lpwan Working Group   JC. Zuniga
Internet-Draft   SIGFOX
Intended status: Standards Track   C. Gomez
Expires: December 13, 2021   S. Aguilar
                              Universitat Politecnica de Catalunya
                                                           L. Toutain
                                                        IMT-Atlantique
                                                          S. Cespedes
                                                           D. Wistuba
                                     NIC Labs, Universidad de Chile
                                                        June 11, 2021

SCHC over Sigfox LPWAN
draft-ietf-lpwan-schc-over-sigfox-06

Abstract

   The Generic Framework for Static Context Header Compression and
   Fragmentation (SCHC) specification describes two mechanisms: i) an
   application header compression scheme, and ii) a frame fragmentation
   and loss recovery functionality. SCHC offers a great level of
   flexibility that can be tailored for different Low Power Wide Area
   Network (LPWAN) technologies.

   The present document provides the optimal parameters and modes of
   operation when SCHC is implemented over a Sigfox LPWAN. This set of
   parameters are also known as a "SCHC over Sigfox profile."

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on December 13, 2021.

Copyright Notice

Table of Contents

[1](#). Introduction

The Generic Framework for Static Context Header Compression and
Fragmentation (SCHC) specification [[RFC8724](#)] describes two
mechanisms: i) an application header compression scheme, and ii) a
frame fragmentation and loss recovery functionality. Either can be
used on top of all the four LWPAN technologies defined in [[RFC8376](#)].
These LPWANs have similar characteristics such as star-oriented
topologies, network architecture, connected devices with built-in
applications, etc.

SCHC offers a great level of flexibility to accommodate all these
LPWAN technologies. Even though there are a great number of
similarities between them, some differences exist with respect to the
transmission characteristics, payload sizes, etc. Hence, there are
optimal parameters and modes of operation that can be used when SCHC
is used on top of a specific LPWAN technology.

This document describes the recommended parameters, settings, and
modes of operation to be used when SCHC is implemented over a Sigfox
LPWAN. This set of parameters are also known as a "SCHC over Sigfox
profile."

[2](#). Terminology

It is assumed that the reader is familiar with the terms and
mechanisms defined in [[RFC8376](#)] and in [[RFC8724](#)].

[3](#). SCHC: Generic Framework for Static Context Header Compression and Fragmentation

The Generic Framework for Static Context Header Compression and
Fragmentation (SCHC) described in [[RFC8724](#)] takes advantage of the
predictability of data flows existing in LPWAN applications to avoid
context synchronization.

Contexts must be stored and pre-configured on both ends. This can be
done either by using a provisioning protocol, by out of band means,
or by pre-provisioning them (e.g. at manufacturing time). The way
contexts are configured and stored on both ends is out of the scope
of this document.

[4](#). SCHC over Sigfox

4.1. Network Architecture

   Figure 1 represents the architecture for compression/decompression
   (C/D) and fragmentation/reassembly (F/R) based on the terminology
   defined in [RFC8376], where the Radio Gateway (RG) is a Sigfox Base
   Station and the Network Gateway (NGW) is the Sigfox cloud-based
   Network.

```
   Sigfox Device                                        Application
+----------------+                          +--------------+
| APP1 APP2 APP3 |                          |APP1 APP2 APP3|
+----------------+                          +--------------+
|   UDP  |       |                          |    |  UDP   |
|   IPv6 |       |                          |    | IPv6   |
+--------+       |                          |    +--------+
| SCHC C/D & F/R |                          |             |
|                |                          |             |
+-------+--------+                          +--------+-----+
        $                                          .
        $   +---------+     +--------------+     +---------+    .
        $   |         |     |              |     | Network |    .
      +~~ |Sigfox BS|     |Sigfox Network|     |  SCHC   |    .
          | (RG)    | === |    (NGW)      | === |C/D & F/R|.....
          +---------+     +--------------+     +---------+
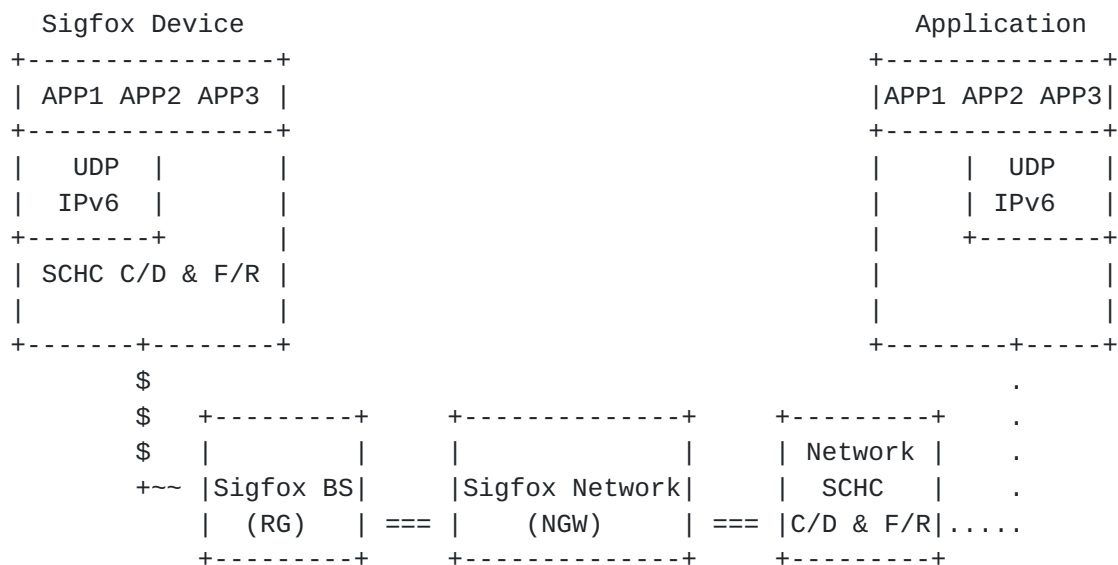```

                     Figure 1: Network Architecture

   In the case of the global Sigfox Network, RGs (or Base Stations) are
   distributed over multiple countries wherever the Sigfox LPWAN service
   is provided.  The NGW (or cloud-based Sigfox Core Network) is a
   single entity that connects to all Sigfox base stations in the world,
   providing hence a global single star network topology.

   The Device sends application flows that are compressed and/or
   fragmented by a SCHC Compressor/Decompressor (SCHC C/D + F/R) to
   reduce headers size and/or fragment the packet.  The resulting SCHC
   Message is sent over a layer two (L2) Sigfox frame to the Sigfox Base
   Stations, which then forward the SCHC Message to the Network Gateway
   (NGW).  The NGW then delivers the SCHC Message and associated
   gathered metadata to the Network SCHC C/D + F/R.

   The Sigfox Network (NGW) communicates with the Network SCHC C/D + F/R
   for compression/decompression and/or for fragmentation/reassembly.
   The Network SCHC C/D + F/R shares the same set of rules as the Dev
   SCHC C/D + F/R.  The Network SCHC C/D + F/R can be collocated with

the NGW or it could be located in a different place, as long as a
tunnel or secured communication is established between the NGW and
the SCHC C/D + F/R functions.  After decompression and/or reassembly,
the packet can be forwarded over the Internet to one (or several)
LPWAN Application Server(s) (App).

The SCHC C/D + F/R processes are bidirectional, so the same
principles are applicable on both uplink (UL) and downlink (DL).

## 4.2.  Uplink

Uplink Sigfox transmissions occur in repetitions over different times
and frequencies.  Besides time and frequency diversities, the Sigfox
network also provides space diversity, as potentially an uplink
message will be received by several base stations.

Since all messages are self-contained and base stations forward all
these messages back to the same Sigfox Network, multiple input copies
can be combined at the NGW providing for extra reliability based on
the triple diversity (i.e., time, space and frequency).

A detailed description of the Sigfox Radio Protocol can be found in
[sigfox-spec].

Messages sent from the Device to the Network are delivered by the
Sigfox network (NGW) to the Network SCHC C/D + F/R through a
callback/API with the following information:

o  Device ID

o  Message Sequence Number

o  Message Payload

o  Message Timestamp

o  Device Geolocation (optional)

o  RSSI (optional)

o  Device Temperature (optional)

o  Device Battery Voltage (optional)

The Device ID is a globally unique identifier assigned to the Device,
which is included in the Sigfox header of every message.  The Message
Sequence Number is a monotonically increasing number identifying the
specific transmission of this uplink message, and it is also part of

the Sigfox header.  The Message Payload corresponds to the payload
that the Device has sent in the uplink transmission.

The Message Timestamp, Device Geolocation, RSSI, Device Temperature
and Device Battery Voltage are metadata parameters provided by the
Network.

A detailed description of the Sigfox callbacks/APIs can be found in
[sigfox-callbacks].

Only messages that have passed the L2 Cyclic Redundancy Check (CRC)
at network reception are delivered by the Sigfox Network to the
Network SCHC C/D + F/R.

```
         +---------------+-----------------+
         | Sigfox Header | Sigfox payload |
         +---------------+---------------- +
                         | SCHC message |
                         +-----------------+
```
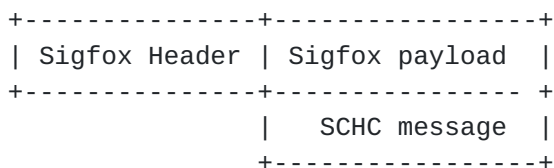
Figure 2: SCHC Message in Sigfox

Figure 2 shows a SCHC Message sent over Sigfox, where the SCHC
Message could be a full SCHC Packet (e.g. compressed) or a SCHC
Fragment (e.g. a piece of a bigger SCHC Packet).

## 4.3.  Downlink

Downlink transmissions are Device-driven and can only take place
following an uplink communication that so indicates.  Hence, a Device
explicitly indicates its intention to receive a downlink message
using a donwlink request flag when sending the preceding uplink
message to the network.  After completing the uplink transmission,
the Device opens a fixed window for downlink reception.  The delay
and duration of the reception opportunity window have fixed values.
If there is a downlink message to be sent for this given Device (e.g.
either a response to the uplink message or queued information waiting
to be transmitted), the network transmits this message to the Device
during the reception window.  If no message is received by the Device
after the reception opportunity window has elapsed, the Device closes
the reception window opportunity and gets back to the normal mode
(e.g., continue UL transmissions, sleep, stand-by, etc.)

When a downlink message is sent to a Device, a reception
acknowledgement is generated by the Device and sent back to the

Network through the Sigfox radio protocol and reported in the Sigfox
Network backend.

A detailed description of the Sigfox Radio Protocol can be found in
[sigfox-spec] and a detailed description of the Sigfox callbacks/APIs
can be found in [sigfox-callbacks].

## 4.4.  SCHC-ACK on Downlink

As explained previously, downlink transmissions are Device-driven and
can only take place following a specific uplink transmission that
indicates and allows a following downlink opportunity.  For this
reason, when SCHC bi-directional services are used (e.g.  Ack-on-
Error fragmentation mode) the SCHC protocol implementation needs to
consider the times when a downlink message (e.g.  SCHC-ACK) can be
sent and/or received.

For the UL ACK-on-Error fragmentation mode, a DL opportunity MUST be
indicated by the last fragment of every window (i.e.  FCN = All-0, or
FCN = All-1).  The Device sends the fragments in sequence and, after
transmitting the FCN = All-0 or FCN = All-1, it opens up a reception
opportunity.  The Network SCHC can then decide to respond at that
opportunity (or wait for a further one) with a SCHC-ACK indicating in
case there are missing fragments from the current or previous
windows.  If there is no SCHC-ACK to be sent, or if the network
decides to wait for a further DL transmission opportunity, then no DL
transmission takes place at that opportunity and after a timeout the
UL transmissions continue.  Intermediate SCHC fragments with FCN
different from All-0 or All-1 MUST NOT use the DL request flag to
request a SCHC-ACK.

## 4.5.  SCHC Rules

The RuleID MUST be included in the SCHC header.  The total number of
rules to be used affects directly the Rule ID field size, and
therefore the total size of the fragmentation header.  For this
reason, it is recommended to keep the number of rules that are
defined for a specific device to the minimum possible.

RuleIDs can be used to differentiate data traffic classes (e.g.  QoS,
control vs. data, etc.), and data sessions.  They can also be used to
interleave simultaneous fragmentation sessions between a Device and
the Network.

## 4.6.  Fragmentation

The SCHC specification [RFC8724] defines a generic fragmentation
functionality that allows sending data packets or files larger than
the maximum size of a Sigfox payload.  The functionality also defines
a mechanism to send reliably multiple messages, by allowing to resend
selectively any lost fragments.

The SCHC fragmentation supports several modes of operation.  These
modes have different advantages and disadvantages depending on the
specifics of the underlying LPWAN technology and application Use
Case.  This section describes how the SCHC fragmentation
functionality should optimally be implemented when used over a Sigfox
LPWAN for the most typical Use Case applications.

As described in section 8.2.3 of [RFC8724], the integrity of the
fragmentation-reassembly process of a SCHC Packet MUST be checked at
the receive end.  Since only UL messages/fragments that have passed
the CRC-check are delivered to the Network SCHC C/D + F/R, and each
one has an associated Sigfox Message Sequence Number (see
Section 4.2), integrity can be guaranteed when no consecutive
messages are missing from the sequence and all FCN bitmaps are
complete.  In order to support multiple flows/RuleIDs (potentially
interleaved), the implementation of a central message sequence
counter at the Network SCHC C/D + F/R is required.  With this
functionality and in order to save protocol overhead, the use of a
dedicated Reassembly Check Sequence (RCS) is NOT RECOMMENDED.

The L2 Word Size used by Sigfox is 1 byte (8 bits).

## 4.6.1.  SCHC Compound ACK

The present SCHC over Sigfox specification extends SCHC ACK format
defined in [RFC8724] with the SCHC Compound ACK concept.

The SCHC Compound ACK is a SCHC ACK message that can contain several
bitmaps, each bitmap being identified by its corresponding window
number.  The SCHC Compound ACK concept is meant to reduce the number
of downlink transmissions (i.e., SCHC ACKs) by including bitmaps of
several windows in a single SCHC message (i.e., the SCHC Compound
ACK), and hence making an efficient use of downlink channel
transmissions.

When the ACK-on-Error mode is used for uplink fragmentation, SCHC
Compound ACKs MUST be used in the downlink responses.

The SCHC Compound ACK:

o  provides feedback only for windows with fragment losses,

o  has a variable size that depends on the number of windows with
   fragment losses being reported in the single Compound SCHC ACK,

o  includes the window number (i.e., W) for each bitmap,

o  has a format coincident with that of a SCHC ACK (RFC 8724) when
   only one window with losses is reported,

o  might not cover all windows with fragment losses of a SCHC Packet,

o  is distinguishable from the SCHC Receiver-Abort.

The SCHC Compound ACK groups the window number (W) with its
corresponding bitmap.  The included window numbers and corresponding
bitmap MUST be ordered from the lowest-numbered to the highest-
numbered window.

## 4.6.2.  Uplink Fragmentation

Sigfox uplink transmissions are completely asynchronous and take
place in any random frequency of the allowed uplink bandwidth
allocation.  In addition, devices may go to deep sleep mode, and then
wake up and transmit whenever there is a need to send information to
the network.  Data packets are self-contained (aka "message in a
bottle") with all the required information for the network to process
them accordingly.  Hence, there is no need to perform any network
attachment, synchronization, or other procedure before transmitting a
data packet.

Since uplink transmissions are asynchronous, a SCHC fragment can be
transmitted at any given time by the Device.  Sigfox uplink messages
are fixed in size, and as described in [RFC8376] they can carry 0-12
bytes payload.  Hence, a single SCHC Tile size per fragmentation mode
can be defined so that every Sigfox message always carries one SCHC
Tile.

### 4.6.2.1.  Uplink No-ACK Mode

No-ACK is RECOMMENDED to be used for transmitting short, non-critical
packets that require fragmentation and do not require full
reliability.  This mode can be used by uplink-only devices that do
not support downlink communications, or by bidirectional devices when
they send non-critical data.

Since there are no multiple windows in the No-ACK mode, the W bit is
not present.  However it is RECOMMENDED to use the FCN field to

indicate the size of the data packet.  In this sense, the data packet
would need to be splitted into X fragments and, similarly to the
other fragmentation modes, the first transmitted fragment would need
to be marked with FCN = X-1.  Consecutive fragments MUST be marked
with decreasing FCN values, having the last fragment marked with FCN
= (All-1).  Hence, even though the No-ACK mode does not allow
recovering missing fragments, it allows indicating implicitly the
size of the expected packet to the Network and hence detect at the
receiver side whether all fragments have been received or not.

The RECOMMENDED Fragmentation Header size is 8 bits, and it is
composed as follows:

o  RuleID size: 4 bits

o  DTag size (T): 0 bits

o  Fragment Compressed Number (FCN) size (N): 4 bits

o  As per [RFC8724], in the No-ACK mode the W (window) field is not
   present.

o  RCS size: 0 bits (Not used)

**4.6.2.2.  Uplink ACK-on-Error Mode: Single-byte SCHC Header**

ACK-on-Error with single-byte header is RECOMMENDED for medium to
large size packets that need to be sent reliably.  ACK-on-Error is
optimal for Sigfox transmissions, since it leads to a reduced number
of ACKs in the lower capacity downlink channel.  Also, downlink
messages can be sent asynchronously and opportunistically.

Allowing transmission of packets/files up to 300 bytes long, the SCHC
uplink Fragmentation Header size is RECOMMENDED to be 8 bits in size
and is composed as follows:

o  Rule ID size: 3 bits

o  DTag size (T): 0 bits

o  Window index (W) size (M): 2 bits

o  Fragment Compressed Number (FCN) size (N): 3 bits

o  MAX_ACK_REQUESTS: 5

o  WINDOW_SIZE: 7 (with a maximum value of FCN=0b110)

   o  Tile size: 11 bytes

   o  Retransmission Timer: Application-dependent

   o  Inactivity Timer: Application-dependent

   o  RCS size: 0 bits (Not used)

   The correspondent SCHC ACK in the downlink is 13 bits long, so
   padding is needed to complete the required 64 bits of Sigfox payload.

## 4.6.2.3.  Uplink ACK-on-Error Mode: Two-byte SCHC Header

   ACK-on-Error with two-byte header is RECOMMENDED for very large size
   packets that need to be sent reliably.  ACK-on-Error is optimal for
   Sigfox transmissions, since it leads to a reduced number of ACKs in
   the lower capacity downlink channel.  Also, downlink messages can be
   sent asynchronously and opportunistically.

   In order to allow transmission of very large packets/files up to 2250
   bytes long, the SCHC uplink Fragmentation Header size is RECOMMENDED
   to be 16 bits in size and composed as follows:

   o  Rule ID size is: 8 bits

   o  DTag size (T) is: 0 bits

   o  Window index (W) size (M): 3 bits

   o  Fragment Compressed Number (FCN) size (N): 5 bits.

   o  MAX_ACK_REQUESTS: 5

   o  WINDOW_SIZE: 31 (with a maximum value of FCN=0b11110)

   o  Tile size: 10 bytes

   o  Retransmission Timer: Application-dependent

   o  Inactivity Timer: Application-dependent

   o  RCS size: 0 bits (Not used)

   The correspondent SCHC ACK in the downlink is 43 bits long, so
   padding is needed to complete the required 64 bits of Sigfox payload.

4.6.2.4.  **All-1 behaviour + Sigfox Sequence Number**

   For ACK-on-Error, as defined in [RFC8724], it is expected that the
   last SCHC fragment of the last window will always be delivered with
   an All-1 FCN.  Since this last window may not be full (i.e. it may be
   comprised of less than WINDOW_SIZE fragments), an All-1 fragment may
   follow a value of FCN higher than 1 (0b01).  In this case, the
   receiver could not derive from the FCN values alone whether there are
   any missing fragments right before the All-1 fragment or not.

   However, since a Message Sequence Number is provided by the Sigfox
   protocol together with the Sigfox Payload, the receiver can detect if
   there are missing fragments before the All-1 and hence construct the
   corresponding SCHC ACK Bitmap accordingly.

4.6.3.  **Downlink Fragmentation**

   In some LPWAN technologies, as part of energy-saving techniques,
   downlink transmission is only possible immediately after an uplink
   transmission.  This allows the device to go in a very deep sleep mode
   and preserve battery, without the need to listen to any information
   from the network.  This is the case for Sigfox-enabled devices, which
   can only listen to downlink communications after performing an uplink
   transmission and requesting a downlink.

   When there are fragments to be transmitted in the downlink, an uplink
   message is required to trigger the downlink communication.  In order
   to avoid potentially high delay for fragmented datagram transmission
   in the downlink, the fragment receiver MAY perform an uplink
   transmission as soon as possible after reception of a downlink
   fragment that is not the last one.  Such uplink transmission MAY be
   triggered by sending a SCHC message, such as a SCHC ACK.  However,
   other data messages can equally be used to trigger DL communications.

   Sigfox downlink messages are fixed in size, and as described in
   [RFC8376] they can carry up to 8 bytes payload.  Hence, a single SCHC
   Tile size per mode can be defined so that every Sigfox message always
   carries one SCHC Tile.

   For reliable downlink fragment transmission, the ACK-Always mode is
   RECOMMENDED.

   The SCHC downlink Fragmentation Header size is RECOMMENDED to be 8
   bits in size and is composed as follows:

   o  RuleID size: 3 bits

   o  DTag size (T): 0 bits

o  Window index (W) size (M) is: 0 bits

o  Fragment Compressed Number (FCN) size (N): 5 bits

o  MAX_ACK_REQUESTS: 5

o  WINDOW_SIZE: 31 (with a maximum value of FCN=0b11110)

o  Tile size: 7 bytes

o  Retransmission Timer: Application-dependent

o  Inactivity Timer: Application-dependent

o  RCS size: 0 bits (Not used)

## 4.7.  SCHC-over-Sigfox F/R Message Formats

This section depicts the different formats of SCHC Fragment, SCHC ACK
(including the SCHC Compound ACK defined in Section 4.6.1), and SCHC
Abort used in SCHC over Sigfox.

### 4.7.1.  Uplink ACK-on-Error Mode: Single-byte SCHC Header

#### 4.7.1.1.  Regular SCHC Fragment

Figure 3 shows an example of a regular SCHC fragment for all
fragments except the last one.  As tiles are of 11 bytes, padding
MUST NOT be added.

```
              |-- SCHC Fragment Header --|
              + ----------------------- + ------- +
              | RuleID |   W    | FCN    | Payload |
              + ------ + ------ + ------ + ------- +
              | 3 bits | 2 bits | 3 bits | 88 bits |
```

Figure 3: Regular SCHC Fragment format for all fragments except the
last one

The use of SCHC ACK REQ is NOT RECOMMENDED, instead the All-1 SCHC
Fragment SHOULD be used to request a SCHC ACK from the receiver
(Network SCHC). As per [RFC8724], the All-0 message is
distinguishable from the SCHC ACK REQ (All-1 message). The
penultimate tile of a SCHC Packet is of regular size.

4.7.1.2. All-1 SCHC Fragment

Figure 4 shows an example of the All-1 message. The All-1 message
MUST contain the last tile of the SCHC Packet. The last tile MUST be
of at least 1 byte (one L2 word). Padding MUST NOT be added, as the
resulting size is L2-word-multiple.

```
         |---  SCHC Fragment Header ---|
         + ------------------------- + ----------- +
         | RuleID |   W    | FCN=ALL-1 |   Payload   |
         + ------ + ------ + --------- + ----------- +
         | 3 bits | 2 bits |  3 bits   | 8 to 88 bits |
```

           Figure 4: All-1 SCHC Message format with last tile

As per [RFC8724] the All-1 must be distinguishable from a SCHC
Sender-Abort message (with same Rule ID, M, and N values). The All-1
MUST have the last tile of the SCHC Packet, which MUST be of at least
1 byte. The SCHC Sender-Abort message header size is of 1 byte, with
no padding bits.

For the All-1 message to be distinguishable from the Sender-Abort
message, the Sender-Abort message MUST be of 1 byte (only header with
no padding). This way, the minimum size of the All-1 is 2 bytes, and
the Sender-Abort message is 1 byte.

4.7.1.3. SCHC ACK Format

Figure 5 shows the SCHC ACK format when all fragments have been
correctly received (C=1). Padding MUST be added to complete the
64-bit Sigfox downlink frame payload size.

```
           |---- SCHC ACK Header ----|
           + --------------------- + ------- +
           | RuleID |   W    | C=b'1 | b'0-pad |
           + ------ + ------ + ----- + ------- +
           | 3 bits | 2 bits | 1 bit | 58 bits |
```

             Figure 5: SCHC Success ACK message format

In case SCHC fragment losses are found in any of the windows of the
SCHC Packet (C=0), the SCHC Compound ACK MUST be used. The SCHC
Compound ACK message format is shown in Figure 6. The window
numbered 00, if present in the SCHC Compound ACK, MUST be placed

between the Rule ID and the C bit to avoid confusion with padding
bits. If padding is needed for the SCHC Compound ACK, padding bits
MUST be 0 to make subsequent window numbers and bitmaps
distinguishable.

```
|---- SCHC ACK Header ----|-W = x -|...| --- W = x + i ---|
+ --------------------- + ------ +...+ ------- + ------ + ------- +
| RuleID | W=b'x  | C=b'0 | Bitmap |...| W=b'x+i | Bitmap | b'0-pad |
+ ------ + ------ + ----- + ------ +...+ ------- + ------ + ------- +
| 3 bits | 2 bits | 1 bit | 7 bits |   | 2 bits  | 7 bits |
```

   On top are noted the window number of the corresponding bitmap.
   Losses are found in windows x,...,x+i.

                  Figure 6: SCHC Compound ACK message format

The following figures show examples of the Compound ACK message
format.

```
|---- SCHC ACK Header ----|- W=00 -|----- W=01 ------|
+ --------------------- + ------ + ------ + ------ + ------- +
| RuleID | W=b'00 | C=b'0 | Bitmap | W=b'01 | Bitmap | b'0-pad |
+ ------ + ------ + ----- + ------ + ------ + ------ + ------- +
| 3 bits | 2 bits | 1 bit | 7 bits | 2 bits | 7 bits | 42 bits |
```

        Losses are found in windows 00 and 01.

                     Figure 7: SCHC Compound ACK example 1

```
|---- SCHC ACK Header ----|- W=01 -|----- W=11 ------|
+ --------------------- + ------ + ------ + ------ + ------- +
| RuleID | W=b'01 | C=b'0 | Bitmap | W=b'11 | Bitmap | b'0-pad |
+ ------ + ------ + ----- + ------ + ------ + ------ + ------- +
| 3 bits | 2 bits | 1 bit | 7 bits | 2 bits | 7 bits | 42 bits |
```

        Losses are found in windows 01 and 11.

                     Figure 8: SCHC Compound ACK example 2

```
|---- SCHC ACK Header ----|- W=00 -|----- W=10 ------|
+ ---------------------- + ------ + ------ + ------ + ------- +
| RuleID | W=b'00 | C=b'0 | Bitmap | W=b'10 | Bitmap | b'0-pad |
+ ------ + ------ + ----- + ------ + ------ + ------ + ------- +
| 3 bits | 2 bits | 1 bit | 7 bits | 2 bits | 7 bits | 42 bits |

       Losses are found in windows 00 and 10.
```

                Figure 9: SCHC Compound ACK example 3

Figure 10 shows the Compound ACK message format when losses are found
in all windows. The window numbers and its corresponding bitmaps are
ordered from window numbered 00 to 11, notifying all four possible
windows.

```
|- SCHC ACK Header -|W=b'00|-- W=b'01 ---|
+-------------------+------+ ---- +------+
|RuleID|W=b'00|C=b'0|Bitmap|W=b'01|Bitmap|  ...
+------+------+-----+------+------+------+
|3 bits|2 bits|1 bit|7 bits|2 bits|7 bits|


        |--- W=b'10 --|--- W=b'11 --|
        |------+------+------+------+-------+
   ...  |W=b'10|Bitmap|W=b'11|Bitmap|b'0-pad|
        |------+------+------+------+-------+
        |2 bits|7 bits|2 bits|7 bits|24 bits|

    Losses are found in windows 00, 01, 10 and 11.
```

                Figure 10: SCHC Compound ACK example 4

```
|- SCHC ACK Header -|W=b'00|-- W=b'01 ---|--- W=b'10 --|
+-------------------+------+------+------+------+------+-------+
|RuleID|W=b'00|C=b'0|Bitmap|W=b'01|Bitmap|W=b'10|Bitmap|b'0-pad|
+------+------+-----+------+------+------+------+------+-------+
|3 bits|2 bits|1 bit|7 bits|2 bits|7 bits|2 bits|7 bits|33 bits|

       Losses are found in windows 00, 01 and 10.
```

                Figure 11: SCHC Compound ACK example 5

4.7.1.4. **SCHC Sender-Abort Message format**

```
                  |---- Sender-Abort Header ----|
                  + --------------------------- +
                  | RuleID |   W    | FCN=ALL-1  |
                  + ------ + ------ + --------- +
                  | 3 bits | 2 bits |  3 bits   |
```

           Figure 12: SCHC Sender-Abort message format

4.7.1.5. **SCHC Receiver-Abort Message format**

```
                  |- Receiver-Abort Header -|
                  + --------------------- + ------- +
                  | RuleID | W=b'11 | C=b'1 | b'1-pad |
                  + ------ + ------ + ----- + ------- +
                  | 3 bits | 2 bits | 1 bit | 58 bits |
```

           Figure 13: SCHC Receiver-Abort message format

4.7.2. **Uplink ACK-on-Error Mode: Two-byte SCHC Header**

4.7.2.1. **Regular SCHC Fragment**

   Figure 14 shows an example of a regular SCHC fragment for all
   fragments except the last one. The penultimate tile of a SCHC Packet
   is of the regular size.

```
                  |-- SCHC Fragment Header --|
                  + --------------------- + ------- +
                  | RuleID |   W    | FCN    | Payload |
                  + ------ + ------ + ------ + ------- +
                  | 8 bits | 3 bits | 5 bits | 80 bits |
```

      Figure 14: Regular SCHC Fragment format for all fragments except the
                            last one

   The use of SCHC ACK is NOT RECOMMENDED, instead the All-1 SCHC
   Fragment SHOULD be used to request a SCHC ACK from the receiver
   (Network SCHC). As per [RFC8724], the All-0 message is
   distinguishable from the SCHC ACK REQ (All-1 message).

4.7.2.2. **All-1 SCHC Fragment**

Figure 15 shows an example of the All-1 message. The All-1 message
MUST contain the last tile of the SCHC Packet.

```
              |---  SCHC Fragment Header ---|
              + ------------------------- + ----------- +
              | RuleID |   W   | FCN=ALL-1 |   Payload   |
              + ------ + ------ + -------- + ----------- +
              | 8 bits | 3 bits |  5 bits   | 8 to 80 bits |
```

Figure 15: All-1 SCHC message format with last tile

As per [RFC8724] the All-1 must be distinguishable from the a SCHC
Sender-Abort message (with same Rule ID, M and N values). The All-1
MUST have the last tile of the SCHC Packet, that MUST be of at least
1 byte. The SCHC Sender-Abort message header size is of 2 byte, with
no padding bits.

For the All-1 message to be distinguishable from the Sender-Abort
message, the Sender-Abort message MUST be of 2 byte (only header with
no padding). This way, the minimum size of the All-1 is 3 bytes, and
the Sender-Abort message is 2 bytes.

4.7.2.3. **SCHC ACK Format**

Figure 16 shows the SCHC ACK format when all fragments have been
correctly received (C=1). Padding MUST be added to complete the
64-bit Sigfox downlink frame payload size.

```
              |----- SCHC ACK Header ----|
              + ---------------------- + ------ +
              | RuleID |    W   | C=b'1 | b'0-pad |
              + ------ + ------ + ----- + ------- +
              | 8 bits | 3 bits | 1 bit | 52 bits |
```

Figure 16: SCHC Success ACK message format

The SCHC Compound ACK message MUST be used in case SCHC fragment
losses are found in any window of the SCHC Packet (C=0). The SCHC
Compound ACK message format is shown in Figure 17. The SCHC Compound
ACK can report up to 3 windows with losses. The window number (W)
and its corresponding bitmap MUST be ordered from the lowest-numbered
window number to the highest-numbered window. If window numbered 000

is present in the SCHC Compound ACK, the window number 000 MUST be
placed between the Rule ID and C bit to avoid confusion with padding
bits. The SCHC Compound ACK MUST be 0 padded (Padding bits must be
0).

```
|- SCHC ACK Header -| W=b'x |...|--- W=b'x+i ---|
+-------------------+-------+...+-------+-------+-------+
|RuleID|W=b'x |C=b'0|Bitmap |...|W=b'x+i|Bitmap |b'0-pad|
+------+------+-----+-------+...|-------+-------+-------+
|8 bits|3 bits|1 bit|15 bits|   | 3 bits|15 bits|
```

On top are noted the window number
         of the corresponding bitmap.
Losses are found in windows x,...,x+i.

Figure 17: SCHC Compound ACK message format

4.7.2.4. SCHC Sender-Abort Messages

```
|---- Sender-Abort Header ----|
+ --------------------------- +
| RuleID |   W    | FCN=ALL-1 |
+ ------ + ------ + --------- +
| 8 bits | 3 bits |  5 bits   |
```

Figure 18: SCHC Sender-Abort message format

4.7.2.5. SCHC Receiver-Abort Message

```
|-- Receiver-Abort Header -|
+ ----------------------- + ------- +
| RuleID | W=b'111 | C=b'1 | b'1-pad |
+ ------ + ------- + ----- + ------- +
| 8 bits |  3 bits | 1 bit | 52 bits |
```

Figure 19: SCHC Receiver-Abort message format

4.8. Padding

The Sigfox payload fields have different characteristics in uplink
and downlink.

Uplink frames can contain a payload size from 0 to 12 bytes. The
Sigfox radio protocol allows sending zero bits, one single bit of
information for binary applications (e.g. status), or an integer
number of bytes. Therefore, for 2 or more bits of payload it is
required to add padding to the next integer number of bytes. The
reason for this flexibility is to optimize transmission time and
hence save battery consumption at the device.

Downlink frames on the other hand have a fixed length. The payload
length MUST be 64 bits (i.e. 8 bytes). Hence, if less information
bits are to be transmitted, padding MUST be used with bits equal to
0.

5. Fragmentation Sequence Examples

In this section, some sequence diagrams depicting messages exchanges
for different fragmentation modes and use cases are shown. In the
examples, 'Seq' indicates the Sigfox Sequence Number of the frame
carrying a fragment.

5.1. Uplink No-ACK Examples

The FCN field indicates the size of the data packet. The first
fragment is marked with FCN = X-1, where X is the number of fragments
the message is split into. All fragments are marked with decreasing
FCN values. Last packet fragment is marked with the FCN = All-1
(1111).

Case No losses - All fragments are sent and received successfully.

```
        Sender                              Receiver
          |-------FCN=6 (0110), Seq=1-------->|
          |-------FCN=5 (0101), Seq=2-------->|
          |-------FCN=4 (0100), Seq=3-------->|
          |-------FCN=3 (0011), Seq=4-------->|
          |-------FCN=2 (0010), Seq=5-------->|
          |-------FCN=1 (0001), Seq=6-------->|
          |-------FCN=15 (1111), Seq=7------->| All fragments received
        (End)
```

                   Figure 20: UL No-ACK No-Losses

When the first SCHC fragment is received, the Receiver can calculate
the total number of SCHC fragments that the SCHC Packet is composed

of. For example, if the first fragment is numbered with FCN=6, the
receiver can expect six more messages/fragments (i.e., with FCN going
from 5 downwards, and the last fragment with a FCN equal to 15).

Case losses on any fragment except the first.

```
      Sender                      Receiver
        |-------FCN=6, Seq=1-------->|
        |-------FCN=5, Seq=2----X--->|
        |-------FCN=4, Seq=3-------->|
        |-------FCN=3, Seq=4-------->|
        |-------FCN=2, Seq=5-------->|
        |-------FCN=1, Seq=6-------->|
        |-------FCN=15, Seq=7------->| Missing Fragment - Unable to
reassemble
      (End)
```

Figure 21: UL No-ACK Losses (scenario 1)

5.2. Uplink ACK-on-Error Examples: Single-byte SCHC Header

The single-byte SCHC header ACK-on-Error mode allows sending up to 28
fragments and packet sizes up to 300 bytes. The SCHC fragments may
be delivered asynchronously and DL ACK can be sent opportunistically.

Case No losses

The downlink flag must be enabled in the sender UL message to allow a
DL message from the receiver. The DL Enable in the figures shows
where the sender should enable the downlink, and wait for an ACK.

```
          Sender                       Receiver
             |-----W=0, FCN=6, Seq=1----->|
             |-----W=0, FCN=5, Seq=2----->|
             |-----W=0, FCN=4, Seq=3----->|
             |-----W=0, FCN=3, Seq=4----->|
             |-----W=0, FCN=2, Seq=5----->|
             |-----W=0, FCN=1, Seq=6----->|
  DL Enable |-----W=0, FCN=0, Seq=7----->|
         (no ACK)
             |-----W=1, FCN=6, Seq=8----->|
             |-----W=1, FCN=5, Seq=9----->|
             |-----W=1, FCN=4, Seq=10---->|
  DL Enable |-----W=1, FCN=7, Seq=11---->| All fragments received
             |<------ ACK, W=1, C=1 ------| C=1
         (End)
```

                   Figure 22: UL ACK-on-Error No-Losses

   Case Fragments lost in first window

   In this case, fragments are lost in the first window (W=0). After
   the first All-0 message arrives, the Receiver leverages the
   opportunity and sends an ACK with the corresponding bitmap and C=0.

   After the missing fragments from the first window (W=0) are resent,
   the sender continues transmitting the fragments of the following
   window (W=1) without opening a reception opportunity. Finally, the
   All-1 fragment is sent, the downlink is enabled, and the ACK is
   received with C=1.

```
         Sender                       Receiver
            |-----W=0, FCN=6, Seq=1----->|
            |-----W=0, FCN=5, Seq=2--X-->|
            |-----W=0, FCN=4, Seq=3----->|
            |-----W=0, FCN=3, Seq=4----->|
            |-----W=0, FCN=2, Seq=5--X-->|
            |-----W=0, FCN=1, Seq=6----->|
DL Enable |-----W=0, FCN=0, Seq=7----->| Missing Fragments W=0 => FCN=5, Seq=2
and FCN=2, Seq=5
            |<------ ACK, W=0, C=0 ------| Bitmap:1011011
            |-----W=0, FCN=5, Seq=8----->|
            |-----W=0, FCN=2, Seq=9----->|
            |-----W=1, FCN=6, Seq=10---->|
            |-----W=1, FCN=5, Seq=11---->|
            |-----W=1, FCN=4, Seq=12---->|
DL Enable |-----W=1, FCN=7, Seq=13---->| All fragments received
            |<------ ACK, W=1, C=1 ------| C=1
          (End)
```

                Figure 23: UL ACK-on-Error Losses on First Window

   Case Fragments All-0 lost in first window (W=0)

   In this example, the All-0 of the first window (W=0) is lost.
   Therefore, the Receiver waits for the next All-X message to generate
   the corresponding ACK, notifying the absence of the All-0 of window
   0.

   The sender resends the missing All-0 messages (with any other missing
   fragment from window 0). Note that this behaviour can take place in
   any intermediate window if the All-0 message is lost.

```
        Sender                      Receiver
          |-----W=0, FCN=6, Seq=1----->|
          |-----W=0, FCN=5, Seq=2----->|
          |-----W=0, FCN=4, Seq=3----->|
          |-----W=0, FCN=3, Seq=4----->|
          |-----W=0, FCN=2, Seq=5----->|
          |-----W=0, FCN=1, Seq=6----->| DL Enable
                  |-----W=0, FCN=0, Seq=7--X-->|
        (no ACK)
          |-----W=1, FCN=6, Seq=8----->|
          |-----W=1, FCN=5, Seq=9----->|
          |-----W=1, FCN=4, Seq=10---->|
DL Enable |-----W=1, FCN=7, Seq=11---->| Missing Fragment W=0, FCN=0, Seq=7
          |<------ ACK, W=0, C=0 ------| Bitmap:1111110
          |-----W=0, FCN=0, Seq=13---->| All fragments received
DL Enable |-----W=1, FCN=7, Seq=14---->|
          |<------ ACK, W=1, C=1 ------| C=1
        (End)
```

Figure 24: UL ACK-on-Error All-0 Lost on First Window

In the following diagram, besides the All-0 there are other lost
fragments in the first window (W=0).

```
        Sender                      Receiver
           |-----W=0, FCN=6, Seq=1----->|
           |-----W=0, FCN=5, Seq=2--X-->|
           |-----W=0, FCN=4, Seq=3----->|
           |-----W=0, FCN=3, Seq=4--X-->|
           |-----W=0, FCN=2, Seq=5----->|
           |-----W=0, FCN=1, Seq=6----->|
DL Enable  |-----W=0, FCN=0, Seq=7--X-->|
      (no ACK)
           |-----W=1, FCN=6, Seq=8----->|
           |-----W=1, FCN=5, Seq=9----->|
           |-----W=1, FCN=4, Seq=10---->|
DL Enable  |-----W=1, FCN=7, Seq=11---->| Missing Fragment W=0 => FCN= 5, 3 and
0
           |<------ ACK, W=0, C=0 ------| Bitmap:1010110
           |-----W=0, FCN=5, Seq=13---->|
           |-----W=0, FCN=3, Seq=14---->|
           |-----W=0, FCN=0, Seq=15---->| All fragments received
DL Enable  |-----W=1, FCN=7, Seq=16---->|
           |<------ ACK, W=1, C=1 ------| C=1
        (End)
```

Figure 25: UL ACK-on-Error All-0 and other Fragments Lost on First
                                Window

In the next examples, there are losses in both the first (W=0) and
second (W=1) windows. The retransmission cycles after the All-1 is
sent (i.e., not in intermediate windows) should always finish with an
with an All-1. In case an intermediate All-0 is lost and then
retransmitted, the All-1 is resent after, as it serves as an ACK
Request message to confirm the correct reception of the retransmitted
fragments.

```
          Sender                              Receiver
             |-----W=0, FCN=6 (110), Seq=1----->|
             |-----W=0, FCN=5 (101), Seq=2--X-->|
             |-----W=0, FCN=4 (100), Seq=3----->|
             |-----W=0, FCN=3 (011), Seq=4--X-->|
             |-----W=0, FCN=2 (010), Seq=5----->|
             |-----W=0, FCN=1 (001), Seq=6----->|
DL enable |-----W=0, FCN=0 (000), Seq=7--X-->|
      (no ACK)
             |-----W=1, FCN=6 (110), Seq=8--X-->|
             |-----W=1, FCN=5 (101), Seq=9----->|
             |-----W=1, FCN=4 (011), Seq=10-X-->|
DL enable |-----W=1, FCN=7 (111), Seq=11---->| Missing Fragment W=0 => FCN= 5,
3 and 0
             |<--------- ACK, W=0, C=0 ---------| Bitmap:1010110
             |-----W=0, FCN=5 (101), Seq=13---->|
             |-----W=0, FCN=3 (011), Seq=14---->|
             |-----W=0, FCN=0 (000), Seq=15---->|
DL enable |-----W=1, FCN=7 (111), Seq=16---->| Missing Fragment W=1 => FCN= 6
and 4
             |<--------- ACK, W=1, C=0 ---------| Bitmap:0100001
             |-----W=1, FCN=6 (110), Seq=18---->|
             |-----W=1, FCN=4 (011), Seq=19---->| All fragments received
DL enable |-----W=1, FCN=7 (111), Seq=20---->|
             |<--------- ACK, W=1, C=1 ---------| C=1
          (End)
```

   Figure 26: UL ACK-on-Error All-0 and other Fragments Lost on First
                        and Second Windows (1)

   Similar case as above, but with less fragments in the second window
   (W=1)

```
        Sender                           Receiver
           |-----W=0, FCN=6 (110), Seq=1----->|
           |-----W=0, FCN=5 (101), Seq=2--X-->|
           |-----W=0, FCN=4 (100), Seq=3----->|
           |-----W=0, FCN=3 (011), Seq=4--X-->|
           |-----W=0, FCN=2 (010), Seq=5----->|
           |-----W=0, FCN=1 (001), Seq=6----->|
DL enable |-----W=0, FCN=0 (000), Seq=7--X-->|
       (no ACK)
           |-----W=1, FCN=6 (110), Seq=8--X-->|
DL enable |-----W=1, FCN=7 (111), Seq=9----->| Missing Fragment W=0 => FCN= 5,
3 and 0
           |<--------- ACK, W=0, C=0 ---------| Bitmap:1010110
           |-----W=0, FCN=5 (101), Seq=10---->|
           |-----W=0, FCN=3 (011), Seq=11---->|
DL enable |-----W=0, FCN=0 (000), Seq=12---->| Missing Fragment W=1 => FCN= 6
and 4
           |<--------- ACK, W=1, C=0 ---------| Bitmap:0000001
           |-----W=1, FCN=6 (110), Seq=15---->| All fragments received
DL enable |-----W=1, FCN=7 (111), Seq=17---->|
           |<--------- ACK, W=1, C=1 ---------| C=1
       (End)
```

Figure 27: UL ACK-on-Error All-0 and other Fragments Lost on First
                       and Second Windows (2)

Case ACK is lost

SCHC over Sigfox does not implement the SCHC ACK REQ message.
Instead it uses the SCHC All-1 message to request an ACK, when
required.

```
          Sender                      Receiver
            |-----W=0, FCN=6, Seq=1----->|
            |-----W=0, FCN=5, Seq=2----->|
            |-----W=0, FCN=4, Seq=3----->|
            |-----W=0, FCN=3, Seq=4----->|
            |-----W=0, FCN=2, Seq=5----->|
            |-----W=0, FCN=1, Seq=6----->|
  DL Enable |-----W=0, FCN=0, Seq=7----->|
        (no ACK)
            |-----W=1, FCN=6, Seq=8----->|
            |-----W=1, FCN=5, Seq=9----->|
            |-----W=1, FCN=4, Seq=10---->|
  DL Enable |-----W=1, FCN=7, Seq=11---->| All fragments received
            |<------ ACK, W=1, C=1 ---X--| C=1
  DL Enable |-----W=1, FCN=7, Seq=13---->| RESEND ACK
            |<------ ACK, W=1, C=1 ------| C=1
         (End)
```

Figure 28: UL ACK-on-Error ACK Lost

The number of times an ACK will be requested is determined by the
MAX_ACK_REQUESTS.

5.3. SCHC Abort Examples

Case SCHC Sender-Abort

The sender may need to send a Sender-Abort to stop the current
communication. This may happen, for example, if the All-1 has been
sent MAX_ACK_REQUESTS times.

```
          Sender                     Receiver
             |-----W=0, FCN=6, Seq=1----->|
             |-----W=0, FCN=5, Seq=2----->|
             |-----W=0, FCN=4, Seq=3----->|
             |-----W=0, FCN=3, Seq=4----->|
             |-----W=0, FCN=2, Seq=5----->|
             |-----W=0, FCN=1, Seq=6----->|
  DL Enable  |-----W=0, FCN=0, Seq=7----->|
        (no ACK)
             |-----W=1, FCN=6, Seq=8----->|
             |-----W=1, FCN=5, Seq=9----->|
             |-----W=1, FCN=4, Seq=10---->|
  DL Enable  |-----W=1, FCN=7, Seq=11---->| All fragments received
             |<------ ACK, W=1, C=1 ---X--| C=1
  DL Enable  |-----W=1, FCN=7, Seq=14---->| RESEND ACK  (1)
             |<------ ACK, W=1, C=1 ---X--| C=1
  DL Enable  |-----W=1, FCN=7, Seq=15---->| RESEND ACK  (2)
             |<------ ACK, W=1, C=1 ---X--| C=1
  DL Enable  |-----W=1, FCN=7, Seq=16---->| RESEND ACK  (3)
             |<------ ACK, W=1, C=1 ---X--| C=1
  DL Enable  |-----W=1, FCN=7, Seq=17---->| RESEND ACK  (4)
             |<------ ACK, W=1, C=1 ---X--| C=1
  DL Enable  |-----W=1, FCN=7, Seq=18---->| RESEND ACK  (5)
             |<------ ACK, W=1, C=1 ---X--| C=1
  DL Enable  |----Sender-Abort, Seq=19--->| exit with error condition
          (End)
```

                Figure 29: UL ACK-on-Error Sender-Abort

   Case Receiver-Abort

   The reciever may need to send a Receiver-Abort to stop the current
   communication. This message can only be sent after a DL enable.

```
         Sender                        Receiver
           |-----W=0, FCN=6, Seq=1----->|
           |-----W=0, FCN=5, Seq=2----->|
           |-----W=0, FCN=4, Seq=3----->|
           |-----W=0, FCN=3, Seq=4----->|
           |-----W=0, FCN=2, Seq=5----->|
           |-----W=0, FCN=1, Seq=6----->|
 DL Enable |-----W=0, FCN=0, Seq=7----->|
           |<-------  RECV ABORT -------| under-resourced
       (Error)
```

Figure 30: UL ACK-on-Error Receiver-Abort

6. Security considerations

The radio protocol authenticates and ensures the integrity of each
message. This is achieved by using a unique device ID and an AES-128
based message authentication code, ensuring that the message has been
generated and sent by the device with the ID claimed in the message.

Application data can be encrypted at the application level or not,
depending on the criticality of the use case. This flexibility
allows providing a balance between cost and effort vs. risk. AES-128
in counter mode is used for encryption. Cryptographic keys are
independent for each device. These keys are associated with the
device ID and separate integrity and confidentiality keys are pre-
provisioned. A confidentiality key is only provisioned if
confidentiality is to be used.

The radio protocol has protections against reply attacks, and the
cloud-based core network provides firewalling protection against
undesired incoming communications.

7. Acknowledgements

The authors would like to thank Clement Mannequin, Rafael Vidal and
Antonis Platis for their useful comments and implementation design
considerations.

8. References

8.1. Normative References

[RFC8376]   Farrell, S., Ed., "Low-Power Wide Area Network (LPWAN)
            Overview", RFC 8376, DOI 10.17487/RFC8376, May 2018,
            <https://www.rfc-editor.org/info/rfc8376>.

[RFC8724]   Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC.
            Zuniga, "SCHC: Generic Framework for Static Context Header
            Compression and Fragmentation", RFC 8724,
            DOI 10.17487/RFC8724, April 2020,
            <https://www.rfc-editor.org/info/rfc8724>.

8.2. Informative References

[sigfox-callbacks]
            Sigfox, "Sigfox Callbacks",
            <https://support.sigfox.com/docs/callbacks-documentation>.

[sigfox-spec]
            Sigfox, "Sigfox Radio Specifications",
            <https://build.sigfox.com/sigfox-device-radio-
            specifications>.

Authors' Addresses

Juan Carlos Zuniga
SIGFOX
Montreal QC
Canada

Email: JuanCarlos.Zuniga@sigfox.com
URI: http://www.sigfox.com/

Carles Gomez
Universitat Politecnica de Catalunya
C/Esteve Terradas, 7
08860 Castelldefels
Spain

Email: carlesgo@entel.upc.edu

Sergio Aguilar
Universitat Politecnica de Catalunya
C/Esteve Terradas, 7
08860 Castelldefels
Spain

Email: sergio.aguilar.romero@upc.edu

Laurent Toutain
IMT-Atlantique
2 rue de la Chataigneraie
CS 17607
35576 Cesson-Sevigne Cedex
France

Email: Laurent.Toutain@imt-atlantique.fr

Sandra Cespedes
NIC Labs, Universidad de Chile
Av. Almte. Blanco Encalada 1975
Santiago
Chile

Email: scespedes@niclabs.cl

Diego Wistuba
NIC Labs, Universidad de Chile
Av. Almte. Blanco Encalada 1975
Santiago
Chile

Email: wistuba@niclabs.cl