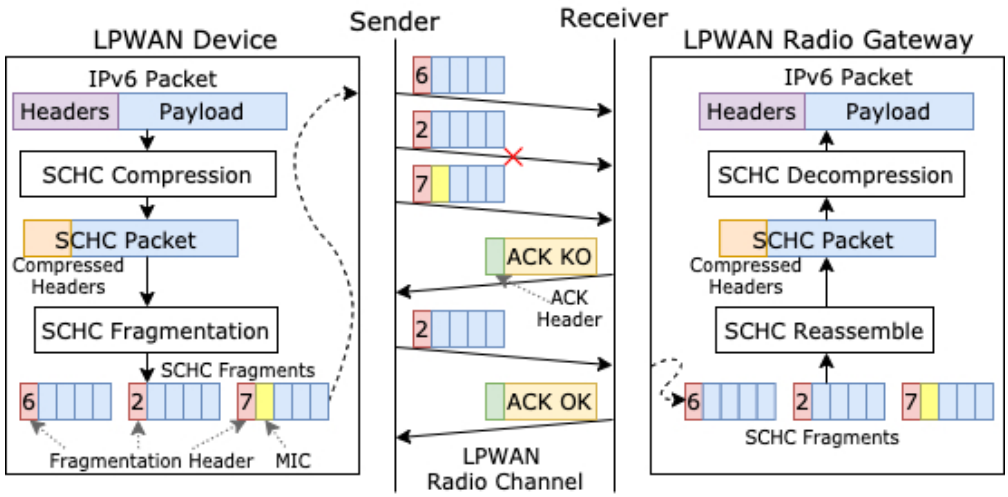
Performance Analysis and Optimal Tuning of IETF LPWAN SCHC ACK-on-Error Mode

Sergio Aguilar, Patrick Maillé, Laurent Toutain, Carles Gomez, Rafael Vidal, Nicolas Montavont, Georgios Papadopoulos

Performance Analysis and Optimal Tuning of IETF LPWAN SCHC ACK-on-Error Mode

Journal:	*IEEE Sensors Journal*
Manuscript ID	Draft
Manuscript Type:	Regular Paper
Date Submitted by the Author:	n/a
Complete List of Authors:	Aguilar, Sergio; Universitat Politecnica de Catalunya, Network Engineering Maillé, Patrick; IMT Atlantic Brittany-Pays de Loire - Rennes Campus Toutain, Laurent; IMT Atlantic Brittany-Pays de Loire - Rennes Campus Gomez, Carles; Universitat Politecnica de Catalunya Vidal Ferré, Rafael; Universitat Politecnica de Catalunya, Network Engineering Montavont, Nicolas; IMT Atlantic Brittany-Pays de Loire - Rennes Campus Papadopoulos, Georgios Z. ; TELECOM Bretagne, RSM; Télécom Bretagne
Keywords:	NETW



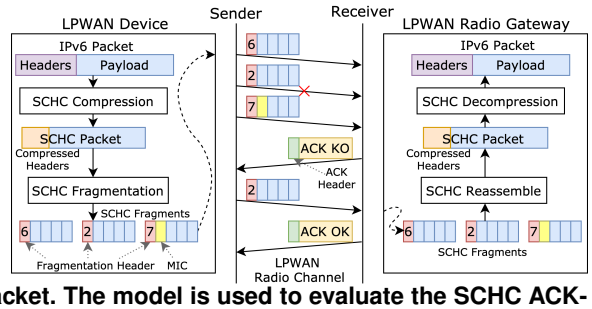
194x96mm (72 x 72 DPI)

IEEE Sensors Council IEEE SENSORS JOURNAL, VOL. XX, NO. XX, XXXX 2017 1

Performance Analysis and Optimal Tuning of IETF LPWAN SCHC ACK-on-Error Mode

Sergio Aguilar, Patrick Maillé, Laurent Toutain, Carles Gomez, Rafael Vidal, Nicolas Montavont, Georgios Z. Papadopoulos

***Abstract*— The Internet Engineering Task Force (IETF) Low Power Wide Area Network (LPWAN) Working Group has developed the Static Context Header Compression (SCHC) framework to enable IPv6 over LPWAN. In order to support 1280-byte packets, as required for IPv6, SCHC includes a fragmentation functionality, since relevant LPWAN technologies offer very short data unit sizes and do not provide native fragmentation mechanisms. SCHC offers 3 fragmentation modes: No-ACK, ACK-Always, and ACK-on-Error, the latter being especially promising due to its reliability and high efficiency. In this article, we develop a mathematical model to compute the most critical performance parameters for the SCHC ACK-on-Error mode, namely, the acknowledgment 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.**



***Index Terms*— LPWAN, SCHC, ACK-on-Error, LoRaWAN, Sigfox, IoT, LoRa, IETF, mathematical model, fragmentation**

I. Introduction

LOW Power Wide Area Networks (LPWANs) refer to network technologies designed for the Internet of Things (IoT) that are characterized by a long-range and low-energy operation [1–3]. LPWAN technologies are based on star topology deployments, where a potentially high number of Internet of Things (IoT) devices are directly connected to a radio gateway.

In some quintessential LPWAN technologies, such as LoRaWAN and Sigfox, the layer 2 Maximum Transmission Unit (L2 MTU) ranges from tens to hundreds of bytes [1]. To carry IPv6 packets over LPWAN despite those limitations, the Internet Engineering Task Force (IETF) has defined a new adaptation layer called Static Context Header Compression (SCHC) [4], [5], which along with a header compression functionality, provides fragmentation mechanisms to transport an IPv6 packet over several LPWAN frames. SCHC defines 3 Fragmentation/Reassembly (F/R) modes called No-ACK, ACK-Always and ACK-on-Error. This paper focuses on the ACK-on-Error mode, which is promising due to its reliability and high efficiency by minimizing the number of acknowledgments (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, that can be especially binding for gateways, which may 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 is limited [6]. 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 paper 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 [7].

More specifically, in this paper we propose 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

This work was supported in part by the Spanish Government through project TEC2016-79988-P, AEI/FEDER, EU.

S.Aguilar, C. Gomez and R. Vidal are with the Department of Network Engineering, Universitat Politècnica de Catalunya, 08860 Castelldefels, Barcelona, Spain (e-mail: sergio.aguilar.romero@upc.edu,{carlesgo, rafael.vidal}@entel.upc.edu)

P. Maillé, L. Toutain, N. Montavont and G. Z. Papadopoulos are with the Department of Network Systems, Cyber Security and Digital Law (SCRD), IMT-Atlantique, IRISA, 35576 Campus of Rennes, France (e-mail: {patrick.maille, laurent.toutain, nicolas.montavont, georgios.papadopoulos}@imt-atlantique.fr)

[1]See, e.g., `https://iotmarket.orange.com/connectivity.html` (accessed on 30/06/2019)

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 the specifics of LoRaWAN [8] and Sigfox [9] technologies.

The remainder of this paper is organized as follows. In Section II we first present the existing works related to LPWAN fragmentation and its performance. We then detail the SCHC framework in Section III, and the SCHC F/R modes in Section IV, especially focusing on the ACK-on-Error mode. The mathematical model used to analyze the ACK-on-Error mode is developed in Section V, and is it used in Section VI to evaluate the performance metrics, mostly regarding the amount of ACK messages. Section VII explains how our results can be applied to state-of-the-art LPWAN technologies such as LoRaWAN and Sigfox. Finally, we provide some conclusions in Section VIII.

II. Related Work

Recent attention on LPWAN technologies has partly focused on the evaluation of the physical layers [6], [7], [10], [11]. In [7] the authors analyze LoRaWAN and explore its limitations. The results show 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 works provide a mathematical model that characterizes LoRaWAN and Sigfox end-device energy consumption, lifetime and energy involved in data delivery [6], [10]. In [11] a performance evaluation of Sigfox scalability is presented. Together, these studies provide important insights into the physical layer of LPWAN technologies, but do not consider the compatibility with IPv6, nor fragmentation mechanisms.

Several studies compare different LPWAN technologies [12–14]. While Mroue et al. [12] 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 [14]. The study in [13] 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 [15] evaluates the effect of fragmentation, and its efficiency in terms of energy consumption, throughput, goodput and end time delay in dense networks. Suciu et al [15] showed that fragmentation increases reliability, especially when sending several fragments instead of only one of the MTU size. Other works focus on IPv6 over LPWAN by means of using SCHC [5], [16–20]. Some of these propose enhancements to SCHC Header compression, but do not consider SCHC F/R [18], [19]. On the other hand, Moons et al. [16] compared SCHC and 6LoWPAN compression and fragmentation functionalities. Their results show that SCHC has a smaller footprint, uses less memory and the header overhead is twenty times smaller when compared with 6LoWPAN. Ayoub et al. [17] 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 [5] provided an overview of SCHC and a simple evaluation of the different F/R modes, but it is a superficial study due to its tutorial purpose. In [20], 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, the latter provided 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.

III. Technical Background: SCHC Fragmentation and Reassembly

This section provides an overview of SCHC F/R. We first introduce the SCHC adaptation layer, then focus on the main SCHC F/R components and tools.

A. 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 compression and F/R functionalities specifically designed for LPWAN [4]. 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 Compression and the Fragmentation sublayers. Fig. 1 shows those sublayers between the IPv6 layer and the LPWAN technology layer. When an IPv6 packet needs to be sent, compression is

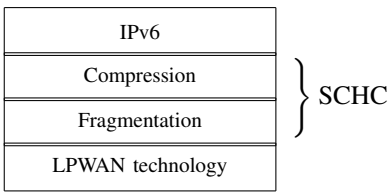


Fig. 1. Protocol stack illustrates the location of the SCHC sublayers between the IPv6 layer and the underlying LPWAN technology [4].

performed. The compressed IPv6 is called a SCHC Packet. If the SCHC Packet size is greater than the L2 MTU, SCHC fragmentation is performed at the sender. At the receiver, the SCHC Packet is reassembled and the IPv6 packet is decompressed.

In SCHC F/R, a SCHC Packet is fragmented into units called ***tiles***. 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***, 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).

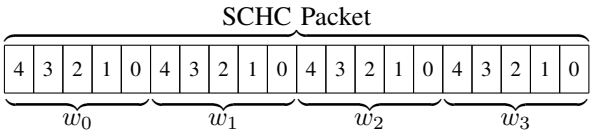


Fig. 2. Example of a SCHC Packet fragmented into 20 tiles, with 5 tiles per window. The tile index in the window, called the Fragment Compressed Number (FCN) is indicated for each tile.

B. 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 number identifies the window 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.
- Message Integrity Check (MIC): 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 MIC for this SCHC Packet. Fig. 3 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 III-E). Fig. 4 shows the SCHC ACK format. A SCHC ACK carries the C field.

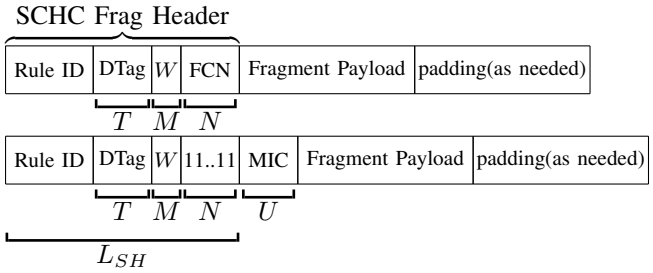


Fig. 3. Illustration of a regular SCHC Fragment, and an All-1 SCHC Fragment with the MIC field. The length in bits of each header field is indicated below each field.

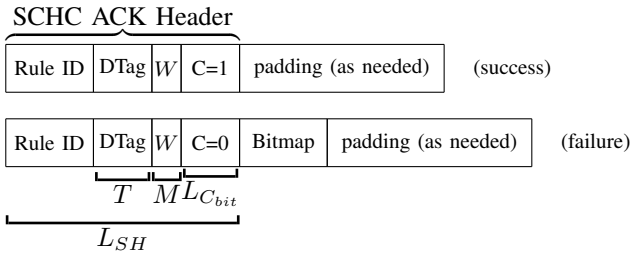


Fig. 4. Illustration of a SCHC ACK message. The top SCHC ACK Message notifies successful reassembly of a SCHC Packet by carrying a $C = 1$. The bottom one indicates a failed SCHC Packet reassembly ($C = 0$) and carries a bitmap. The length of each header field (in bits) is indicated below each field.

C. Tiles

As previously explained (see Fig. 2), a SCHC Packet is fragmented into units called tiles, which have a size of t bytes. In ACK-on-Error mode, each tile must be of the same size, except for the last one, which can be smaller. In the No-ACK and ACK-Always modes, tiles can be of different sizes. If the payload field is present in a SCHC Fragment, it must carry at least one tile. In ACK-on-Error mode, each tile of a SCHC Packet is uniquely identified by the window and tile numbers. The FCN of the SCHC Fragment, together with the window index W, identifies the first tile carried by the SCHC Fragment.

D. 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 shows the fragmentation of a SCHC Packet in 4 windows, with 5 tiles per window.

E. 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. 5. The C field is set to 0, indicating the packet reassembly was not successful, since the SCHC Packet was not received completely.

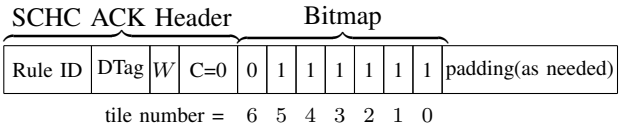


Fig. 5. Example of a SCHC ACK, where a window of 7 tiles ($\boldsymbol{w = 7}$) was sent and tile FCN # 6 was not received successfully.

IV. 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.

A. The 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. Variable L2 MTU size is supported. Tiles can be of different sizes, while windows are not used.

B. ACK-Always mode

The ACK-Always mode is a window-based mechanism for in-sequence delivery of SCHC Fragments that supports reliability. 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.

C. 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 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.

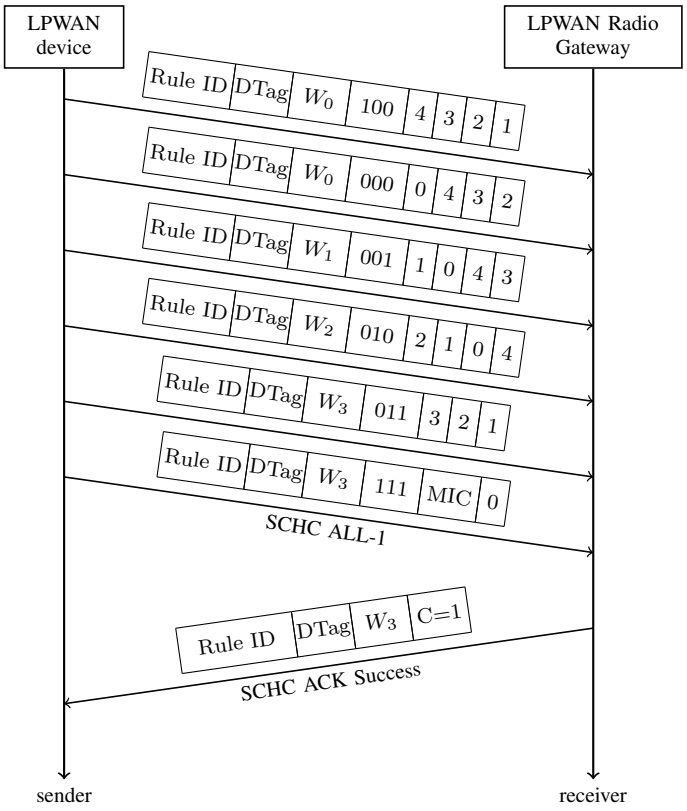


Fig. 6. Example of a transmission of the SCHC Packet shown in Fig. 2 with no errors. The transmission ends with a SCHC ACK that indicates successful SCHC Packet reassembly.

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, finally when an inactivity timer at the receiver indicates that the transmission has been inactive for too long. Fig. 6 shows the transmission of the fragmented SCHC Packet of Fig. 2 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 MIC calculated with the MIC 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. 7 shows the transmission of the SCHC Packet presented in Fig. 2 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 MIC, performs the integrity check and sends a SCHC ACK reporting successful SCHC Packet reassembly.

V. 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.

For the 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. 4) 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.

A. 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 will depend 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_{success}) for a SCHC Fragment can be related to the Bit Error Rate (BER) through the relation

$$P_{\text{success}} = (1 - BER)^{8F}, \tag{1}$$

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 V-C.) Assuming P_{success} fixed and all transmissions 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.,

$$\mathbb{P}(N_{\text{TA}} = n) = (1 - P_{\text{success}})^{n-1} P_{\text{success}}. \tag{2}$$

Recall that at most only one ACK per window is generated: 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.

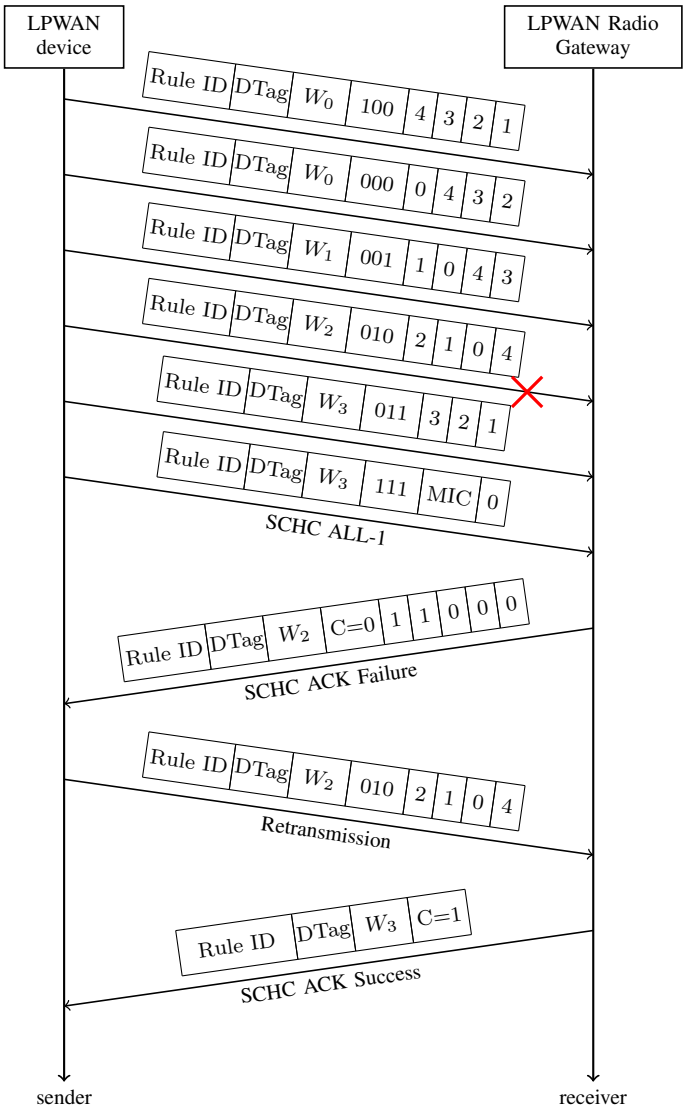


Fig. 7. Example of a transmission of a SCHC Packet with one lost SCHC Fragment in ACK-on-Error mode. Once the All-1 SCHC Message is received by the receiver, a SCHC ACK with $C = 0$ and its corresponding bitmap is generated. The sender performs the SCHC Fragment retransmission. When the lost SCHC Fragment 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.

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, denote by A_j 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 $\binom{j}{i} P_{\text{success}}^{j-i}(1-P_{\text{success}})^i$, 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 simply A_i, hence the recursive relation

$$A_j = 1 + \sum_{i=0}^{j} \binom{j}{i} P_{\text{success}}^{j-i} (1 - P_{\text{success}})^i A_i, \tag{3}$$

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

$$A_j = \frac{1}{1+(1-P_{\text{success}})^j} \left[1 + \sum_{i=0}^{j-1} \binom{j}{i} P_{\text{success}}^{j-i} (1 - P_{\text{success}})^i A_i \right] \tag{4}$$

Using (4) with $A_0 = 0$, the number of ACKs required per window is simply $A_k - 1$, removing the last success 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 in a 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:

$$k = \begin{cases} k_1 = \lfloor \frac{w}{f} \rfloor, & \text{with probability } 1 + \lfloor \frac{w}{f} \rfloor - \frac{w}{f} \\ k_2 = \lceil \frac{w}{f} \rceil, & \text{with probability } \frac{w}{f} - \lfloor \frac{w}{f} \rfloor, \end{cases} \tag{5}$$

as illustrated in Fig. 8.

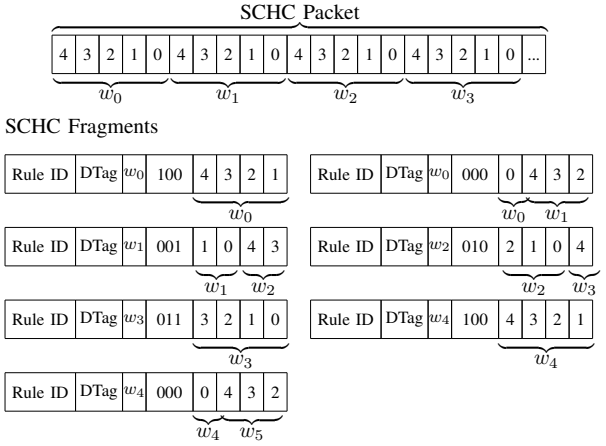


Fig. 8. Example of a SCHC Packet and SCHC Fragments. The SCHC Packet is fragmented in 5 tiles per window ($\boldsymbol{w = 5}$) and the SCHC Fragments can transport 4 tiles per fragment ($\boldsymbol{f = 4}$). For a successful transmission of the packet, $\boldsymbol{w_0}$ needs $\boldsymbol{k_2 = 2}$ successful SCHC Fragment transmissions, then $\boldsymbol{w_1, w_2,}$ and $\boldsymbol{w_3}$ need each $\boldsymbol{k_1 = 1}$ more successful SCHC Fragment transmission, in compliance with (5): a proportion $\boldsymbol{\frac{w}{f} - \lfloor \frac{w}{f} \rfloor = 1/4}$ of windows need $\boldsymbol{k_2 = \lceil \frac{w}{f} \rceil}$ new SCHC Fragment transmissions, while the other windows need only $\boldsymbol{k_1 = \lfloor \frac{w}{f} \rfloor = 1}$ 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), and E_k is the weighted average of $A_{k_1} - 1$ and $A_{k_2} - 1$ with the weights of (5). This gives us the E_k for a given window size, fragment size and tile size, as follows:

$$E_k = \left(1 + \lfloor \tfrac{w}{f} \rfloor - \tfrac{w}{f}\right) \cdot A_{\lfloor \frac{w}{f} \rfloor} - 1 + \left(\tfrac{w}{f} - \lfloor \tfrac{w}{f} \rfloor\right) \cdot A_{\lceil \frac{w}{f} \rceil} - 1. \tag{6}$$

B. 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:

$$O_{\text{ACK}} = \frac{(8L_{\text{SH}} + w) \cdot E_k}{8wt}. \tag{7}$$

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 produces smaller bitmaps, thus smaller ACKs. Recall that the O_{ACK} is from the point of view of the SCHC framework layer, as it only considers the SCHC Headers.

C. ACK Bit Overhead with L2 Headers

As the SCHC framework sits on top of a LPWAN technology, there is an extra overhead due to L2 Headers. 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:

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

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 while optimizing the ACK size and volume, reducing the utilization of the LPWAN gateway and leveraging the available resources in *duty-cycle*-constrained networks.

D. 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 technologies 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, equals

$$P_U = \frac{f \cdot t}{F - L_{\text{SH}}} \cdot 100. \tag{9}$$

In LPWAN technologies such as LoRaWAN, F can change during an on-going fragmented packet transmission [8] 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 calculated as follows:

$$f_{\text{unused}} = (F - L_{\text{SH}}) - (f \cdot t). \tag{10}$$

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 and $F = 53$, 6 bytes per SCHC Fragment will not be used.

E. 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 SCHC Fragment, i.e. 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 data on a window. The maximum bitmap size in tiles, i.e., number of bits in the bitmap, is therefore calculated as follows:

$$\text{MBS} = 8 \cdot (F - L_{\text{SH}}), \tag{11}$$

and the maximum window size in bytes is then:

$$\text{MWS} = (t \cdot \text{MBS}) - U. \tag{12}$$

VI. Performance Evaluation

In this section, we use the models derived in section V to evaluate the ACK-on-Error mode in terms of the performance metrics presented.

The fragment sizes F used in the performance evaluation are 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. Note that the physical layer configuration in LoRaWAN (including data unit size, and thus fragment size) is determined by the Data Rate (DR) and Spreading Factor (SF) [8]. The SCHC ACK and SCHC Fragment Header size used is 2 bytes ($L_{\text{SH}} = 2$).

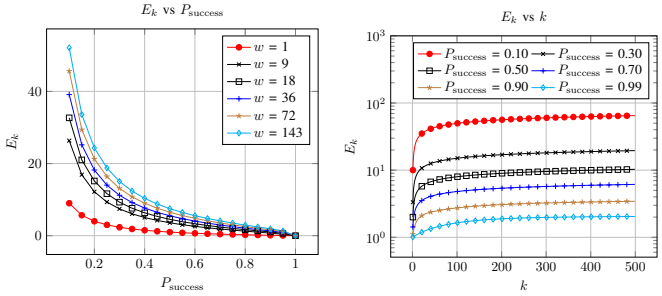
A. ACK Message Overhead

Fig. 9a, shows the ACK message overhead, (E_k) as a function of P_{success} for different window sizes for $F = 11$ and $t = 9$ bytes so that $f = 1$ (one tile per fragment), $k = w$, and no unused bits. 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 lower P_{success} than with higher P_{success}. This happens because E_k has a logarithmic behavior as a function of k (see Fig. 9b) and "flattens" when P_{success} increases: for small values of k and P_{success}, a small variation in k produces large changes in E_k. The E_k variation decreases as k increases. For higher P_{success}, the difference in E_k is less dependent on k, since less ACKs are produced.

For $P_{\text{success}} = 1$, 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.

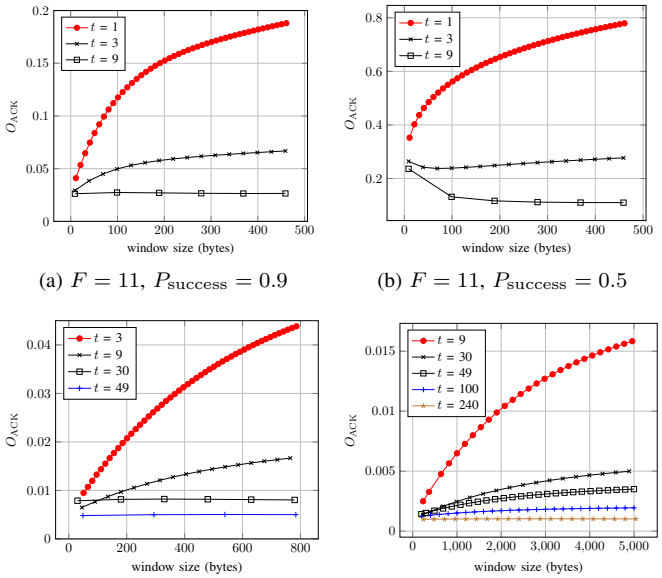
B. ACK Bit Overhead

We now consider the ACK bit overhead metric O_{ACK} defined in (7), with the aim of minimizing that metric.



(a) (b)

Fig. 9. E_k for $F = 11$, $t = 9$ and $L_{\text{SH}} = 2$ for different windows sizes as a function of P_{success} (a) and a function of k (b).



(a) $F = 11$, $P_{\text{success}} = 0.9$ (b) $F = 11$, $P_{\text{success}} = 0.5$

(c) $F = 51$, $P_{\text{success}} = 0.9$ (d) $F = 242$, $P_{\text{success}} = 0.9$

Fig. 10. O_{ACK} vs window size. Impact of the tile size t on the O_{ACK}. $L_{\text{SH}} = 2$.

1) Optimal Tile Size: Fig. 10a shows the impact of the tile size on O_{ACK}, for $F = 11$ and $P_{\text{success}} = 0.9$. For small tiles, O_{ACK} increases faster with the window size than for large tiles; indeed a smaller tile size will require a larger bitmap for each data bit transferred, since the bitmap size equals the number of tiles in a window, hence is inversely proportional to the tile size for a fixed window size (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. 9b) 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 $F = 51$ and $F = 242$ for $P_{\text{success}} = 0.9$ are shown in Fig. 10c and Fig. 10d, respectively. 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. 10d shows.

Fig. 10b shows O_{ACK} for $P_{\text{success}} = 0.5$ 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_{success}. Hence, for technologies with fixed MTU such as Sigfox, the optimal tile size is simply the SCHC Fragment payload size. For technologies with variable 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 MTU. In the case the SCHC Fragment size is known beforehand, the Rule can be chosen with the optimal tile and windows sizes. If the MTU changes, then the tile size can change accordingly.

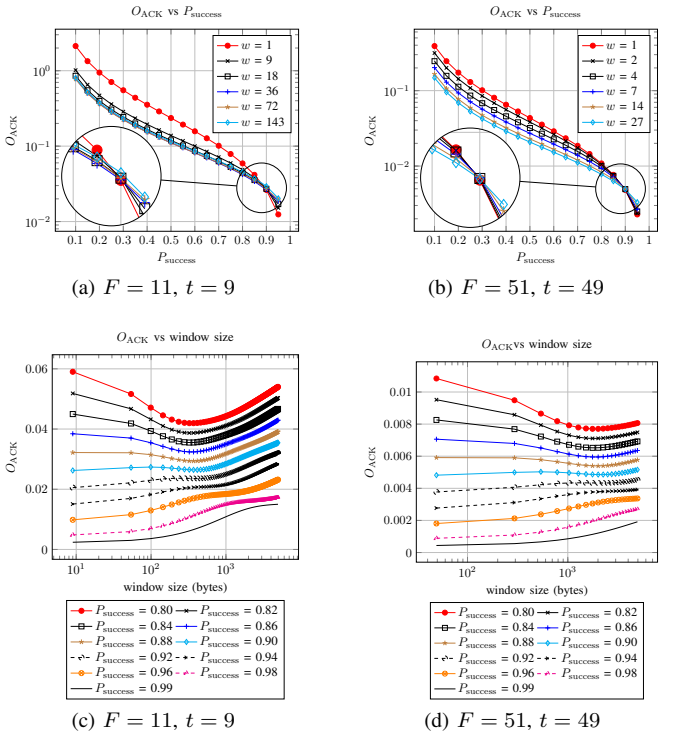
2) Optimal Window Size: The tile size is set to a fixed value, i.e., equal to the SCHC Fragment payload size. Fig. 11a and Fig. 11b illustrate the O_{ACK} as a function of P_{success} for different window sizes, for $F = 11$, $t = 9$ and $F = 51$, $t = 49$, respectively. When P_{success} is low, many 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_{success} 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_{success} increases, there exists one point (see zoom in Fig. 11a and Fig. 11b) beyond which smaller windows outperform larger window sizes. From this point forward, having smaller ACKs becomes more important since the number of losses is low. For example, the window size of 1 tile performs worst for lower P_{success} but as P_{success} increases, it becomes the one yielding the lowest O_{ACK}. When P_{success} is greater than 0.9, the window size of just 1 tile is optimal because rarely a SCHC Fragment will get 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_{success}, but also on F and t. Fig. 11c and Fig. 11d show the values for O_{ACK} for different windows sizes, and for $F = 11$, $t = 9$ and $F = 51$, $t = 49$, respectively. When P_{success} 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. 11c. As P_{success} increases, O_{ACK} increases with the window size and for $P_{\text{success}} \geq 0.9$ the benefit of ACK pooling is lost, leading to an optimal window size of one tile ($w = 1$).



(a) $F = 11$, $t = 9$ (b) $F = 51$, $t = 49$

(c) $F = 11$, $t = 9$ (d) $F = 51$, $t = 49$

Fig. 11. Impact of the window size w (in tiles) or $8wt$ (in bits) on the ACK overhead O_{ACK}. The window size is a multiple of the tile size using w. $L_{\text{SH}} = 2$.

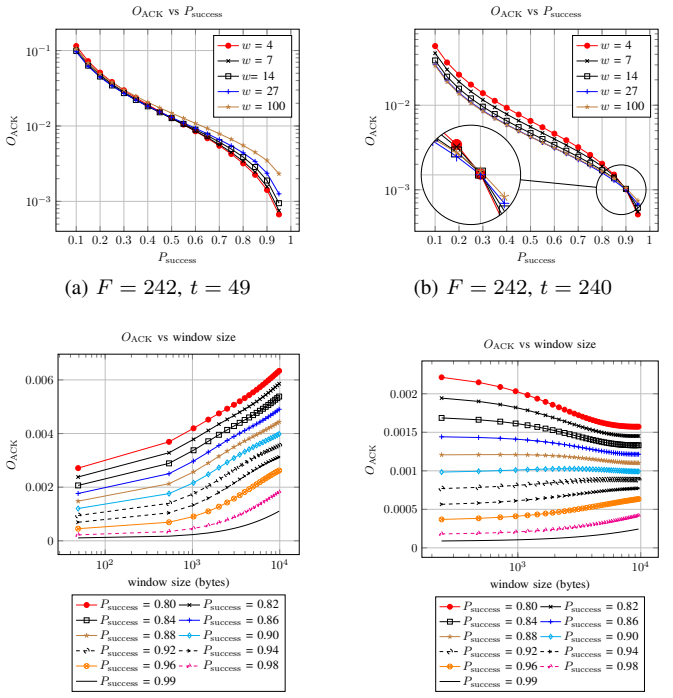
Fig. 12a and Fig. 12b illustrate the results for $F = 240$ with $t = 49$ and $t = 240$, respectively. When the tile size is small compared with the SCHC Fragment payload size ($f > 1$) there is no gain from ACK pooling for $P_{\text{success}} > 0.5$. Hence smaller windows perform better, as Fig. 12a shows. As the window size increases, more small tiles are required when compared with larger tile sizes (see Fig. 12b), making the larger windows size yield a greater O_{ACK} when using smaller tiles. Fig. 12c shows how O_{ACK} is minimum for small window sizes when $P_{\text{success}} > 0.80$ for $F = 240$ and $t = 49$: the optimal window size is the smaller window, i.e. $w = 4$. By contrast, Fig. 12d illustrates for $F = 242$ and $t = 240$ how a larger tile size will yield a larger optimal window size and benefit from ACK pooling for $P_{\text{success}} < 0.9$.

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. 13a presents the optimal window size as a function of P_{success}, for different F values. When P_{success} is below 0.8, in most of the cases, the optimal window size is large. The optimal window size decreases as P_{success} increases, as expected the importance of ACK pooling decreasing with respect to the ACK size. Fig. 13b presents the optimal window sizes for different P_{success} and for larger F values. As for the case of smaller F, as P_{success} becomes higher, the optimal window size becomes smaller.

C. ACK Bit Overhead with L2 Headers

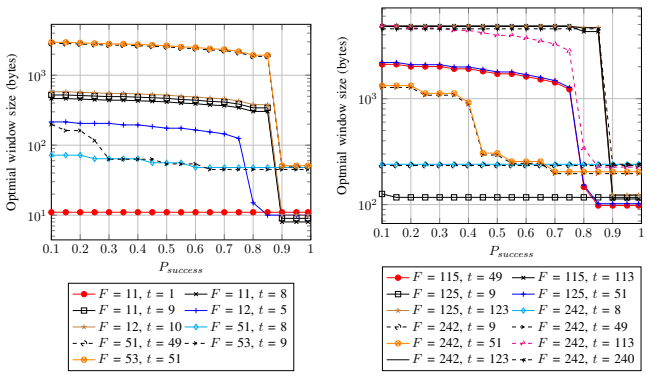
Previous figures were considering the ACK overhead as defined in (7), i.e., ignoring L2 headers in the size of ACKs.



(a) $F = 242$, $t = 49$ (b) $F = 242$, $t = 240$

(c) $F = 242$, $t = 49$ (d) $F = 242$, $t = 240$

Fig. 12. Impact of the window size $\boldsymbol{w}$ (in tiles) or $\boldsymbol{8wt}$ (in bits) on the ACK overhead $\boldsymbol{O_{ACK}}$. The window size is a multiple of the tile size using $\boldsymbol{w}$. $\boldsymbol{L_{SH} = 2}$.



(a) $F = 11, 12, 51, 53$ (b) $F = 115, 125, 242$

Fig. 13. Optimal window size vs $\boldsymbol{P_{success}}$ for different settings.

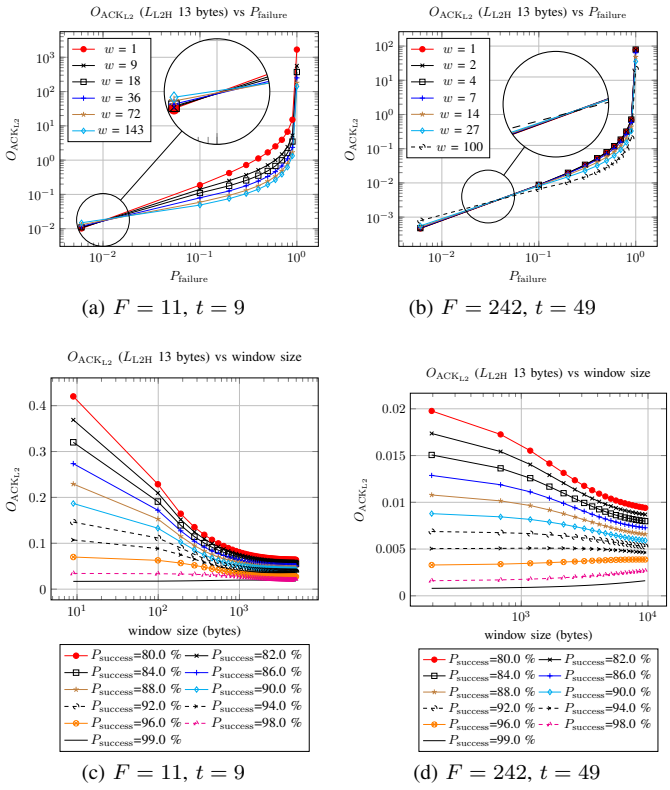
This may be justified if the operator charges the user based on the volume of L2 payloads. In this section we investigate the impact of counting the whole L2 ACK size by counting those headers, as suggested in (8).

The L2 headers of Sigfox according to [6] is 21 bytes in downlink with a payload of 8 bytes. The L2 headers of LoRaWAN according to [8] are 13 byte long, hence we will especially focus on those values.

For clarity in the figures, we define P_{failure} as $1 - P_{\text{success}}$. Fig 14a and Fig. 14b illustrate $O_{\text{ACK}_{\text{L2}}}$ for $F = 11$, $t = 9$ and for $F = 242$, $t = 49$, with $L_{\text{L2H}} = 13$, respectively.

Smaller windows outperform larger ones for $P_{\text{success}} > 0.99$ (see the zoom in Figs. 14a and 14b), whereas in Section VI-B it is for $P_{\text{success}} > 0.90$. 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 $P_{\text{success}} > 0.99$, i.e., $P_{\text{failure}} < 10^{-2}$, does the ACK length effect take over the ACK pooling effect, as Fig. 14a shows. For $P_{\text{success}} < 0.99$, the best window size is 1 tile ($w = 1$), but as P_{success} decreases, the optimal window size increases (e.g. $w = 143$ for $P_{\text{success}} > 0.99$).

Fig. 14c and Fig. 14d confirm our conclusions, by showing $O_{\text{ACK}_{\text{L2}}}$ for $F = 11$, $t = 9$ and for $F = 242$, $t = 49$ with $L_{\text{L2H}} = 13$ for different P_{success}, respectively.



(a) $F = 11$, $t = 9$ (b) $F = 242$, $t = 49$

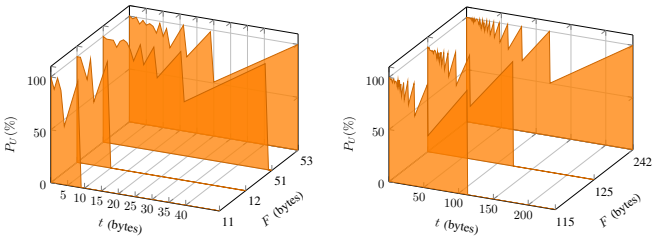
(c) $F = 11$, $t = 9$ (d) $F = 242$, $t = 49$

Fig. 14. Impact of the window size $\boldsymbol{w}$ (in tiles) or $\boldsymbol{8wt}$ (in bits) on the ACK overhead $\boldsymbol{O_{ACK_{L2}}}$. The window size is a multiple of the tile size using $\boldsymbol{w}$. $\boldsymbol{L_{SH} = 2}$ and $\boldsymbol{L_{L2H} = 13}$.

D. 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. 15a illustrates P_U for $F = 11, 12, 51, 53$ and different tiles sizes. When fragment 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. 15b shows the percentage of used bits for $F = 115, 125, 242$. As with smaller SCHC Fragments, some tile sizes have a better P_U than others. Moreover, as the tile size



(a) $F = 11, 12, 51, 53$ (b) $F = 115, 125, 242$

Fig. 15. Percentage of usage P_U vs t. $L_{SH} = 2$.

gets larger and only one tile can be fitted in the SCHC Fragment payload, the percentage of used 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%.

VII. Technology-Oriented Evaluation

This section provides configuration guidance for the main parameters of SCHC F/R ACK-on Error mode when used over LoRaWAN (EU868 and US915) and Sigfox, with respect to the ACK bit overhead O_{ACK} and $O_{\text{ACK}_{\text{L2}}}$ metrics. Furthermore, we present the maximum window and bitmap size for each fragment and tile size.

Considering that the ACK size has to be a byte multiple, we obtained the optimal window size value as the next integer value that will not require padding. For the SCHC ACK Header size we consider $L_{\text{SH}} = 2$. This makes 3 bytes the smallest negative ACK size possible without padding, and with a bitmap of 8 bits. The values presented in sections VII-A and VII-B are for $P_{\text{success}} \geq 0.85$. The FCN field size ($N$) is set to handle the number of tiles in the optimal or maximum window size. The length of the Window field (M) is calculated according to the number of windows required to transmit a 320-byte and 1280-byte SCHC Packet.

In SCHC ACK-on-Error mode, several F/R parameters, such as window size, tile size, N and M must be selected when starting a transmission of a SCHC Packet. However, instead of using a fixed configuration, a device may select the most suitable ACK-on-Error mode configuration values by estimating the BER, therefore P_{success}, and considering the available SCHC Fragment payload size.

A. ACK Bit Overhead

Table I presents the optimal window sizes for the ACK bit overhead O_{ACK} and SCHC F/R ACK-on-Error configuration parameters for LoRaWAN fragment sizes (F) in EU868 and US915 bands. Table II presents the optimal window size and SCHC F/R ACK-on-Error mode configuration parameters for the fragment size of Sigfox (only the uplink is considered).

B. ACK Bit Overhead with L2 Headers

Tables III and IV present the optimal window sizes from the perspective of the ACK bit overhead with L2 headers $O_{\text{ACK}_{\text{L2}}}$ and SCHC F/R ACK-on-Error configuration parameters, for LoRaWAN in EU868 and US915 bands, and for Sigfox, respectively.

TABLE I
Optimal Window Sizes (O_{ACK}) for LoRaWAN

Region	F (bytes)	SF	DR	t (bytes)	P_{success}	Optimal Window Size (tiles)	N (bits)	M (bits) SCHC Packet Size 320 (bytes)	M (bits) SCHC Packet Size 1280 (bytes)
Europe EU868	51	12, 11,10	DR0, DR1,DR2	49	$0.85 \leq P_{\text{success}} < 0.90$	40	6	1	1
					$0.90 \leq P_{\text{success}} \leq 1$	8	4	1	3
	115	9	DR3	49	$0.85 \leq P_{\text{success}} \leq 1$	8	4	1	3
				113	$0.85 \leq P_{\text{success}} < 0.90$	40	6	1	1
					$0.90 \leq P_{\text{success}} \leq 1$	8	4	1	2
	242	8, 7	DR4, DR5	49	$0.85 \leq P_{\text{success}} \leq 1$	8	4	1	3
				113	$0.85 \leq P_{\text{success}} \leq 1$	8	4	1	2
				240	$0.85 \leq P_{\text{success}} \leq 0.90$	24	5	1	1
					$0.90 < P_{\text{success}} \leq 1$	8	4	1	1
USA US915	11	10	DR0	9	$0.85 \leq P_{\text{success}} < 0.90$	48	6	1	3
					$0.90 \leq P_{\text{success}} \leq 1$	40	6	1	3
	53	9	DR1	9	$0.85 \leq P_{\text{success}} \leq 1$	8	4	3	5
				51	$0.85 \leq P_{\text{success}} \leq 0.90$	40	6	1	1
					$0.90 < P_{\text{success}} \leq 1$	8	4	1	3
	125	8	DR2	9	$0.85 \leq P_{\text{success}} \leq 1$	16	5	2	4
				51	$0.85 \leq P_{\text{success}} \leq 1$	8	4	1	3
				123	$0.85 \leq P_{\text{success}} \leq 0.90$	40	6	1	1
					$0.90 < P_{\text{success}} \leq 1$	16	5	1	1
	242	7, 8	DR3, DR4	9	$0.85 \leq P_{\text{success}} \leq 1$	32	6	2	3
				51	$0.85 \leq P_{\text{success}} \leq 1$	8	4	1	3
				123	$0.85 \leq P_{\text{success}} < 0.90$	40	6	1	1
					$0.90 \leq P_{\text{success}} \leq 1$	8	4	1	2
				240	$0.85 \leq P_{\text{success}} \leq 0.90$	24	5	1	1
					$0.90 < P_{\text{success}} \leq 1$	8	4	1	1

$L_{\text{SH}} = 2$. LoRaWAN physical layer options are defined and identified by the Spreading Factor (SF) and Data Rate (DR) parameters.

TABLE II
Optimal Window Sizes (O_{ACK}) for Sigfox

F (bytes)	t (bytes)	P_{success}	Optimal Window Size (tiles)	N (bits)	M (bits) SCHC Packet Size 320 (bytes)	M (bits) SCHC Packet Size 1280 (bytes)
12	10	$0.85 \leq P_{\text{success}} < 0.90$	40	6	1	3
		$0.90 \leq P_{\text{success}} \leq 1$	8	4	3	5

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

C. Maximum window and bitmap sizes

Table V shows the maximum bitmap and window sizes for LoRaWAN and Sigfox with $L_{\text{SH}} = 2$. Different tile sizes are presented considering different SCHC F/R ACK-on-Error parameters configurations. Recall that the optimal tile size is the size that entirely fills a SCHC Fragment. The maximum bitmap size does not depend on the tile size, but on the SCHC Fragment and SCHC Header sizes as shown in (11). Note that, Sigfox has a downlink payload size of 8 bytes [6], hence with $L_{\text{SH}} = 2$ at most 6 bytes can be used for the bitmap.

VIII. Conclusions and Perspectives

The SCHC framework enables LPWAN technologies to comply with IPv6 by performing a static context header compression and fragmentation. ACK-Always and ACK-on-Error modes, provide reliable F/R between the sender and the receiver. In this article, we have 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 yielded to transmission parameters minimizing the ACK burden imposed on the (more sensitive to duty-cycle constraints) downlink. Finally we have 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.

TABLE III
OPTIMAL WINDOW SIZES WITH L2 HEADERS ($O_{\mathbf{ACK_{L2}}}$) FOR LORAWAN

Region	F (bytes)	SF	DR	t (bytes)	P_{success}	Optimal Window Size (tiles)	N (bits)	M (bits) SCHC 320 (bytes)	M (bits) Packet Size 1280 (bytes)
Europe EU868	51	12, 11,10	DR0, DR1,DR2	49	$0.85 \leq P_{\text{success}} \leq 0.98$	392	9	1	1
					$0.98 < P_{\text{success}} \leq 1$	8	4	1	3
	115	9	DR3	49	$0.85 \leq P_{\text{success}} < 0.90$	512	10	1	1
					$0.90 \leq P_{\text{success}} < 0.95$	480	9	1	1
					$0.95 \leq P_{\text{success}} < 0.97$	400	9	1	1
					$0.97 \leq P_{\text{success}} < 0.98$	584	10	1	1
					$0.98 \leq P_{\text{success}} \leq 1$	8	4	1	2
				113	$0.85 \leq P_{\text{success}} < 0.90$	512	10	1	1
					$0.90 \leq P_{\text{success}} < 0.95$	480	9	1	1
					$0.95 \leq P_{\text{success}} < 0.97$	400	9	1	1
					$0.97 \leq P_{\text{success}} < 0.98$	584	10	1	1
					$0.98 \leq P_{\text{success}} \leq 1$	8	4	1	2
	242	8, 7	DR4, DR5	49	$0.85 \leq P_{\text{success}} \leq 1$	8	4	1	2
				113	$0.85 \leq P_{\text{success}} < 0.90$	512	10	1	1
					$0.90 \leq P_{\text{success}} < 0.95$	480	9	1	1
					$0.95 \leq P_{\text{success}} < 0.97$	400	9	1	1
					$0.97 \leq P_{\text{success}} < 0.98$	584	10	1	1
					$0.98 \leq P_{\text{success}} \leq 1$	8	4	1	2
				240	$0.85 \leq P_{\text{success}} < 0.90$	576	10	1	1
					$0.90 \leq P_{\text{success}} < 0.95$	624	10	1	1
					$0.95 \leq P_{\text{success}} < 0.99$	368	9	1	1
					$0.99 \leq P_{\text{success}} \leq 1$	8	4	1	1
US US915	11	10	DR0	9	$0.85 \leq P_{\text{success}} < 0.98$	72	7	1	2
					$0.98 \leq P_{\text{success}} \leq 1$	8	4	3	5
	53	9	DR1	9	$0.85 \leq P_{\text{success}} \leq 0.90$	328	9	1	1
					$0.90 < P_{\text{success}} \leq 1$	8	4	3	5
				51	$0.85 \leq P_{\text{success}} \leq 0.98$	408	9	1	1
					$0.98 < P_{\text{success}} \leq 1$	8	4	1	3
	125	8	DR2	9	$0.85 \leq P_{\text{success}} \leq 1$	16	5	2	4
				51	$0.85 \leq P_{\text{success}} < 0.98$	72	7	1	1
					$0.98 \leq P_{\text{success}} \leq 1$	8	4	1	3
				123	$0.85 \leq P_{\text{success}} < 0.90$	576	10	1	1
					$0.9 \leq P_{\text{success}} < 0.95$	624	10	1	1
					$0.95 \leq P_{\text{success}} < 0.97$	360	9	1	1
					$0.97 \leq P_{\text{success}} < 0.98$	456	9	1	1
					$0.98 \leq P_{\text{success}} < 0.99$	624	10	1	1
					$0.99 \leq P_{\text{success}} \leq 1$	8	4	1	2
	242	7, 8	DR3, DR4	9	$0.85 \leq P_{\text{success}} \leq 1$	32	6	2	3
				51	$0.85 \leq P_{\text{success}} < 0.90$	376	9	1	1
					$0.90 \leq P_{\text{success}} < 0.95$	328	9	1	1
					$0.95 \leq P_{\text{success}} < 0.96$	488	9	1	1
					$0.96 \leq P_{\text{success}} \leq 1$	8	4	1	3
				123	$0.85 \leq P_{\text{success}} < 0.90$	576	10	1	1
					$0.90 \leq P_{\text{success}} < 0.95$	624	10	1	1
					$0.95 \leq P_{\text{success}} < 0.98$	360	9	1	1
					$0.98 \leq P_{\text{success}} < 0.99$	624	10	1	1
					$0.99 \leq P_{\text{success}} \leq 1$	8	4	1	2
				240	$0.85 \leq P_{\text{success}} < 0.90$	576	10	1	1
					$0.90 \leq P_{\text{success}} < 0.95$	624	10	1	1
					$0.95 \leq P_{\text{success}} < 0.99$	368	9	1	1
					$0.99 \leq P_{\text{success}} \leq 1$	8	4	1	1

$L_{\text{SH}} = 2$. LoRaWAN physical layer options are defined and identified by the Spreading Factor (SF) and Data Rate (DR) parameters.

TABLE IV
OPTIMAL WINDOW SIZES WITH L2 HEADERS ($O_{\mathbf{ACK_{L2}}}$) FOR SIGFOX

F (bytes)	t (bytes)	P_{success}	Optimal Window Size (tiles)	N (bits)	M (bits) SCHC 320 (bytes)	M (bits) Packet Size 1280 (bytes)
12	10	$0.85 \leq P_{\text{success}} \leq 0.95$	48	6	1	3
		$0.95 < P_{\text{success}} \leq$	8	4	3	5

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

TABLE V
MAXIMUM BITMAP AND WINDOW SIZE FOR LORAWAN AND SIGFOX

Region & Technology	F (bytes)	t (bytes)	Maximum bitmap size (bits)	Maximum window size (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

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

ACKNOWLEDGMENT

This research was done during Sergio Aguilar visit at IMT-Atlantique and is supported in part by the Spanish Government through project TEC2016-79988-P, AEI/FEDER, EU.

REFERENCES

[1] S. Farrell, "Low-Power Wide Area Network (LPWAN) Overview," RFC 8376, RFC Editor, May 2018.

[2] 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.

[3] 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.

[4] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and J. Zuniga, "LPWAN Static Context Header Compression (SCHC) and fragmentation for IPv6 and UDP," Internet-Draft draft-ietf-lpwan-ipv6-static-context-hc-18.txt, IETF Secretariat, Dec. 2018.

[5] 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 (in press)*.

[6] C. Gomez, J. C. Veras, R. Vidal, L. Casals, and J. Paradells, "A Sigfox Energy Consumption Model," *Sensors*, vol. 19, no. 3, 2019.

[7] 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, pp. 34–40, Sept 2017.

[8] LoRa Alliance Technical Committee, "LoRaWAN 1.1 Specification," https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf, Oct 2017.

[9] Sigfox, "Sigfox connected objects: Radio specifications," https://build.sigfox.com/sigfox-device-radio-specifications, 2 2019. Rev. 1.3.

[10] L. Casals, B. Mir, R. Vidal, and C. Gomez, "Modeling the energy performance of LoRaWAN," *Sensors*, vol. 17, no. 10, 2017.

[11] 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. 35816–35825, 2019.

[12] 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.

[13] N. I. Osman and E. B. Abbas, "Simulation and Modelling of LoRa and Sigfox Low Power Wide Area Network Technologies," in *Proc. of ICCCEEE*, (Guayaquil, Ecuador), Aug 2018.

[14] 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.

[15] I. Suciu, X. Vilajosana, and F. Adelantado, "An analysis of packet fragmentation impact in LPWAN," in *Proc. of IEEE WCNC*, (Barcelona, Spain), 2018.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.), pp. 213–227, Springer International Publishing (Cham), 2019.