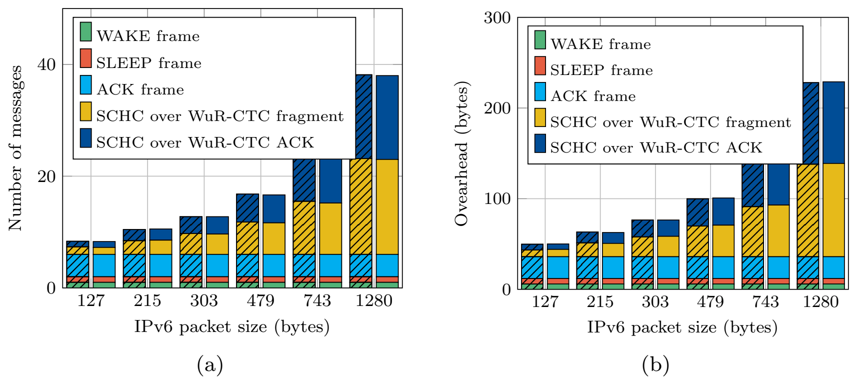


Figure 8.13: (a) Number of SCHC over WuR-CTC messages and (b) overhead (in bytes) of a SCHC over WuR-CTC IPv6 packet transfer, with a RT of 30 ms (with lines) and 1000 ms (without lines).

Fig. 8.14 presents the ranges of IPv6 packet transfer time results obtained. The minimum IPv6 packet transfer time values correspond to in cases where no losses occur, which are thus independent of the RT value. The average and the maximum IPv6 packet transfer times depend on the error rate (see Fig. 8.12) and the RT value.

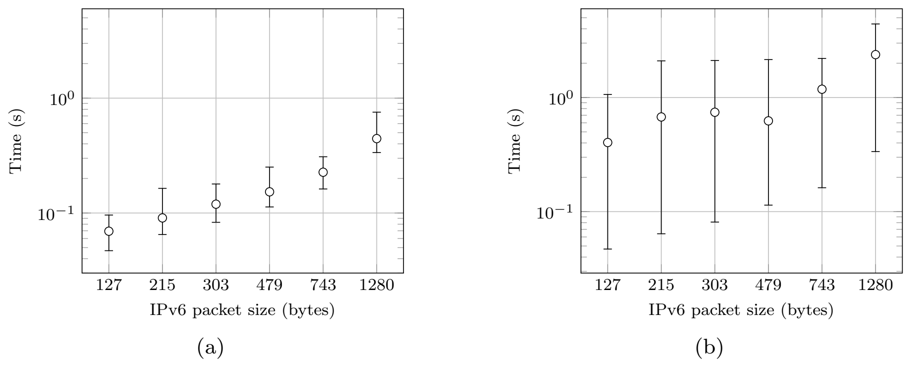


Figure 8.14: IPv6 packet transfer time, for different IPv6 Packet sizes, with (a) 30-ms RT and (b) 1000-ms RT.

The minimum IPv6 packet transfer time value difference between the 127-byte and 215-byte sizes corresponds to the time required to transmit only one additional SCHC Fragment (*i.e.*, one data frame and the corresponding ACK), with a value of approximately 18 ms.

In Fig. 8.14b, since the RT is set to 1000 ms, the IPv6 packet transfer time range increases significantly, since a fragment loss leads o an increase of one second. For example, for the 127-byte IPv6 packet size, the transfer time without message losses is approximately 47 ms, however it increases to 1050 ms when one fragment is lost. As the IPv6 packet size increases, the probability of losing SCHC over WuR-CTC fragments increases, which is reflected in a greater average and maximum IPv6 packet transfer time. For a 30-ms RT (see Fig. 8.14a), the effect of losing one message is significantly reduced, with a maximum IPv6 transfer time value of only 92 ms, for the 127-byte IPv6 packet size, since the device is able to retransmit sooner.

As further shown in Fig. 8.14, the IPv6 packet size of 1280 bytes exhibits the highest transfer time. This is due to the corresponding large number of SCHC over WuR Fragments. Without fragment losses, a 1280-byte IPv6 packet can be transferred in up to 336 ms.

Fig. 8.15 illustrates the throughput vs IPv6 packet size. Below 1280 bytes, the maximum throughput increases linearly with the IPv6 packet size, independently of the RT value. The average throughput decreases for large IPv6 packet sizes, due to the high transfer time (see Fig. 8.14).

The range of throughput values is greater for the RT = 1000 ms (see Fig. 8.15b), because the corresponding transfer time range is also greater. The 30-ms RT provides better throughput results, especially for large IPv6 packet sizes. On the other hand, the average throughput when RT = 1000 ms decreases more rapidly with the IPv6 packet size than when RT = 30 ms. Despite the fact that the 1000-ms RT value might be more suitable in the presence of bursts of errors, each frame loss penalizes the IPv6 packet transfer time to a greater extent, yielding lower throughput values.

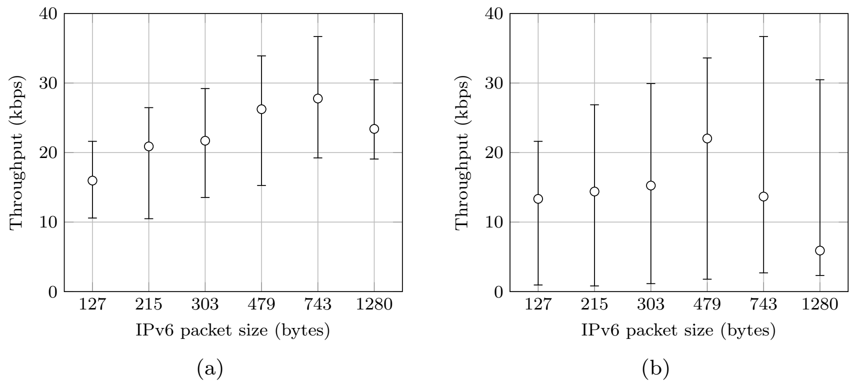


Figure 8.15: Throughput results for different IPv6 packet sizes, with (a) 30-ms RT and (b) 1000-ms RT.

CHAPTER 9

COMPOUND ACK MESSAGE AND LPWAN CONVERGENCE WITH SCHC

<<*Boundaries and holes allow light to pass through.*>>

In this chapter, we present two additional contributions of this thesis to the IETF LPWAN Working Group and to the SCHC standard. The chapter is organized into two sections. The first section presents the Compound ACK message, an adaptation of the SCHC ACK message. The Compound ACK message was initiated as part of the SCHC over Sigfox Profile, and later it was adopted by the LPWAN Working Group as a separate document [IX], as it can be used over other LPWAN technologies. The second section presents the SCHC Convergence Profile, our proposal to enable LPWAN convergence. LPWAN convergence can be understood as a way to enable seamless interoperability between LPWANs, and where both application and application developers are untied from the underlying network. This can be achieved by using IPv6 and the SCHC Convergence Profile. The SCHC Convergence Profile has been submitted as an individual document to the IETF [X].

9.1 Compound ACK message: Reducing downlink traffic

In this section we present the Compound ACK message, along with its motivation, format and benefits.

9.1.1 Introduction

In LPWANs, a key constraint is the amount of downlink traffic, *i.e.*, from the gateway to the IoT device. As explained in Chapter 4, this limitation happens because the gateway manages many data flows, and it is especially critical in regions where *duty-cycle* restrictions apply (*e.g.*, Europe 868 MHz band). Therefore, there is a bottleneck in downlink traffic, as the gateway capacity is limited. This directly reduces network capacity and the number of devices per gateway, which in turn reduces scalability and increases network cost.

In this thesis, we have shown how the ACK burden increases with the fragment error rate and how it depends on the fragment error pattern. In Chapter 4, we explained how ACK pooling provides better results for high error rates, as more fragment losses are concentrated in a single ACK. On the contrary, small ACK message sizes perform better for low fragment error rates. In the current SCHC specification [6], this is translated into changing the window size, which is fixed in the SCHC over LoRaWAN [9], SCHC over Sigfox [7], and SCHC over NB-IoT [8] Profiles.

Moreover, in Chapter 6, when fragment losses are present in the first windows, results showed an excess of ACK traffic in SCHC Packets of more than one window in size. The impact of fragment losses in the first windows results in an increase of SCHC Packet transfer time and number of downlink messages required per transfer. Therefore, we proposed the Compound ACK message to reduce the number of SCHC ACK messages by allowing multiple windows to be acknowledged in a single SCHC ACK message. This message is an extension of the SCHC framework ACK message, which allows aggregate receiver-feedback in a single downlink message by accumulating bitmaps of several windows.

The Compound ACK message can be used over LoRaWAN, Sigfox, NB-IoT, or any other technology using SCHC F/R. With this modification to the generic SCHC framework, ACK-on-Error F/R mode is provided with a variable MTU size in downlink.

9.1.2 Description

The SCHC Compound ACK message can be defined as a failure SCHC ACK message which contains one or more bitmaps. Each bitmap is identified by its corresponding window number. This is accomplished by adding the WINDOW_NUMBER (W) of each bitmap to the SCHC ACK message (which currently only had one W and one bitmap), and grouping the W with its corresponding bitmap. Therefore, as each bitmap is uniquely identified, the window numbers present do not need to be contiguous. However, they are ordered from the lowest-numbered to the highest-numbered window. Hence, if fragment losses are present in window zero, they are placed first, in the SCHC Compound ACK message header. Note that when all fragments have been correctly received (C=1), the Compound ACK message does not modify the success SCHC ACK message format as defined in [6].

Figs. 9.1 and 9.2 present the SCHC Compound ACK message format. The difference in formats between Figs. 9.1 and 9.2 is related to the number of bits that require padding at the end of the message. As the size of the SCHC Compound ACK is variable, M bits signal the end of message. However, in some cases, less than M bits are available without adding an extra L2 word. Note that in Fig. 9.2, what signals the end of the message is the absence of M bits, as it is not possible to find in the message another W and bitmap group. Because windows are ordered from lowest to highest, when present,Window number 0, is always placed in the SCHC ACK Header (w_1). This placement of window number 0 avoids confusion with the M bits that signal the end of message. Padding bits are not restricted to any value, as they can be differentiated in both cases.

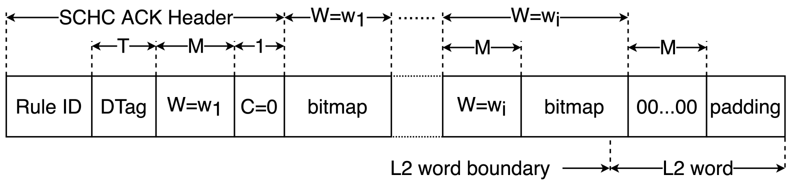


Figure 9.1: Compound ACK message format, modifying the failure SCHC ACK format, when more than M padding bits are required to complete the L2 word.

The Compound ACK message has a variable size, because a variable number of bitmaps, from different windows, can be signaled in a single message. Its variable size allows larger ACK sizes when ACK pooling can be performed or smaller ACK sizes for lower error rates. This can be accomplished with the same window size and the ACK size may depend on network error or congestion conditions.

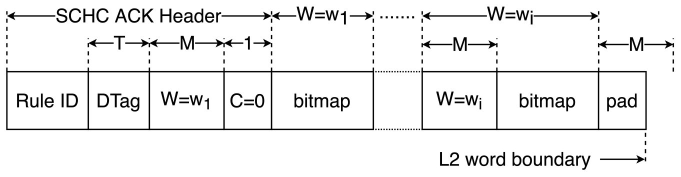


Figure 9.2: Compound ACK message format, modifying the failure SCHC ACK format, when less than M padding bits are required to complete the L2 word. Losses are found in windows $W = w_1, ..., w_i$; where $w_1 \leq w_2 \leq ... \leq w_i$

.

9.1.3 ACK reduction examples

In this subsection we present examples of the ACK reduction that can be achieved by using the Compound ACK message. As previously explained in this thesis, fragment losses in intermediate windows generate at least one ACK message. Fragment losses in the last window generate at least two ACK messages (one failure ACK and one success ACK message). The success ACK message will always be sent to maintain coherence, as it confirms the correct reception of the SCHC Packet. Therefore, reduction is possible in the failure ACK messages.

Fig. 9.3 shows a 14-tile SCHC Packet transfer using SCHC ACK-on-Error mode, with a window size of 7 tiles, and with fragment losses in all windows. After the first 7 tiles are sent, as there are fragment losses in the first window ($W = 0$), one ACK message is sent. In many LPWAN technologies, downlink messages are device-driven, which means that the device must enable a reception window. While the reception window is open, the network server can send a message, in this case an ACK message to the device. After the retransmission of missing fragments starting from $W = 0$, the device will continue sending fragments from the second window ($W = 1$). As there are fragment losses in $W = 1$, another ACK message is sent, reporting missing fragments. Finally, after all missing fragments in $W = 1$ are retransmitted, a success ACK message is sent, concluding the SCHC Packet transfer.

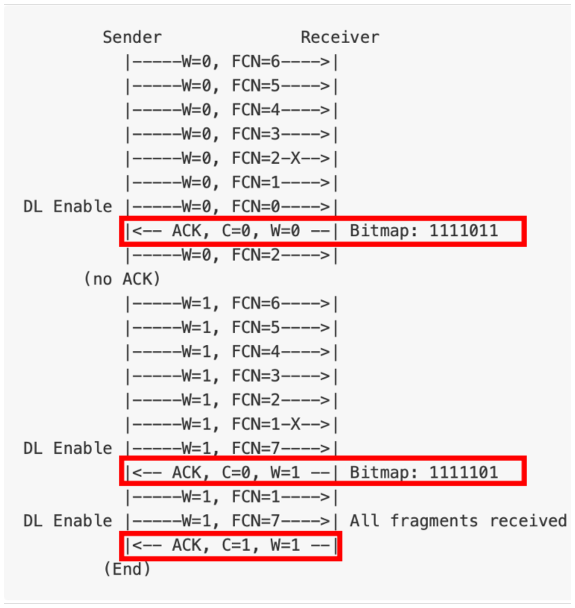
Fig. 9.4 shows the same SCHC Packet transfer as Fig. 9.3 but using the Compound ACK message. In contrast with the previous example, the Receiver (network) can select whether or not to send a Compound ACK message. After the first window of tiles ($W = 0$) is sent, even though there are fragment losses, the network can pool the fragment losses and wait for the next downlink opportunity. This provides flexibility to the network, *e.g.*, losses may be due to congestion, and sending more messages will worsen the network status. The sender (device) will continue sending fragments until the last one (FCN=7). This message will trigger a Compound ACK message reporting missing fragments from previous windows. Finally, after the retransmission of missing fragments is performed, a success ACK message is sent to end the SCHC Packet transfer.

The ACK reduction provided by using the Compound ACK message will depend on the number of retransmissions required after the first Compound ACK message is sent. The ACK reduction in the first transmission attempt, *i.e.*, each fragment sent at least once, can be calculated as:

$$ACK_{reduction} = N_W - 1, \tag{9.1}$$

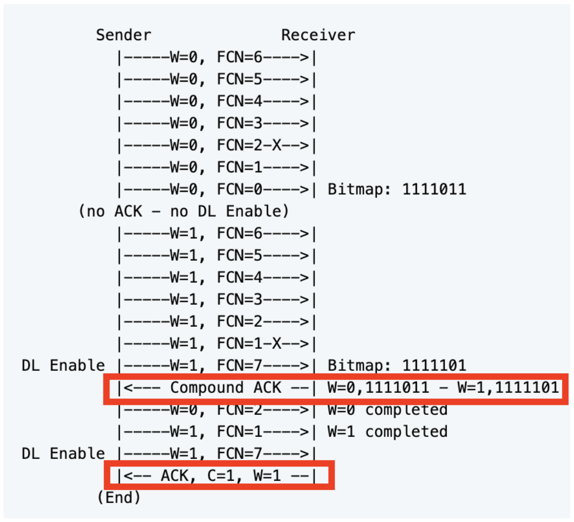
where N_W is the total number of windows with losses in the first transmission attempt of a SCHC Packet transfer. Therefore, as the SCHC Packet size increases and more windows are required, the benefits of using the Compound ACK message increases as the $ACK_{reduction}$ increases.

Moreover, as there is no need to increase the window size to achieve a larger ACK size, smaller window sizes can be used to benefit from ACK pooling. Small window sizes also provide more downlink opportunities, especially in LPWAN technologies in which the downlink message is device driven. The usage of Compound ACK and smaller window sizes provides more opportunities to



```
          Sender               Receiver
            |-----W=0, FCN=6----->|
            |-----W=0, FCN=5----->|
            |-----W=0, FCN=4----->|
            |-----W=0, FCN=3----->|
            |-----W=0, FCN=2-X--->|
            |-----W=0, FCN=1----->|
  DL Enable |-----W=0, FCN=0----->|
            |<-- ACK, C=0, W=0 --| Bitmap: 1111011
            |-----W=0, FCN=2----->|
        (no ACK)
            |-----W=1, FCN=6----->|
            |-----W=1, FCN=5----->|
            |-----W=1, FCN=4----->|
            |-----W=1, FCN=3----->|
            |-----W=1, FCN=2----->|
            |-----W=1, FCN=1-X--->|
  DL Enable |-----W=1, FCN=7----->|
            |<-- ACK, C=0, W=1 --| Bitmap: 1111101
            |-----W=1, FCN=1----->|
  DL Enable |-----W=1, FCN=7----->| All fragments received
            |<-- ACK, C=1, W=1 --|
          (End)
```

Figure 9.3: SCHC Packet transfer example with SCHC ACKs. The SCHC Packet size is 14 tiles and window size is 7 tiles. Red frames highlight the ACK messages.



```
          Sender               Receiver
            |-----W=0, FCN=6----->|
            |-----W=0, FCN=5----->|
            |-----W=0, FCN=4----->|
            |-----W=0, FCN=3----->|
            |-----W=0, FCN=2-X--->|
            |-----W=0, FCN=1----->|
            |-----W=0, FCN=0----->| Bitmap: 1111011
        (no ACK - no DL Enable)
            |-----W=1, FCN=6----->|
            |-----W=1, FCN=5----->|
            |-----W=1, FCN=4----->|
            |-----W=1, FCN=3----->|
            |-----W=1, FCN=2----->|
            |-----W=1, FCN=1-X--->|
  DL Enable |-----W=1, FCN=7----->| Bitmap: 1111101
            |<--- Compound ACK --| W=0,1111011 - W=1,1111101
            |-----W=0, FCN=2----->| W=0 completed
            |-----W=1, FCN=1----->| W=1 completed
  DL Enable |-----W=1, FCN=7----->|
            |<-- ACK, C=1, W=1 --|
          (End)
```

Figure 9.4: SCHC Packet transfer example with Compound ACK messages. The SCHC Packet size is 14 tiles and window size is 7 tiles. Red frames highlight the ACK messages.

send feedback, *i.e.*, a Compound ACK, to the device, while still being able to decide on network conditions (larger or smaller ACK sizes).

Furthermore, the Compound ACK message variable size is of special interest for LPWAN Convergence, as each LPWAN technology has different L2 characteristics, both in uplink and in downlink. The LPWAN Convergence Profile is presented in the following section.

9.2 The SCHC Convergence Profile

This section presents the SCHC Convergence Profile, our proposal to enable a SCHC F/R convergence sublayer across LPWAN technologies. This profile focuses on uplink fragmentation, *i.e.*, traffic generated from the device to the network, which is the typical usage of the SCHC framework.

9.2.1 Introduction

The IETF LPWAN WG has published several technology-specific SCHC profiles over LPWANs, *i.e.*, SCHC over LoRaWAN [9], SCHC over Sigfox [7], and SCHC over NB-IoT [8]. Each profile provides optimal configuration parameters for using SCHC over the specific technology. However, the F/R configuration of these profiles are not compatible between them. This is especially critical in multi-radio devices (*e.g.*, device supporting radio interfaces of LoRaWAN, Sigfox and NB-IoT), as it requires one implementation per technology, not only in the device but also in the backend server.

Another use case requiring multiple implementations of the SCHC F/R sublayer are multi-network scenario. In a muti-network scenario, the same application is deployed over different LPWAN technologies. For each deployment, a specific implementation is required at the device and backend server.

We performed a comparison of the current SCHC Profiles for uplink fragmentation, and the results showed no major differences between them. Therefore, it is possible to produce a single profile that works over all LPWAN technologies, taking into consideration the specifics of each technology. Moreover, this profile will reduce implementation complexity by providing a single convergence SCHC F/R sublayer, and will also leverage the benefits of the Compound ACK by using smaller windows and more downlink opportunities.

9.2.2 Motivation

In IoT application development, the LPWAN constraints influence the design of the application itself. The definition of which LPWAN is used in a specific solution presents problems when migrating the application to another LPWAN, as it may imply a complete IoT application redesign. This impacts the device code and the backend server code. In the IP domain, the lower layer, especially L2, is transparent to the application. However, this is not the case for LPWAN technologies.

SCHC was built to provide interoperability for IoT applications running in LPWAN networks, and a single C/D sublayer was proposed. However, current implementations require different implementations of the SCHC F/R sublayer, with slightly different F/R modes. Therefore, multiple SCHC F/R implementations are required when using multiple LPWANs, which is not the case for the C/D sublayer.

The motivation of the SCHC Convergence Profile is to reduce code complexity and maintenance by achieving a single F/R sublayer. The SCHC Convergence Profile works over LoRaWAN, Sigfox, and NB-IoT simultaneously, considering the singularities of these technologies (*e.g.*, when a downlink message can be sent) while providing general F/R modes. Therefore, it can be used for muti-radio devices and multi-network applications.

9.2.3 Use cases

We have identified several use cases for the SCHC Convergence Profile. These include:

- A generic SCHC F/R mode for implementation and testing over a new technology.
- Devices with more than one LPWAN radio, *i.e.*, multi-radio devices.

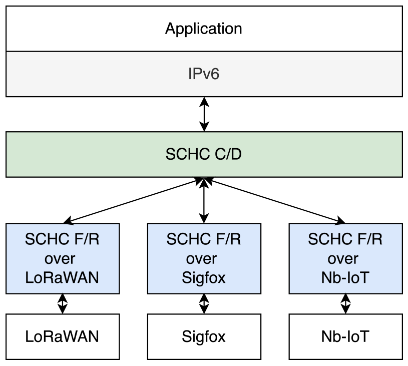
- Applications deployed over more than one LPWAN, *i.e.*, multi-network applications.
- For network redundancy:
  - Use another LPWAN as backup,
  - Send the same SCHC Fragment via different LPWANs, increasing the probability of success.
- Device *duty-cycle* can be increased, as more networks are available, *e.g.*, if transmission over LoRAWAN is not possible due to *duty-cycle* restriction, it may be performed over Sigfox or NB-IoT. This applies for SCHC Fragments and SCHC Packets.
- Check available network coverage by sending SCHC Fragments over different LPWANs.

9.2.4 Description

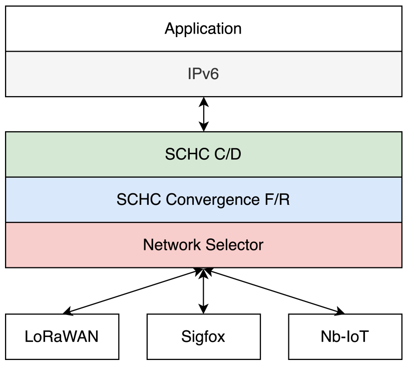
9.2.4.1 Protocol stack

Fig. 9.5 shows a comparison of the SCHC Convergence Profile with the existing SCHC protocol stack. Both protocol stacks provide a transparent LPWAN layer to the application, however they converge at different SCHC sublayers. Fig. 9.5a shows the convergence at the SCHC C/D sublayer using the existing SCHC protocol stack. In this case, the implementations requires as many SCHC F/R sublayers as LPWAN technologies are used. As the SCHC C/D sublayer is the same regardless of the LPWAN technology, SCHC Packets can be sent using any network. However, all SCHC Fragments must be sent using the same network, as the SCHC F/R configuration depends on the underlying technology. Once the complete SCHC Packet is received, it will be delivered to the SCHC C/D. This increases complexity and code maintenance.

Fig. 9.5b presents the SCHC Convergence Profile protocol stack. By using this profile, it is possible to reduce the complexity as the same SCHC F/R sublayer is used regardless of the underlying technology, making it a convergence SCHC F/R sublayer. By doing this, SCHC Fragments can be sent over different LPWANs. With the SCHC Convergence Profile, the IPv6 and application layers do not depend on the underlying LPWAN technology. A new element can be added called Network Selector. The Network Selector is in charge of, as its name states, selecting the LPWAN network for next SCHC Fragment transmission, opening a wide range of options. For example, the network can be selected according to several criteria such as network availability (*i.e.*, network with available coverage), network without *duty-cycle*, economic cost, among others.



(a) SCHC Multiple F/R Protocol Stack.

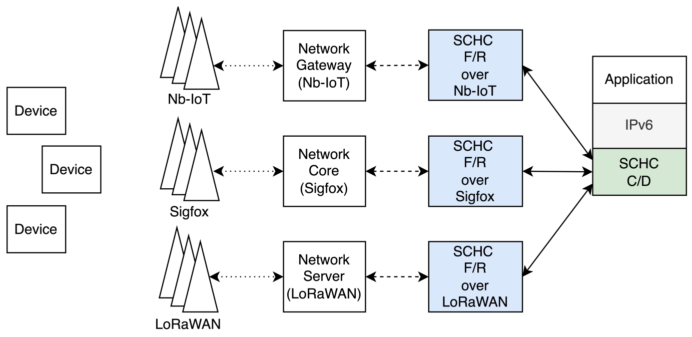


(b) SCHC Convergence Protocol Stack.

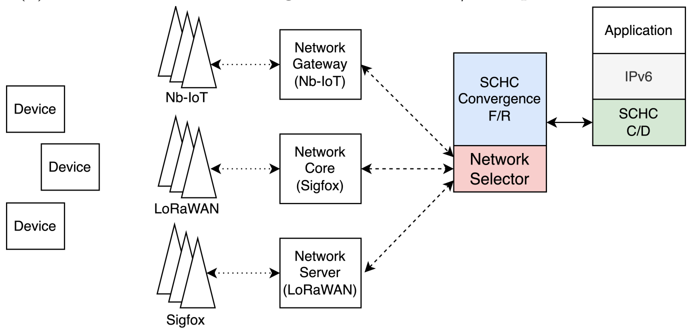
Figure 9.5: Comparison of SCHC protocol stack (a) with the proposed SCHC Convergence Profile protocol stack (b)

9.2.4.2 Architecture

Fig. 9.6 compares the current SCHC network architecture (Fig. 9.6a) with that proposed for LPWAN convergence (Fig. 9.6b). The latter provides a single implementation of the SCHC F/R sublayer, reducing code complexity. Moreover, with a single SCHC F/R implementation, the same F/R Rule space can be used, independently of the LPWAN technology. This reduces device memory usage and complexity when compared to multiple SCHC F/R implementation.



(a) Architecture when using several SCHC F/R implementations.



(b) SCHC Convergence Profile architecture

Figure 9.6: Comparison of current SCHC architecture (a) with SCHC Convergence Profile architecture (b).

9.2.5 Single SCHC ID

Each technology-specific SCHC Profile defines a different RuleID space. Therefore, to simplify the access to RuleIDs and to converge the different device IDs of each network involved, a new identifier called the single SCHC ID is needed. The single SCHC ID helps in two ways: i) unified RuleID space and ii) converged device ID of multiple networks into a single ID. This is accomplished with a device ID translation table, which maps the network device ID to the single SCHC ID. Then, with the single SCHC ID, it is possible to look up and identify the corresponding Rule set for such device. The use of a single SCHC ID dissociates the network device ID, allowing the usage of the same Rule set for the same device, independently of the access network. This reduces memory in the device and complexity in the Backend server.

Fig. 9.7 presents a diagram of the SCHC Convergence Profile architecture including the device ID translation table. This table is where the different device IDs are translated to the single SCHC ID. Also, this table can be used to save information related to a device, for example, IPv6 address or port. This way, when an IPv6 packet arrives to the SCHC C/D, the device ID translation table can provide mapping to the specific network ID from the IPv6 address.

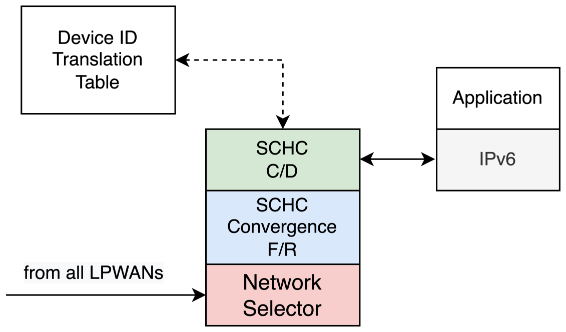


Figure 9.7: Single SCHC device ID translation table diagram

LPWAN technologies define different device IDs to identify the devices in each network. Currently, the devices IDs that converge in the SCHC Convergence Profile are:

- LoRaWAN: DevID
- Sigfox: DeviceID
- NB-IoT: IMEI

9.2.6 Uplink fragmentation

As stated in Chapters 3 and 4, the ACK-on-Error mode provides better performance than the other SCHC F/R modes for reliable SCHC Packet transfer. Moreover, when using different LPWAN technologies, there are different requirements at L2. The ACK-on-Error mode supports variable MTU sizes (which is critical when sending SCHC Fragments spread across different LPWANs), and out-of-order delivery (in case SCHC Fragments are received out-of-order at the SCHC F/R receiver). This makes the ACK-on-Error mode suitable for uplink fragmentation.

SCHC over LoRaWAN [9], SCHC over Sigfox [7], and SCHC over NB-IoT [8] specify SCHC F/R for uplink transmission. At the SCHC Fragment level, these profiles are not compatible with one another as their header field sizes are different. However, one of the SCHC over Sigfox uplink fragmentation modes present in the profile has several similarities with the ACK-on-Error SCHC over LoRaWAN profile. NB-IoT profile provides a more general and configurable ACK-on-Error mode, as NB-IoT has a less restrictive MTU size therefore, this profile is not considered in the comparison. Such similarities between the SCHC over Sigfox (Two-byte Option 2) and SCHC over LoRaWAN Profiles include:

- 2-byte SCHC Fragmentation Header size.
- 10-byte tile size.
- 1-byte Rule ID size.
- No DTag.

Differences between the SCHC over LoRaWAN and SCHC over Sigfox (Two-byte Option 2) uplink fragmentation profiles include:

- WINDOW_SIZE (tiles per window).

- M size (maximum number of windows).
- N size (tiles per window).
- RCS size and algorithm.

The SCHC over LoRaWAN Profile, in the ACK-on-Error mode, includes a WINDOW_SIZE of 64 tiles. This provides better performance in error-prone environments with larger ACKs. The SCHC over Sigfox Profile (Two-byte Option 2) includes a smaller WINDOW_SIZE of 32 tiles. The smaller window, half the size in this case, will double the downlink opportunities. Larger ACK sizes can be obtained by using the Compound ACK message. This provides the opportunity for reporting on more than one window on error-prone environments, while providing the option of sending smaller ACKs, reporting on one window, for low error rates. The SCHC Convergence Profile uses smaller WINDOW_SIZE as it uses the Compound ACK message to accomplish larger ACK size, while still having the option of smaller ACKs and more downlink opportunities.

9.2.7 SCHC Convergence Profile Uplink ACK-on-Error mode

The SCHC Convergence Profile uplink ACK-on-Error mode takes advantages of the similarities between current LPWAN profiles. The SCHC Uplink Fragmentation Header size is 16 bits composed as follows:

- Rule ID size is: 8 bits
- DTag size (T) is: 0 bits
- Window index (W) size (M): 3 bits
- Fragment Compressed Number (FCN) size (N): 5 bits.

Other configuration parameters are set as follows:

- MAX_ACK_REQUESTS: 5
- WINDOW_SIZE: 31 (with a maximum value of FCN=0b1011)
- Regular tile size: 10 bytes
- All-1 tile size: 1 to 10 bytes
- Retransmission Timer: Application-dependent.
- Inactivity Timer: Application-dependent.
- RCS size: 32 bits

9.2.8 SCHC Convergence Profile message formats

This section depicts the different formats of SCHC Fragment, SCHC ACK, SCHC Compound ACK, SCHC Aborts, and ACK Request used in the SCHC Convergence Profile for ACK-on-Error mode.

9.2.8.1 Regular SCHC fragment

Fig. 9.8 shows an example of a regular SCHC fragment. This fragment format is used for all fragments except for the last one.

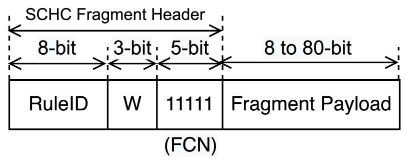


Figure 9.8: Regular SCHC fragment

9.2.8.2 All-1 SCHC fragment

Figs. 9.9 and 9.10 show examples of an All-1 SCHC fragment, with and without a tile, respectively. The last fragment of a SCHC Packet is the All-1 SCHC fragment. Depending on the SCHC Packet size, the All-1 may or may not carry the last tile.

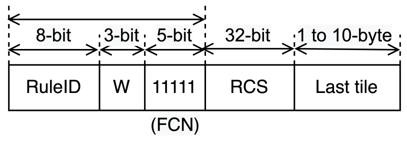


Figure 9.9: All-1 SCHC fragment (with tile)

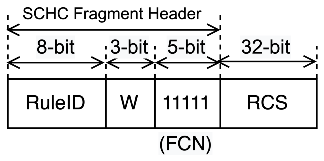


Figure 9.10: All-1 SCHC fragment (no tile)

9.2.8.3 SCHC ACK message

Fig. 9.11 shows an example of a success SCHC ACK message. The success SCHC ACK message was the C bit (the integrity check bit) set to 1 and no bitmap, as it does not report any fragment losses.

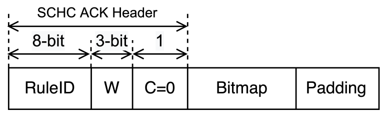


Figure 9.11: Successful SCHC ACK

Fig. 9.12 shows an example of a failure SCHC Compound ACK message. The Compound ACK message has a variable size which depends on total number of window that are reported in a single message.

9.2.8.4 SCHC Receiver-Abort message

Fig. 9.13 shows an example of a SCHC Receiver-Abort message. The SCHC Receiver-Abort message enables the receiver to abort current SCHC Packet transfer. For example, the receiver is

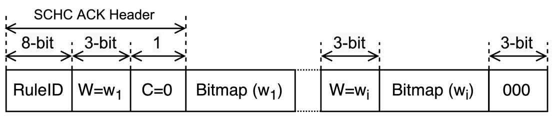


Figure 9.12: Failure SCHC Compound ACK message. Losses are found in windows $W = w_1, ..., w_i$; where $w_1 \leq w_2 \leq ... \leq w_i$

an application server, and it may need to abort current transfer due to overload.

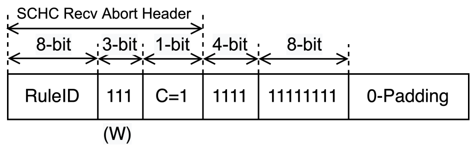


Figure 9.13: SCHC Receiver-Abort message

9.2.8.5 SCHC Sender-Abort messages

Fig. 9.14 shows an example of a SCHC Sender-Abort message. The SCHC Sender-Abort message is sent by the sender, *e.g.*, the device, when it needs to cancel current SCHC Packet transfer.

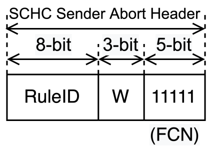


Figure 9.14: SCHC Sender-Abort message

9.2.8.6 SCHC ACK Request message

Fig. 9.15 shows an example of a SCHC ACK Request (ACK-REQ) message. This messages is used by the sender, *i.e.*, the device, to request a Compound ACK message.

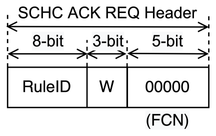


Figure 9.15: SCHC ACK Request message

CHAPTER 10 CONCLUSIONS AND FUTURE WORK

<<Light dwells in what it generates>>

In this chapter we present the main conclusions of this thesis investigation, along with future work directions. The first section presents the main conclusions, emphasizing our main findings and contributions. The second section provides some future work directions, with a perspective on the SCHC framework.

10.1 Conclusions

This thesis was motivated by the need for developing, evaluating and improving IPv6 support over LRLPWNs, as there existed a gap in the state of the art at the research level. During the time of this thesis investigation, the adoption of SCHC as a standard by the IETF drove the need to evaluate its performance. SCHC follows a technology-agnostic approach and, as a new framework, performance and optimal configuration values were unknown. In this context, this thesis main contributions were: i) analysis of the three SCHC F/R modes, ii) a mathematical model and an optimal tuning of ACK-on-Error mode, iii) analysis of new RFTs for SCHC, iv) development, implementation and evaluation of the SCHC over Sigfox Profile, v) design, implementation and evaluation of SCHC over WuR-CTC to support IPv6, and vi) creation of the Compound ACK and SCHC Convergence profile Internet Drafts. Following are the main conclusions of this thesis investigation, organized by chapter.

In Chapter 3, we performed an analysis of the three F/R modes of the SCHC Framework and their trade-offs. Firstly, we provided an overview of the three SCHC F/R modes: No-ACK, ACK-Always, and ACK-on-Error. Secondly, we explained the performance metrics used for the analysis. Lastly, we employed the OpenSCHC simulator, to which we added an implementation of the ACK-Always mode, to evaluate how the SCHC F/R modes perform under an error-free LoRaWAN communication channel. The three SCHC F/R modes have different characteristics and are focused on different use cases. Our evaluation shows that, in an ideal scenario with no errors, No-ACK mode can reduce the total delay and, due to its lower overhead, has a higher goodput, lower CO_T, and does not consume receiver resources. In the considered scenarios, the differences in total delay and CO_T between ACK-Always and ACK-on-Error are directly proportional to the number of windows required to transmit a given packet. This yields lower values of goodput, while increasing total delay and CO_T. Even though ACK-on-Error outperforms ACK-Always, the latter may be preferred for downlink fragmentation as the LoRaWAN gateway can be restricted to send a response right after the reception of a message.

The SF has a high impact on the performance achieved by the SCHC F/R modes. With higher SF, the differences observed between F/R modes performance increase. This is related to the fragment size, tile size, and number of fragments required for a given IPv6 packet size.

The reliability of No-ACK is very low since the loss of any fragment will lead to the loss of the complete packet being carried and the associated resources involved (*i.e.*, channel bandwidth, energy, etc.). ACK-Always and ACK-on-Error, on the other hand, provide high reliability at the expense of more overhead and thus, more receiver resources.

In Chapter 4, we developed a mathematical model to analyze the ACK-on-Error mode, focusing on the number of ACKs required to transmit data. We were then able to perform an analysis that allowed to determine the values of SCHC parameters minimizing the ACK burden imposed on the downlink which is more sensitive to *duty-cycle* constraints. We then applied our mathematical model to state-of-the-art LPWAN technologies such as LoRaWAN and Sigfox to minimize the ACK bit overhead, taking into account the retransmission process subtleties of the ACK-on-Error mode. Finally, we illustrated how the optimal parameter settings derived allow to decrease the ACK overhead.

In Chapter 5 we presented a thorough evaluation of RFTs for reliable fragmentation over LPWAN. We considered CB (which is the RFT used in SCHC), UB (as a benchmark), and two alternative RFTs: LLF and LoD (the latter, with 4 different encoding variants). We developed the Sim-RFT simulator to perform the evaluation. LoRaWAN was assumed as the underlying LPWAN technology.

Our results show that the optimal RFT depends on the channel conditions (in terms of error rate and error distribution), on L2 MTU, and on the size of the packet to be fragmented. CB is not optimal in all the scenarios evaluated. CB tends to offer the best performance for small packets and high error rates. In such conditions, its encoding based on a bitmap (which is further optimized compared with UB) is efficient. As packet size increases, regardless of the L2 MTU, LoD variants tend to become optimal. Regarding the latter, for uniform errors, 3-bit and 4-bit bases offer the best trade-off. For burst errors, LoD-2 offers the best performance, as it minimizes the encoding overhead for the very frequent delta of 1. Finally, for high quality links with very low error rate and uniformly distributed errors, LLF provides the most efficient encoding. Using the optimal RFT for a given scenario allows to achieve performance benefits such as a higher downlink network capacity (which is especially critical when *duty-cycle* regulations are in force), greater network scalability, and lower IoT device energy consumption.

In Chapter 6, we presented a performance evaluation of the SCHC ACK-on-Error F/R mode over Sigfox. We provided a theoretical analysis and an experimental evaluation conducted on real testbeds, for different Sigfox radio settings. Through measurements, we obtained the time required by the LoPy4 device for the different transmission stages of a Sifgox uplink and downlink frame, and provided a theoretical model for errorless transmissions, that was validated by experimental results.

We investigate the transfer time and the number of uplink and downlink frames, for all Sigfox RCs and for different SCHC Packet sizes. The number of uplink and downlink frames is critical in LPWAN networks, as message rates are limited. The transfer time of a SCHC Packet provides important insights for delay-sensitive applications and is especially critical when *duty-cycle* restrictions are enforced.

Furthermore, we proposed a new RCS mechanism which simplifies calculations, based on two facts: i) the integrity of each SCHC Fragment is guarantee by the L2, therefore the complete SCHC Packet integrity is guaranteed, and ii) the RCS helps identify missing fragments in the last windows, specially when this window is not full. Therefore, the new RCS mechanism includes the number of SCHC Fragments of the last window. By doing so, the receiver is able to know if there are missing SCHC Fragments in the last window and sends a SCHC ACK as necessary.

In Chapter 7, we presented a model and an evaluation of the device current and energy consumption of reliable packet fragmentation by using SCHC over Sigfox. We built our model by measuring the current consumption and duration of the states involved in a SCHC Packet transfer on a real Sigfox device.

The average current consumption of a single SCHC Packet transfer decreases with the SCHC Packet size since the device spends more time in the Sleep state to conform to the RC1 *duty-cycle* restrictions. For periodic SCHC Packet transfers, the average current consumption decreases with the SCHC Packet size and with the time between transfers. In contrast, the average energy consumption increases linearly with SCHC Packet size due to the energy consumed when the device is in the Sleep state.

We analyzed two fragment transmission strategies, which are compliant with RC1 *duty-cycle* restrictions: sending one or up to six back-to-back SCHC Fragments per cycle, respectively. The latter is more energy-efficient. In addition, we evaluated the light and deep sleep modes provided by the Sigfox device. The results highlight that the SCHC Packet size, the packet sending period, and the number of SCHC Fragments per cycle have a significant impact on the device lifetime.

In Chapter 8, we presented the design, implementation, and evaluation of a SCHC over WuR-CTC Profile in order to efficiently support IPv6. This Profile enables full interoperability between IEEE 802.15.4 and IEEE 802.11 devices, without the need of a gateway. Furthermore, the SCHC over WuR-CTC Profile can compress the IPv6 header down to only 1 byte, and optimally adapt the SCHC F/R ACK-Always mode to the WuR-CTC Link Layer, while adding only 1 byte of overhead per frame, and allowing IPv6 packet transfers up to 1280 bytes.

The main overhead in IPv6 over WuR-CTC is the ACK traffic incurred by WuR-CTC, which is leveraged in SCHC over WuR-CTC by sending SCHC ACKs, and includes also the WuR-CTC ACKs present in the three-way handshakes performed at the beginning and at the end of packet transmission. IPv6 packet transfer time increases with the error rate. Experiments have shown that a low RT value yields low IPv6 packet transfer times. SCHC over WuR-CTC can support real-time applications involving a human, especially for smaller IPv6 packet sizes and RT values, where the average transfer time for 127-byte IPv6 packets is only 69 ms. Finally, by providing IPv6 support over WuR-CTC, we have demonstrated the possibilities of exploiting SCHC to enable IPv6-based communications beyond the LPWAN landscape, for which SCHC was originally designed.

In Chapter 9, we presented two further contributions to the IETF LPWAN WG, which are a result of this thesis investigation. As part of the analysis performed using the SCHC over Sigfox profile, the Compound ACK was created. The Compound ACK is an optimization of the SCHC ACK message that allows fragment error feedback from one or more windows, making downlink traffic more efficient, specially for the ACK-on-Error F/R mode. Moreover, the LPWAN can select when to send the Compound ACK, giving the network flexibility and allowing larger or smaller ACKs, depending on the number of windows reported per ACK.

The creation of different SCHC Profiles, one per LPWAN technology, led to multiple implementations of the SCHC F/R sublayer, with smaller differences. To converge to a single SCHC F/R sublayer, we proposed the SCHC Convergence Profile. We carried out a comparison between the different F/R modes of each current technology-specific Profile and performed an adaptation, taking in consideration the Compound ACK flexibility. By having a single SCHC convergence Profile for F/R, it is possible to converge the different LPWAN technologies at the SCHC Fragment level, making it possible for multi-radio devices to send SCHC Fragments through different LPWANs or use other LPWANs as backup. The SCHC Convergence Profile makes the LPWAN layer transparent to the application and application development, as it is in the IP domain.

10.2 Future work

In this section, we provide future work directions that stem from this thesis investigation and the collaboration with the IETF LPWAN WG. This section is divided in two subsections. The first subsection presents the future research topics related to SCHC and interoperability. In the second subsection, we present SCHC future use cases beyond the LPWAN landscape.

10.2.1 A new era of interoperability

The SCHC framework has opened the doors for a new era of interoperability, by providing IPv6-based protocols to LPWANs. IPv6 provides a scalable address space, support for security protocols and interoperability, by using the Internet protocol stack. This becomes crucial to enable a scalable and secure IoT ecosystem. With SCHC, it is possible to assign an IPv6 address or a port to every LPWAN device. By doing this, each device is connected to the Internet. However, how and where this information is stored, and how the device SCHC session is handled is still an open challenge. The SCHC Convergence Profile provides a device ID mapping table, which can be extended to include the device IPv6 address and port. It may also include information related to currently active SCHC Packet transfers, which is called "SCHC Session". Implementation and evaluation of the SCHC Convergence Profile, including the device ID mapping table, constitute an interesting research direction.

As an IoT deployment grows, the need for device management increases. The device management overhead, summed with the IPv6 and ACKs overhead, can become a problem, specially in large deployments. In this sense, SCHC provides a new layer where device generated data and management data overhead can be compressed and reliably transferred. By using SCHC compression, the amount of data is reduced, providing a network load reduction. An interesting future research item is to study the compression impact on the data generated from deployment management, specially when a large number of devices is involved. As new open protocols (*e.g.*, CoAP, Lightweight Machine to Machine -LwM2M-) are used to perform IoT management, it will be interesting to experimentally evaluate the benefits of using SCHC in the management plane. This can be performed using the SCHC Convergence Profile, which provides device communication independently of the underlying network, simplifying device access.

The SCHC Convergence Profile provides a way to converge all LPWANs at the SCHC F/R by using the same SCHC Fragment structure, *i.e.*, the same header fields. This profile is now focused on uplink fragmentation with ACK-on-Error mode, but it can be extended to downlink fragmentation with ACK-Always mode, and to uplink fragmentation with No-ACK mode. Moreover, in this profile, the network that will perform next fragment transmission can be selected on-the-fly. Therefore, a future research direction would be to study the Network Selector layer, focusing on the criteria that can be used to select the optimal network *e.g.*, based on network coverage, record of fragments retransmissions, among others. The network selection can be performed by using Machine Learning and Artificial Intelligence models, which can be trained for optimal network selection.

Similarly, the SCHC Compound ACK message gives the network the opportunity to decide when to send feedback to the fragment sender. However, mechanisms to select the most appropriate downlink opportunity are still to be designed and evaluated. These mechanisms may depend on network load, device transmission record, and metadata obtained from the network, *e.g.*, if the network is receiving SCHC Fragments with losses, and the device is not moving (network metadata shows the same radio base for all SCHC Fragments), then the network may wait for the next downlink opportunity to send the SCHC Compound ACK message.

The SCHC Compound ACK message provided a new way to provide receiver feedback. The SCHC over WuR-CTC Profile uses the piggybacking functionality of WuR-CTC to transfer the ACK data. A future research direction can be to actually include data in the SCHC ACK, resulting in piggybacking. By doing this, bidirectional communication is possible. One way this can be achieved is by adding one bit to the M tail bits of the Compound ACK, which can signal the usage of piggybacking.

The SCHC over WuR-CTC Profile proves the benefits of combining WuR-CTC and SCHC. The support of IPv6-based communication in the WuR channel allows applications to communicate, without the need of a gateway. Future research work can include the production of a printed circuit of the SCHC over WuR-CTC testbed, considering the integration of the Front-End radio and Baseband modules, including measurement point and calibration potentiometers. The printed circuit schematics can be open-source to promote the adoption of this solution.

10.2.2 SCHC: LPWAN and beyond

As SCHC is becoming a mature standard, it has called the attention of researchers to apply it to other network technologies. This is mainly because of its benefits, namely: IPv6 support, low F/R overhead with reliability, and ultralightweight header compression. These new technologies include IEEE 802.15.4 [40] and PPP [39]. Work has focused on how SCHC header compression can be applied over these non-LPWAN technologies. In SCHC over LPWANs, Rules are associated to the device or the application, with a direction field indicating uplink or downlink transmission. LPWAN technologies are based on a star topology, as a result, device, application, uplink, and downlink are clearly defined. In technologies such as IEEE 802.15.4, this is not always the case, as its architecture can be a mesh network. In non-star (*e.g.*, mesh) topologies, the idea of Dev, App, uplink or downlink is lost. Future research work can focus on the implementation and evaluation of these new SCHC Profiles, such as [40], and how Rules may be adapted to work in all use cases.

The SCHC compression mechanism is based on a static context which must be configured at both ends before the first SCHC Packet is transferred. In most LPWAN technologies, this is performed when the device is bootstrapped, or updated with the latest application version. As SCHC is widely used over new technologies, a need for Rule context exchange arises. Therefore, an interesting future research direction is designing and evaluating new ways to perform the Rule exchange and provide secure Rule access policies. This may include a specific CoAP POST request, with the rule set encoded using Concise Binary Object Representation (CBOR) [146], or a SCHC handshake-like message exchange.

SCHC provides two reliable F/R modes, *i.e.*, ACK-on-Error and ACK-Always. SCHC Packets are protected by the RCS, which allows the receiver to check the integrity of the F/R process and confirm the correct reception of all SCHC Fragments. However, missing SCHC Fragments must be retransmitted. One way to reduce fragment retransmission is to use Forward Error Correction (FEC) techniques, where the sender adds redundant data to the original message, *e.g.*, performs the XOR operation on two fragments and sends the result in a third fragment. This will increase the amount of data sent, but allows the receiver to perform error recovery. This is of special interest in networks where retransmission can increase congestion, or to alleviate the amount of downlink (SCHC ACK) traffic when fragmentation is performed in the uplink. As future research work, design and evaluation of FEC techniques for SCHC can be performed and added to the SCHC Convergence Profile.

Furthermore, wireless technologies where devices are not battery powered and bit rate and frame size are not a limitation, may benefits from the SCHC framework. As these wireless networks scale up, traffic can cause congestion. Therefore, to reduce congestion and improve reliability, SCHC C/D and F/R can be used. With C/D, the objective is to reduce the amount of overhead traffic sent, while F/R provides reliability, *i.e.*, using ACK-on-Error mode. The SCHC Convergence Profile can be used with these technologies, as it provides an out-of-the-box F/R mode. The design and performance evaluation of the SCHC Convergence Profile over these technologies is another promising future research direction.

There are currently efforts to provide SCHC with an Ethernet number and UDP port [147]. This allows to use SCHC over Ethernet links, allowing the link to be aware that SCHC compression is applied to the following data. Using the SCHC Ethernet number and UDP port, defining use cases, and evaluating them is another promising future research direction.

The popularization of satellite IoT opens opportunities to the SCHC framework, as it can also be applied in this context. Future research directions may focus on extending the SCHC Convergence Profile to include satellite IoT connectivity, more specifically on the connectivity singularities of satellite links, *e.g.*, how the device receives the downlink messages.

During the design, implementation and evaluation of the SCHC over Sigfox profile, we saw that the message size in some use cases is small, *i.e.*, fits in one L2 frame, and tends to be periodic, *i.e.*, one message each specific number of minutes or hours. These messages lack a retransmission mechanism and since IPv6 is not involved, SCHC compression is not needed. Therefore, SCHC may be used for reliably sending messages spread across the day with one Compound ACK message at the end of the day. As messages are generated and sent, there is no notion of a SCHC Packet. This produces problems with the RCS as defined in [6], as the device needs to save all messages sent and then calculate the CRC32 to be attached to the last SCHC Fragment. On the other hand, the receiver cannot deliver any measurement to the application, as the complete SCHC Packet has not arrived. This was solved in the SCHC over Sigfox Profile by defining a new RCS mechanism, which leverages the L2 integrity check.

A future research direction would be to produce a new F/R mode for a continuous stream of messages. Instead of sending a SCHC Packet, what is sent is a continuous stream of SCHC Fragments, each being a self-contained message, such as, a sensor measurement or device location. If the L2 does not provide integrity check, each SCHC Fragment may include it. Each time a window of fragments is completed, a SCHC Compound ACK message can be sent. Afterwards, the sender can continue with the transmission of SCHC Fragments from the next window. When SCHC Fragment transmissions pertaining to the last window are completed, the window number can be reset and the transmission of SCHC Fragments can continue (the DTag value can be increased in each cycle). Whenever the connection is aborted, the next transmission can start with the first window and a new DTag number. Design and evaluation of such F/R profile can be an interesting future research direction.

CHAPTER 11 CONTRIBUTIONS AND RECOGNITIONS

<<*Light is unification.*>>

Journal articles:

[I] S. Aguilar, P. Maillé, L. Toutain, C. Gomez, R. Vidal, N. Montavont, and G. Z. Papadopoulos, "Performance Analysis and Optimal Tuning of IETF LPWAN SCHC ACK-on-Error Mode," in IEEE Sensors Journal, vol. 20, no. 23, pp. 14534-14547, 1 Dec.1, **2020**.

[II] S. Aguilar, R. Vidal and C. Gomez, "Evaluation of Receiver-Feedback Techniques for Fragmentation Over LPWANs," in IEEE Internet of Things Journal, vol. 9, no. 9, pp. 6866-6878, 1 May, **2021**.

[III] S. Aguilar, D. S. W. La-Torre, A. Platis, R. Vidal, C. Gomez, S. Céspedes, and J. C. Zúñiga, "Packet Fragmentation Over Sigfox: Implementation and Performance Evaluation of SCHC ACK-on-Error," in IEEE Internet of Things Journal, vol. 9, no. 13, pp. 11057-11070, 1 July, **2022**.

[IV] S. Aguilar, A. Platis, R. Vidal, and C. Gomez. Energy Consumption Model of SCHC Packet Fragmentation over Sigfox LPWAN. Sensors **2022**, 22, 2120.

[V] S. Aguilar, R. Vidal and C. Gomez, "IPv6 over Cross-Technology Communications with Wake-up Radio," in Elsevier Internet of Things Journal, Dec **2022** (under review)

Conference papers:

[VI] S. Aguilar, A. Marquet, L. Toutain, C. Gomez, R. Vidal, N. Montavont, and G. Z. Papadopoulos, "LoRaWAN SCHC fragmentation demystified," in Ad-Hoc, Mobile, and Wireless Networks, M. R. Palattella, S. Scanzio, and S. Coleri Ergen, Eds., vol. 11803. Springer International Publishing (Cham), **2019**, pp. 213-227.

[VII] Wistuba, D.; Cespedes, S.; Zuniga, J.; Munoz, R.; Aguilar, S.; Gomez, C.; Vidal, R., An implementation of IoT LPWAN SCHC Message Fragmentation and Reassembly, Spring School on Networks, Vitoria, Brazil, 14 Dec. **2020**

IETF Internet Standards:

[VIII] J. C. Zuniga, C. Gomez, S. Aguilar, L. Toutain, S. Cespedes, D. Wistuba La Torre, and J. Boite, "SCHC over Sigfox LPWAN," Working Draft, IETF Secretariat, Internet-Draft draftietf-lpwan-schc-over-sigfox-13, November **2021**, work in progress.

[IX] J. C. Zuniga, C. Gomez, S. Aguilar, L. Toutain, S. Cespedes, and D. Wistuba La Torre, "SCHC Compound ACK," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-lpwan-schc-compound-ack-07, October **2021**, work in progress.

[X] S. Aguilar, C. Gomez, and R. Vidal, "SCHC Convergence Profile," Individual Submission, IETF Secretariat, Internet-Draft draft-aguilar-lpwan-schc-convergence-00, October **2021**, work in progress.

Open source projects

OpenSCHC. Implementation of SCHC. Main contributor.
Adopted by the LoRa Alliance as certification reference for IPv6 over LoRaWAN
https://openschc.github.io/openschc/index.html

The SCHC over Sigfox Project. Implementation of the SCHC over Sigfox Internet draft.
https://github.com/schc-over-sigfox

Simulators

Sim-RFT. Simulator for Receiver-Feedback Techniques (RFT) performance evaluation.
https://www.github.com/saguilardevel/sim-RFT.

SCHC over Sigfox Implementation: Sender and Device code SCHC F/R.
https://github.com/saguilarDevel/schcfox

Projects

ALLINONE Project (code: TEC2016-79988-P)

WINTERTIME Project (code: PID2019-106808RA-I00)

International research visit

Research exchange to IMT-Atlantique, Rennes, France.
Dates: 07/01/2019 to 01/07/2019

IETF active participation

IETF 110 Remote
Presentation of the SCHC Compound ACK to the LPWAN WG.
Dates: 10/03/2021

IETF 112 Remote
Presentation of Hackathon implementation and results.
Dates: 09/11/2021

IETF 114 Vienna - Remote
Presentation of advances in the SCHC Compound ACK draft and SCHC over Sigfox draft.
Dates: 09/11/2021

IETF 115 London - Onsite
Presentation of the SCHC Convergence Profile.
Dates: 09/11/2021

IETF Hackathon participations

IETF 105 Hackathon Montreal - Remote
Contributions to Open SCHC
Dates: 20-21/07/2019

IETF 106 Hackathon Singapore - Remote
Contributions to Open SCHC
Dates: 16-17/11/2019

IETF 108 Hackathon Remote
Off-line coding between UChile and UPC, implementation of ACK-on-Error mode over Sigfox.
Dates: 20-24/07/2020

IETF 110 Hackathon Remote
Off-line coding between UChile and UPC, implementation of ACK-on-Error mode over Sigfox.
Dates: 1-5/05/2021

IETF 112 Hackathon Remote
Example of SCHC over Sigfox draft in a ”real case” scenario, a Mini weather station: Measure temperature and humidity. Sent JSON file with measured data from Sigfox Device.
Equipment:
LoPy4, Pysense
Codebase:
SCHC over Sigfox Project in github, url: https://github.com/schc-over-sigfox/Hackathon-IETF112
Dates: 1-5/05/2021

IETF 113 Hackathon Vienna - Remote
Asset tracker using SCHC over Sigfox.
Equipment:
LoPy4, GPS, Temperature Humidity Sensor

Codebase:
SCHC over Sigfox Project in github, url: https://github.com/schc-over-sigfox/Hackathon-IETF113
Dates: 19-20/03/2022

IETF 115 Hackathon London - On Site
Documentation of Open SCHC and integration with Sigfox.
Dates: 05-06/11/2022

Recognitions

Network Engineer Day 2022 - NErD22.
First Prize in the Ph.D. 2 Category (more than 3 years)
Network Engineering Department
Universitat Politèctica de Catalunya
Dates: 05/07/2022

Grants

Awarded an Internet Research Task Force (IRTF) travel grant to attend the IETF 115 meeting
London, England
Dates: 05/11/2022 - 11/11/2022

BIBLIOGRAPHY

[1] S. Farrell, “Low-Power Wide Area Network (LPWAN) Overview,” Internet Requests for Comments, RFC Editor, RFC 8376, May 2018.

[2] S. Deering and R. Hinden, “Internet protocol, version 6 (ipv6) specification,” Internet Requests for Comments, RFC Editor, RFC 8200, July 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8200.txt

[3] LoRa Alliance Technical Committee, “LoRaWAN 1.1 Specification,” https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf, Oct 2017.

[4] Sigfox, “Sigfox Connected Objects: Radio specifications,” https://build.sigfox.com/sigfox-device-radio-specifications, 2 2019, Rev. 1.3, Accessed on 04-05-2021.

[5] M. Caballe, A. Calveras, E. Lopez-Aguilera, E. Garcia-Villegas, I. Demirkol, and J. Aspas, “An Alternative to IEEE 802.11ba: Wake-Up Radio with Legacy IEEE 802.11 Transmitters,” *IEEE Access*, vol. 7, pp. 48 068–48 086, 2019.

[6] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and J.-C. Zúñiga, “SCHC: Generic Framework for Static Context Header Compression and Fragmentation,” RFC 8724, April 2020.

[7] J.-C. Zúñiga, C. Gomez, S. Aguilar, L. Toutain, S. Cespedes, and D. S. W. L. Torre, “SCHC over Sigfox LPWAN,” Internet Engineering Task Force, Internet Draft draft-ietf-lpwan-schc-over-sigfox-13, Nov. 2022.

[8] E. Ramos and A. Minaburo, “SCHC over NB-IoT,” Internet Engineering Task Force, Internet-Draft draft-ietf-lpwan-schc-over-nbiot-13, Nov. 2022.

[9] O. Gimenez and I. Petrov, “Static Context Header Compression and Fragmentation (SCHC) over LoRaWAN,” Internet Requests for Comments, RFC Editor, RFC 9011, April 2021.

[10] D. Davcev, K. Mitreski, S. Trajkovic, V. Nikolovski, and N. Koteli, “IoT agriculture system based on LoRaWAN,” in *Proc. of 14th IEEE WFCS*, Imperia, Italy, 2018.

[11] U. Grunde and D. Zagars, “LoPy as a building block for internet of things in coastal marine applications,” in *Proc. of 25th Telecommunications Forum*, Belgrade, Serbia, 2017.

[12] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, “Understanding the limits of LoRaWAN,” *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, Sept 2017.

[13] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low Power Wide Area Networks: An Overview," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 855–873, Secondquarter 2017.

[14] LoRa Alliance Technical Committee, "What is LoRaWAN," https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf, Nov 2015.

[15] L. Casals, B. Mir, R. Vidal, and C. Gomez, "Modeling the energy performance of LoRaWAN," *Sensors*, vol. 17, no. 10, 2017.

[16] I. Suciu, X. Vilajosana, and F. Adelantado, "An analysis of packet fragmentation impact in LPWAN," in *Proc. of IEEE WCNC*, Barcelona, Spain, 2018.

[17] Semtech, "SX1272/3/6/7/8: LoRa Modem Designer's Guide AN1200.13," July 2013, accessed: 2019-05-21. [Online]. Available: https://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf

[18] A. Lavric, A. I. Petrariu, and V. Popa, "Long range SigFox communication protocol scalability analysis under large-scale, high-density conditions," *IEEE Access*, vol. 7, pp. 35 816–35 825, 2019.

[19] N. I. Osman and E. B. Abbas, "Simulation and modeling of LoRa and Sigfox low power wide area network technologies," in *Proc. of ICCCEEE*, Guayaquil, Ecuador, Aug 2018.

[20] C. Gomez, J. C. Veras, R. Vidal, L. Casals, and J. Paradells, "A Sigfox energy consumption model," *Sensors*, vol. 19, no. 3, 2019.

[21] Sigfox, "Sigfox Technical Overview," May 2017, accessed: 2019-10-20. [Online]. Available: https://www.disk91.com/wp-content/uploads/2017/05/4967675830228422064.pdf

[22] ——. Radio Configurations. Accessed on 04-05-2021. [Online]. Available: https://build.sigfox.com/sigfox-radio-configurations-rc

[23] B. Martinez, F. Adelantado, A. Bartoli, and X. Vilajosana, "Exploring the Performance Boundaries of NB-IoT," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 03 2019.

[24] A. Larmo, A. Ratilainen, and J. Saarinen, "Impact of CoAP and MQTT on NB-IoT system performance," *Sensors*, vol. 19, p. 7, 12 2018.

[25] Schlienz, J and Raddino, D, "Narrowband internet of things whitepaper,," 2017. [Online]. Available: https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/dl_application/application_notes/1ma266/1MA266_0e_NB_IoT.pdf

[26] H. Mroue, A. Nasser, S. Hamrioui, B. Parrein, E. Motta-Cruz, and G. Rouyer, "MAC layer-based evaluation of IoT technologies: LoRa, SigFox and NB-IoT," in *Proc. of IEEE MENACOMM*, Jounieh, Lebanon, April 2018.

[27] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "Overview of cellular LPWAN technologies for IoT deployment: Sigfox, LoRaWAN, and NB-IoT," in *Proc. of IEEE PerCom Workshops*, Athens, Greece, March 2018.

[28] H. Malik, H. Pervaiz, M. Alam, Y. Le Moullec, A. Kuusik, and M. Imran, "Radio Resource Management Scheme in NB-IoT Systems," *IEEE Access*, vol. PP, 06 2018.

[29] Ingenu, https://www.ingenu.com, accessed: 2019-10-22.

[30] Telensa, https://www.telensa.com, accessed: 2019-10-22.

[31] Qowisio, https://www.qowisio.com, accessed: 2019-10-22.

[32] IEEE, https://www.ieee.org, accessed: 2019-10-22.

[33] ETSI, https://www.etsi.org, accessed: 2019-10-22.

[34] 3GPP, https://www.3gpp.org, accessed: 2019-10-22.

[35] Weightless SIG, https://www.weightless.org, accessed: 2019-10-22.

[36] "Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper." [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[37] C. Gomez, J. Paradells, C. Bormann, and J. Crowcroft, "From 6LoWPAN to 6Lo: Expanding the Universe of IPv6-Supported Technologies for the Internet of Things," *IEEE Communications Magazine*, vol. 55, 12 2017.

[38] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. McCann, and K. Leung, "A survey on the IETF protocol suite for the Internet of Things: Standards, challenges, and opportunities," *Wireless Communications, IEEE*, vol. 20, pp. 91–98, 12 2013.

[39] P. Thubert, "SCHC over PPP," Internet Engineering Task Force, Internet Draft draft-thubert-intarea-schc-over-ppp-03, Apr. 2021.

[40] C. Gomez and A. Minaburo, "Transmission of SCHC-compressed packets over IEEE 802.15.4 networks," Internet Engineering Task Force, Internet Draft draft-gomez-6lo-schc-15dot4-03, Jul. 2022.

[41] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," IETF RFC 4919, August 2007.

[42] G. Montenegro, N. Kushalnagar, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944, September 2007.

[43] G. Papadopoulos, P. Thubert, S. Tsakalidis, and N. Montavont, "RFC 4944: Per-hop Fragmentation and Reassembly Issues," in *Proc. of IEEE CSCN*, Paris, France, Oct. 2018.

[44] E. Kim, D. Kaspar, C. Gomez, and C. Bormann, "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing," Internet Requests for Comments, RFC Editor, RFC 6606, May 2012.

[45] C. Gomez, J. Paradells, C. Bormann, and J. Crowcroft, "From 6LoWPAN to 6Lo: Expanding the Universe of IPv6-Supported Technologies for the Internet of Things," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 148–155, Dec 2017.

[46] B. Moons, A. Karaagac, J. Haxhibeqiri, E. De Poorter, and J. Hoebeke, "Using SCHC for an optimized protocol stack in multimodal LPWAN solutions," in *Proc. of IEEE WF-IoT*, Limerick, Ireland, 04 2019.

[47] W. Ayoub, F. Nouvel, S. Hmede, A. E. Samhat, M. Mroue, and J.-C. Prévotet, "Implementation of SCHC in NS-3 Simulator and Comparison with 6LoWPAN," in *Proc. of 26th ICT*, Hanoi, Vietnam, Apr. 2019.

[48] K. Q. Abdelfadeel, V. Cionca, and D. Pesch, "LSCHC: Layered Static Context Header Compression for LPWANs," in *Proc. of CHANTS, Session: Security & IoT*, New York, USA, 2017.

[49] D. Singh, O. G. Aliu, and M. Kretschmer, "LoRaWAN Evaluation for IoT Communications," in *Proc. of ICACCI*, Bangalore,India, Sep. 2018, pp. 163–171.

[50] T. Petrić, M. Goessens, L. Nuaymi, L. Toutain, and A. Pelov, "Measurements, performance and analysis of LoRa FABIAN, a real-world implementation of LPWAN," in *Proc. of 27th IEEE PIMRC)*, Valencia, Spain, Sep. 2016.

[51] A. Rahman and M. Suryanegara, "The development of IoT LoRa: A performance evaluation on LoS and Non-LoS environment at 915 MHz ISM frequency," in *Proc. of IEEE ICSigSys*, Bali, Indonesia, May 2017, pp. 163–167.

[52] L. Angrisani, P. Arpaia, F. Bonavolontà, M. Conti, and A. Liccardo, "LoRa protocol performance assessment in critical noise conditions," in *Proc. of 3rd IEEE RTSI*, Modena, Italy, Sep. 2017.

[53] D. Zorbas, G. Z. Papadopoulos, P. Maille, N. Montavont, and C. Douligeris, "Improving LoRa Network Capacity Using Multiple Spreading Factor Configurations," in *Proc. of 25th IEEE ICT*, Saint-Malo, France, 2018, pp. 516–520.

[54] OpenSCHC, https://github.com/openschc/openschc, 2019.

[55] C. Gomez, A. Minaburo, L. Toutain, D. Barthel, and J. C. Zuniga, "IPv6 over LPWANs: connecting low power wide area networks to the internet (of things)," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 206–213, 2020.

[56] K. Q. Abdelfadeel, V. Cionca, and D. Pesch, "Dynamic context for static context header compression in LPWANs," in *Proc. of 14th DCOSS*, New York, USA, June 2018.

[57] S. Aguilar, A. Marquet, L. Toutain, C. Gomez, R. Vidal, N. Montavont, and G. Z. Papadopoulos, "LoRaWAN SCHC fragmentation demystified," in *Ad-Hoc, Mobile, and Wireless Networks*, M. R. Palattella, S. Scanzio, and S. Coleri Ergen, Eds., vol. 11803. Springer International Publishing (Cham), 2019, pp. 213–227.

[58] W. Eddy and E. Davies, "Using Self-Delimiting Numeric Values in Protocols," Internet Requests for Comments, RFC Editor, RFC 6256, May 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6256

[59] S. Aguilar, P. Maillé, L. Toutain, C. Gomez, R. Vidal, N. Montavont, and G. Z. Papadopoulos, "Performance analysis and optimal tuning of IETF LPWAN SCHC ACK-on-Error Mode," *IEEE Sensors Journal*, vol. PP, 07 2020.

[60] J. Sanchez-Gomez, J. Gallego-Madrid, R. Sanchez-Iborra, J. Santa, and A. F. Skarmeta, "Impact of SCHC compression and fragmentation in LPWAN: A case study with LoRaWAN," *Sensors*, vol. 20, no. 1, 2020.

[61] I. Suciu, X. Vilajosana, and F. Adelantado, "Aggressive fragmentation strategy for enhanced network performance in dense LPWANs," in *Proc. of IEEE PIMRC*, Bologna, Italy, 2018.

[62] "Sim-RFT." [Online]. Available: https://www.github.com/saguilardevel/sim-RFT

[63] A. Pop, U. Raza, P. Kulkarni, and M. Sooriyabandara, "Does bidirectional traffic do more harm than good in LoRaWAN based LPWA networks?" *CoRR*, vol. abs/1704.04174, 2017. [Online]. Available: http://arxiv.org/abs/1704.04174

[64] M. Capuzzo, D. Magrin, and A. Zanella, “Confirmed traffic in LoRaWAN: Pitfalls and countermeasures,” in *Proc. of 17th Med-Hoc-Net*, 2018.

[65] J. Petäjäjärvi, K. Mikhaylov, M. Hämäläinen, and J. Iinatti, “Evaluation of LoRa LPWAN technology for remote health and wellbeing monitoring,” in *Proc. of 10th ISMICT*, Massachusetts, USA, 2016.

[66] “Best practices.” [Online]. Available: https://www.thethingsindustries.com/docs/devices/best-practices/

[67] J. Haxhibeqiri, A. Karaagac, F. Van den Abeele, W. Joseph, I. Moerman, and J. Hoebeke, “LoRa indoor coverage and performance in an industrial environment: Case study,” in *Proc. of ETFA*, Limassol, Cyprus, 2017.

[68] T. Petrić, M. Goessens, L. Nuaymi, L. Toutain, and A. Pelov, “Measurements, performance and analysis of LoRa FABIAN, a real-world implementation of LPWAN,” in *Proc. of 27th IEEE PIMRC2016*, Valencia, Spain, 2016.

[69] G. Callebaut and L. Van der Perre, “Characterization of LoRa point-to-point path loss: Measurement campaigns and modeling considering censored data,” *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1910–1918, 2020.

[70] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, “A study of LoRa: Long range & low power networks for the internet of things,” *Sensors*, vol. 16, no. 9, 2016.

[71] P. J. Marcelis, V. Rao, and R. V. Prasad, “DaRe: Data Recovery through application layer coding for LoRaWAN,” in *Proc. of 2nd IoTDI*, New York, USA, 2017.

[72] U. Coutaud, M. Heusse, and B. Tourancheau, “Adaptive data rate for multiple gateways LoRaWAN networks,” in *Proc. of 16th WiMob*, Virutal Conference, 2020.

[73] E. Ruano Lin, “LoRa protocol. evaluations, limitations and practical test,” Master’s thesis, Universitat Politecnica de Catalunya, https://upcommons.upc.edu/handle/2117/98853, 5 2016.

[74] U. Coutaud, M. Heusse, and B. Tourancheau, “High reliability in LoRaWAN,” in *Proc. of 31st IEEE PIMR2020*, Virtual Conference, 2020.

[75] R. Sanchez-Iborra, I. G. Liaño, C. Simoes, E. Couñago, and A. F. Skarmeta, “Tracking and monitoring system based on LoRa technology for lightweight boats,” *Electronics*, vol. 8, no. 1, 2019.

[76] S.-Y. Wang, J.-E. Chang, H. Fan, and Y.-H. Sun, “Performance comparisons of NB-IoT, LTE Cat-M1, Sigfox, and LoRa moving at high speeds in the air,” in *Proc. of 25th IEEE ISCC*, Rennes, Francia, 2020.

[77] E. Pramunanto, M. Ulfa, and A. Kurniawan, “Coast panic-emergency situation monitoring system on west and east sailing lane of surabaya using LORAWAN technology,” in *Proc. of CENIM*, Surabaya, Indonesia, 2018, pp. 100–105.

[78] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo, “On the coverage of LPWANs: range evaluation and channel attenuation model for LoRa technology,” in *Proc. of 14th ITST*, Copenhagen, Denmark, 2015, pp. 55–59.

[79] J. Herrera-Tapia, E. Hernández-Orallo, A. Tomás, C. T. Calafate, J.-C. Cano, M. Zennaro, and P. Manzoni, “Evaluating the use of sub-gigahertz wireless technologies to improve message delivery in opportunistic networks,” in *Proc. of 14th IEEE ICNSC*, Calabria, Italy, 2017, pp. 305–310.

[80] M. Sandra, S. Gunnarsson, and A. J. Johansson, “Internet of buoys: An internet of things implementation at sea,” in *Proc. of 54th ACSSC*, California, USA, 2020, pp. 1096–1100.

[81] C. A. Trasviña-Moreno, R. Blasco, A. Marco, R. Casas, and A. Trasviña-Castro, “Unmanned aerial vehicle based wireless sensor network for marine-coastal environment monitoring,” *Sensors*, vol. 17, no. 3, 2017.

[82] L. Sciullo, F. Fossemo, A. Trotta, and M. Di Felice, “LOCATE: A LoRa-based mObile emergenCy mAnagement sysTEm,” in *Proc. of IEEE GLOBECOM*, Abu Dhabi, United Arab Emirates, 2018.

[83] O. A. Saraereh, A. Alsaraira, I. Khan, and P. Uthansakul, “Performance evaluation of UAV-Enabled LoRa networks for disaster management applications,” *Sensors*, vol. 20, no. 8, 2020.

[84] J. J. Kang and S. Adibi, “Bushfire disaster monitoring system using low power wide area networks (LPWAN),” *Technologies*, vol. 5, no. 4, 2017.

[85] V. Gupta, S. K. Devar, N. H. Kumar, and K. P. Bagadi, “Modelling of IoT traffic and its impact on LoRaWAN,” in *Proc. of IEEE GLOBECOM*, Singapore, 2017.

[86] R. Casas, A. Hermosa, A. Marco, T. Blanco, and F. J. Zarazaga-Soria, “Real-time extensive livestock monitoring using LPWAN smart wearable and infrastructure,” *Applied Sciences*, vol. 11, no. 3, 2021.

[87] D. Patel and M. Won, “Experimental study on low power wide area networks (LPWAN) for mobile internet of things,” in *Proc. of 85th IEEE VTC Spring*, Sydney, Australia, 2017.

[88] U. Coutaud and B. Tourancheau, “Channel coding for better QoS in LoRa networks,” in *Proc. of 14th WiMob*, Limassol, Cyprus, 2018.

[89] T. Ameloot, P. Van Torre, and H. Rogier, “Periodic LoRa signal fluctuations in urban and suburban environments,” in *Proc. of 13th EuCAP*, Krakow, Poland, 2019.

[90] J. Haxhibeqiri, F. Van den Abeele, I. Moerman, and J. Hoebeke, “LoRa scalability: A simulation model based on interference measurements,” *Sensors*, vol. 17, no. 6, 2017.

[91] E. D. Ayele, C. Hakkenberg, J. P. Meijers, K. Zhang, N. Meratnia, and P. J. M. Havinga, “Performance analysis of LoRa radio for an indoor iot applications,” in *Proc. of IoTGC*, Funchal, Madeira Island, 2017.

[92] R. Sanchez-Iborra, J. Sanchez-Gomez, J. Ballesta-Viñas, M.-D. Cano, and A. F. Skarmeta, “Performance evaluation of LoRa considering scenario conditions,” *Sensors*, vol. 18, no. 3, 2018.

[93] Semtech, “ Predicting LoRaWAN Capacity.” [Online]. Available: https://lora-developers.semtech.com/library/tech-papers-and-guides/predicting-lorawan-capacity/

[94] T. Attia, M. Heusse, B. Tourancheau, and A. Duda, “Experimental characterization of LoRaWAN link quality,” in *Proc. of IEEE GLOBECOM*, Hawai, USA, 2019.

[95] “What is LoRa?” [Online]. Available: https://www.semtech.com/lora/what-is-lora

[96] Semtech, "SX1272/3/6/7/8: LoRa Modem Designer's Guide AN1200.13," Jul. 2013. [Online]. Available: https://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf

[97] P. Thubert, "IPv6 over low-power wireless personal area network (6LoWPAN) selective fragment recovery," Internet Requests for Comments, RFC Editor, RFC 8931, November 2020.

[98] Aguilar, Sergio and Wistuba, Diego and Platis, Antonis. SCHC over Sigfox Implementation: Network SCHC F/R. Accessed on 04-05-2021. [Online]. Available: https://github.com/saguilarDevel/schc-sigfox

[99] ——. SCHC over Sigfox Implementation: Sender SCHC F/R. Accessed on 04-05-2021. [Online]. Available: https://github.com/saguilarDevel/schcfox

[100] C. Gomez, A. Minaburo, L. Toutain, D. Barthel, and J. C. Zuniga, "IPv6 over LPWANs: Connecting Low Power Wide Area Networks to the Internet (of Things)," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 206–213, Feb. 2020.

[101] OpenSCHC. SCHC Implementation. Accessed on 04-05-2021. [Online]. Available: https://openschc.github.io/openschc/index.html

[102] R. Muñoz Lara, S. Céspedes, and A. Hafid, "Understanding and Characterizing Transmission Times for Compressed IP packets over LoRaWAN," in *Proc. of IEEE LATINCOM*, Santo Domingo, Dominican Republic, 2019.

[103] R. Muñoz Lara, "Modelado y evaluación de la eficiencia del estándar SCHC para el transporte de paquetes IP sobre LoRaWAN," Master's thesis, Universidad de Chile, Santiago, Chile, 2020.

[104] Laurent Toutain. SCHC. Accessed on 04-05-2021. [Online]. Available: https://github.com/ltn22/SCHC

[105] S. Aguilar, R. Vidal, and C. Gomez, "Evaluation of Receiver-Feedback Techniques for Fragmentation over LPWANs," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[106] D. F. Ramírez, S. Céspedes, C. Becerra, and C. Lazo, "Performance evaluation of future AMI applications in Smart Grid Neighborhood Area Networks," in *Proc. of IEEE COLCOM*, Popayan, Colombia, 2015.

[107] LoRa Alliance and DLMS User Association, "A Solution for Successful Interoperability with DLMS/COSEM and LoRaWAN," https://lora-alliance.org/wp-content/uploads/2020/11/dlms-lorawan-whitepaper_v1.pdf, Accessed on 07-10-2021.

[108] Pycom. LoPy4 Datasheet. Accessed on 04-05-2021. [Online]. Available: https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_LoPy4_v2.pdf

[109] ——. SiPy Datasheet. Accessed on 04-05-2021. [Online]. Available: https://docs.pycom.io/.gitbook/assets/specsheets/Pycom_002_Specsheets_SiPy_v2.pdf

[110] Google. Google Cloud Function. Accessed on 04-05-2021. [Online]. Available: https://cloud.google.com/functions

[111] ——. Google Firebase. Accessed on 04-05-2021. [Online]. Available: https://firebase.google.com/

[112] D. Wistuba, S. Céspedes, S. Aguilar, J. C. Zúñiga, C. Gomez, and R. Vidal, "An Implementation of IoT LPWAN SCHC Message Fragmentation and Reassembly," in *Proc. of SSN2020*, Vitoria, Brasil, 12 2020.

[113] Pycom. Sigfox LoPy4 API Reference. Accessed on 04-05-2021. [Online]. Available: https://docs.pycom.io/firmwareapi/pycom/network/sigfox/

[114] J. Santa, L. Bernal-Escobedo, and R. Sanchez-Iborra, “On-board unit to connect personal mobility vehicles to the IoT,” *Procedia Computer Science*, vol. 175, pp. 173–180, Jan. 2020.

[115] S. Aguilar, D. S. W. La-Torre, A. Platis, R. Vidal, C. Gomez, S. Céspedes, and J. C. Zúñiga, “Packet Fragmentation over Sigfox: Implementation and Performance Evaluation of SCHC ACK-on-Error,” *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[116] R. Muñoz, J. Saez Hidalgo, F. Canales, D. Dujovne, and S. Céspedes, “SCHC over LoRaWAN Efficiency: Evaluation and Experimental Performance of Packet Fragmentation,” *MPDI Sensors*, vol. 22, no. 4, p. 1531, Jan. 2022.

[117] B. Martinez, M. Montón, I. Vilajosana, and J. D. Prades, “The Power of Models: Modeling Power Consumption for IoT Devices,” *IEEE Sensors Journal*, vol. 15, no. 10, pp. 5777–5789, Oct. 2015, conference Name: IEEE Sensors Journal.

[118] P. Ruckebusch, S. Giannoulis, I. Moerman, J. Hoebeke, and E. De Poorter, “Modelling the energy consumption for over-the-air software updates in LPWAN networks: SigFox, LoRa and IEEE 802.15.4g,” *Internet of Things*, vol. 3-4, pp. 104–119, Oct. 2018.

[119] D. M. Hernandez, G. Peralta, L. Manero, R. Gomez, J. Bilbao, and C. Zubia, “Energy and coverage study of lpwan schemes for industry 4.0,” in *Proc. of IEEE ECMSM*, Donostia-San Sebastian, Spain, 2017.

[120] E. Morin, M. Maman, R. Guizzetti, and A. Duda, “Comparison of the Device Lifetime in Wireless Networks for the Internet of Things,” *IEEE Access*, vol. 5, pp. 7097–7114, 2017.

[121] T. Ogawa, T. Yoshimura, and N. Miyaho, “Cloud control dtn utilizing general user’ smartphones for narrowband edge computing,” in *Proc. of IEEE 4th WF-IoT*, Singapore, 2018, pp. 19–24.

[122] A. Perles, E. Pérez-Marín, R. Mercado, J. D. Segrelles, I. Blanquer, M. Zarzo, and F. J. Garcia-Diego, “An energy-efficient internet of things (IoT) architecture for preventive conservation of cultural heritage,” *Future Generation Computer Systems*, vol. 81, pp. 566–581, Apr. 2018.

[123] J. Finnegan and S. Brown, “An Analysis of the Energy Consumption of LPWA-based IoT Devices,” in *Proc. of IEEE ISNCC*, Rome, Italy, Jun. 2018.

[124] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, “Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT,” in *Proc. of IEEE PerCom Workshops*, Athens, Greece, Mar. 2018.

[125] Y. Lykov, A. Paniotova, V. Shatalova, and A. Lykova, “Energy Efficiency Comparison LPWANs: LoRaWAN vs Sigfox,” in *Proc. of IEEE PIC S and T*, Kharkiv, Ukraine, Oct. 2020, pp. 485–490.

[126] “State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally,” May 2022. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/

[127] C. S. Alliance, “Matter Specification Version 1.0,” Sep. 2022. [Online]. Available: https://csa-iot.org/wp-content/uploads/2022/11/22-27349-001_Matter-1.0-Core-Specification.pdf

[128] T. Zachariah, N. Klugman, B. Campbell, J. Adkins, N. Jackson, and P. Dutta, "The Internet of Things Has a Gateway Problem," in *Proc. of the 16th International Workshop on Mobile Computing Systems and Applications, HotMobile '15.* New York, NY, USA: Association for Computing Machinery, Feb. 2015, pp. 27–32.

[129] M. C. Caballé, A. C. Augé, and J. P. Aspas, "Wake-Up Radio: An Enabler of Wireless Convergence," *IEEE Access*, vol. 9, pp. 3784–3797, 2021.

[130] K. Chebrolu and A. Dhekne, "Esense: Energy Sensing-Based Cross-Technology Communication," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2303–2316, Nov. 2013.

[131] S. M. Kim and T. He, "FreeBee: Cross-technology Communication via Free Side-channel," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, MobiCom '15.* Paris, France: Association for Computing Machinery, Sep. 2015, pp. 317–330.

[132] Z. Li and T. He, "WEBee: Physical-Layer Cross-Technology Communication via Emulation," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom '17.* New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 2–14.

[133] ——, "LongBee: Enabling Long-Range Cross-Technology Communication," in *Proc. of IEEE INFOCOM*, Honolulu, USA, Apr. 2018, pp. 162–170.

[134] Z. Chi, Y. Li, H. Sun, Yao, and T. Zhu, "Concurrent Cross-Technology Communication Among Heterogeneous IoT Devices," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 932–947, Jun. 2019.

[135] X. Guo, X. Zheng, and Y. He, "WiZig: Cross-technology energy communication over a noisy channel," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, iSSN: null.

[136] X. Zheng, Y. He, and X. Guo, "StripComm: Interference-Resilient Cross-Technology Communication in Coexisting Environments," in *Proc. of IEEE INFOCOM.* Honolulu, USA: IEEE Press, Apr. 2018, pp. 171–179.

[137] Z. Yin, W. Jiang, S. M. Kim, and T. He, "C-Morse: Cross-technology communication with transparent Morse coding," in *Proc. of IEEE INFOCOM*, Atlanta, GA, USA, May 2017.

[138] W. Jiang, Z. Yin, S. M. Kim, and T. He, "Transparent cross-technology communication over data traffic," in *Proc. of IEEE INFOCOM*, Atlanta, GA, USA, May 2017.

[139] S. Wang, Z. Yin, Z. Li, and T. He, "Networking Support For Physical-Layer Cross-Technology Communication," in *Proc. of IEEE 26th ICNP*, Cambridge, UK, Sep. 2018, pp. 259–269, iSSN: 1092-1648.

[140] W. Ayoub, M. Mroue, A. E. Samhat, F. Nouvel, and J.-C. Prévotet, "SCHC-Based Solution for Roaming in LoRaWAN," in *Proc. of 26th ICT*, Anvers, Belgium, Nov. 2019.

[141] J. Famaey, R. Berkvens, G. Ergeerts, E. D. Poorter, F. V. d. Abeele, T. Bolckmans, J. Hoebeke, and M. Weyn, "Flexible Multimodal Sub-Gigahertz Communication for Heterogeneous Internet of Things Applications," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 146–153, Jul. 2018.

[142] J. Hoebeke, J. Haxhibeqiri, B. Moons, M. Van Eeghem, J. Rossey, A. Karagaac, and J. Famaey, "A Cloud-based Virtual Network Operator for Managing Multimodal LPWA Networks and Devices," in *Proc. of IEEE CIoT*, Paris, France, Jul. 2018.

[143] R. Sanchez-Iborra, J. Sánchez-Gómez, S. Perez, P. J. Fernández, J. Santa, J. L. Hernández-Ramos, and A. F. Skarmeta, "Internet Access for LoRaWAN Devices Considering Security Issues," in *Proc. of IEEE GIoTS*, Bilbao, Spain, Jun. 2018.

[144] B. Moons, E. De Poorter, and J. Hoebeke, "Device Discovery and Context Registration in Static Context Header Compression Networks," *MDPI Information*, vol. 12, no. 2, p. 83, Feb. 2021.

[145] S. Aguilar, A. Platis, R. Vidal, and C. Gomez, "Energy Consumption Model of SCHC Packet Fragmentation over Sigfox LPWAN," *MPDI Sensors*, vol. 22, no. 6, p. 2120, Jan. 2022.

[146] C. Bormann and P. Hoffman, "Concise binary object representation (cbor)," Internet Requests for Comments, RFC Editor, RFC 7049, October 2013.

[147] R. Moskowitz, S. W. Card, and A. Wiethuechter, "Internet protocol number for schc," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-intarea-schc-ip-protocol-number-00, October 2022. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-intarea-schc-ip-protocol-number-00.txt