Stream: Internet Engineering Task Force (IETF)
RFC: 9441
Updates: 8724, 9363
Category: Standards Track
Published: July 2023
ISSN: 2070-1721
Authors: JC. Zúñiga, *Cisco*
C. Gomez, *Universitat Politècnica de Catalunya*
S. Aguilar, *Universitat Politècnica de Catalunya*
L. Toutain, *IMT-Atlantique*
S. Céspedes, *Concordia University*
D. Wistuba, *NIC Labs, Universidad de Chile*

RFC 9441
Static Context Header Compression (SCHC) Compound Acknowledgement (ACK)

Abstract

This document updates the Static Context Header Compression (SCHC) and fragmentation protocol (RFC 8724) and the corresponding YANG module (RFC 9363). It defines a SCHC Compound Acknowledgement (ACK) message format and procedure, which are intended to reduce the number of response transmissions (i.e., SCHC ACKs) in the ACK-on-Error Mode, by accumulating bitmaps of several windows in a single SCHC message (i.e., the SCHC Compound ACK).

Both the message format and procedure are generic, so they can be used, for instance, by any of the four Low-Power Wide Area Network (LPWAN) technologies defined in RFC 8376, which are Sigfox, Long Range Wide Area Network (LoRaWAN), Narrowband Internet of Things (NB-IoT), and IEEE 802.15.4w.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9441.

Copyright Notice

Table of Contents

1. Introduction

The Generic Framework for Static Context Header Compression (SCHC) and Fragmentation specification [RFC8724] describes two mechanisms: i) a protocol header compression scheme and ii) a frame fragmentation and loss recovery functionality. Either can be used on top of radio technologies, such as the four Low-Power Wide Area Networks (LPWANs) listed in [RFC8376],

which are Sigfox, LoRaWAN, NB-IoT, and IEEE 802.15.4w. These LPWANs have similar characteristics, such as star-oriented topologies, network architecture, and connected devices with built-in applications.

SCHC offers a great level of flexibility to accommodate all these LPWAN technologies. Even though there are a 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.

In ACK-on-Error mode in [RFC8724], the SCHC Packet is fragmented into pieces called tiles, where all tiles are the same size except for the last one, which can be smaller. Successive tiles are grouped in windows of fixed size. A SCHC Fragment carries one or several contiguous tiles, which may span multiple windows. When sending all tiles from all windows, the last tile is sent in an All-1 SCHC Fragment. The SCHC receiver will send a SCHC ACK reporting on the reception of exactly one window of tiles after receiving the All-1 SCHC Fragment. In case of SCHC Fragment losses, a bitmap is added to the failure SCHC ACK, where each bit in the bitmap corresponds to a tile in the window. If SCHC Fragment losses span multiple windows, the SCHC receiver will send one failure SCHC ACK per window with losses.

This document updates the SCHC protocol for frame fragmentation and loss recovery. It defines a SCHC Compound ACK format and procedure, which are intended to reduce the number of response transmissions (i.e., SCHC ACKs) in the ACK-on-Error mode of SCHC. The SCHC Compound ACK extends the failure SCHC ACK message format so that it can contain several bitmaps, with each bitmap being identified by its corresponding window number. The SCHC Compound ACK is backwards compatible with the SCHC ACK as defined in [RFC8724], and introduces flexibility, as the receiver has the capability to respond to the All-0 SCHC Fragment, providing more Downlink opportunities and therefore adjusting to the delay requirements of the application.

2. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

3. SCHC Compound ACK

The SCHC Compound ACK is a failure SCHC ACK message that can contain several bitmaps, with each bitmap being identified by its corresponding window number. In [RFC8724], the failure SCHC ACK message only contains one bitmap corresponding to one window. The SCHC

Compound ACK extends this format, allowing more windows to be acknowledged in a single ACK and reducing the total number of failure SCHC ACK messages, especially when fragment losses are present in intermediate windows.

The SCHC Compound ACK **MAY** be used in fragmentation modes that use windows and that allow reporting the bitmaps of multiple windows at the same time; otherwise, the SCHC Compound ACK **MUST NOT** be used.

The SCHC Compound ACK:

- provides feedback only for windows with fragment losses,
- has a variable size that depends on the number of windows with fragment losses being reported in the single SCHC Compound ACK,
- includes the window number (i.e., W) of each bitmap,
- might not cover all windows with fragment losses of a SCHC Packet, and
- is distinguishable from the SCHC Receiver-Abort.

3.1. SCHC Compound ACK Message Format

Figure 1 shows the success SCHC ACK format, i.e., when all fragments have been correctly received (C=1), as defined in [RFC8724].

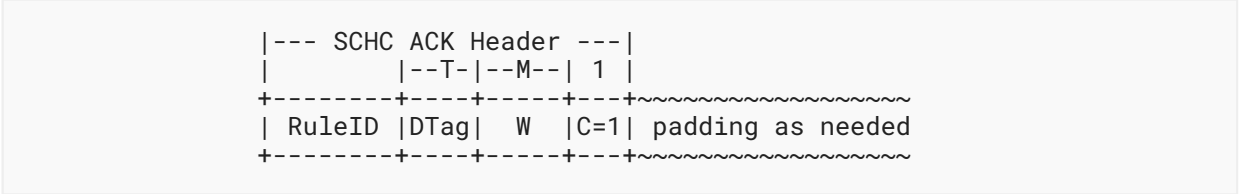

Figure 1: SCHC Success ACK Message Format, as Defined in RFC 8724

In case SCHC Fragment losses are found in any of the windows of the SCHC Packet, the SCHC Compound ACK **MAY** be used. The SCHC Compound ACK message format is shown in Figures 2 and 3.

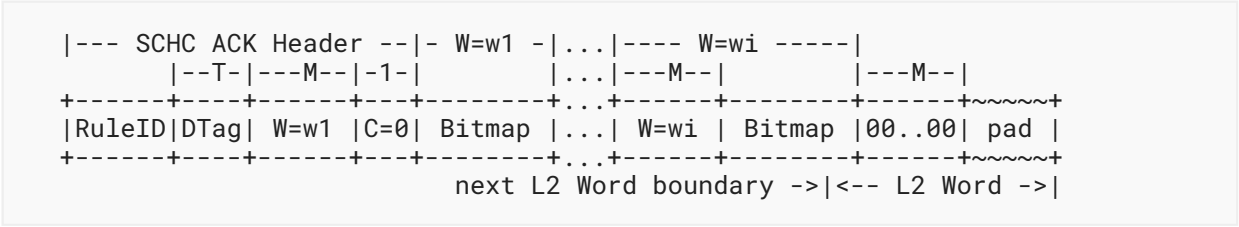

Figure 2: SCHC Compound ACK Message Format. Losses are found in windows W = w1,...,wi, where w1 < w2 <...< wi.

The SCHC Compound ACK groups the window number (W) with its corresponding bitmap. Window numbers do not need to be contiguous. However, the window numbers and their corresponding bitmaps included in the SCHC Compound ACK message **MUST** be ordered from the lowest-numbered to the highest-numbered window. Hence, if the bitmap of window number zero is present in the SCHC Compound ACK message, it **MUST** always be the first one in order and its window number **MUST** be placed in the SCHC ACK Header.

If M or more padding bits would be needed after the last bitmap in the message to fill the last layer two (L2) Word, M bits at 0 **MUST** be appended after the last bitmap, and then padding is applied as needed (see Figure 2). Since window number 0 (if present in the message) is placed as w1, the M bits set to zero can't be confused with window number 0; therefore, they signal the end of the SCHC Compound ACK message.

Figure 3 shows the case when the required padding bits are strictly less than M bits. In this case, the L2 Maximum Transmission Unit (MTU) does not leave room for any extra window value, let alone any bitmap, thereby signaling the end of the SCHC Compound ACK message.

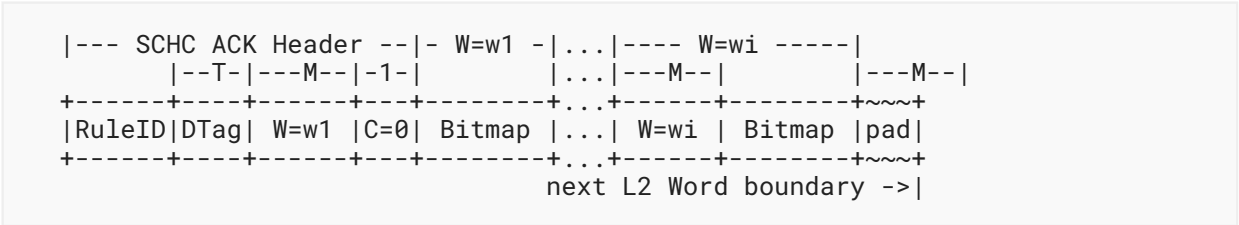

Figure 3: SCHC Compound ACK Message Format with Less than M Padding Bits. Losses are found in windows W = w1,...,wi, where w1 < w2 <...< wi.

The SCHC Compound ACK **MUST NOT** use the Compressed Bitmap format for intermediate windows/bitmaps (i.e., bitmaps that are not the last one of the SCHC Compound ACK message); therefore, intermediate bitmap fields **MUST** be of size WINDOW_SIZE. Hence, the SCHC Compound ACK **MAY** use a Compressed Bitmap format only for the last bitmap in the message. The optional usage of this Compressed Bitmap for the last bitmap **MUST** be specified by the technology-specific SCHC Profile.

The case where the last bitmap is effectively compressed corresponds to Figure 3, with the last bitmap ending (by construction) on an L2 Word boundary, therefore resulting in no padding at all.

Figure 4 illustrates a bitmap compression example of a SCHC Compound ACK, where the bitmap of the last window (wi) indicates that the first tile has not been correctly received. Because the compression algorithm resulted in effective compression, no padding is needed.

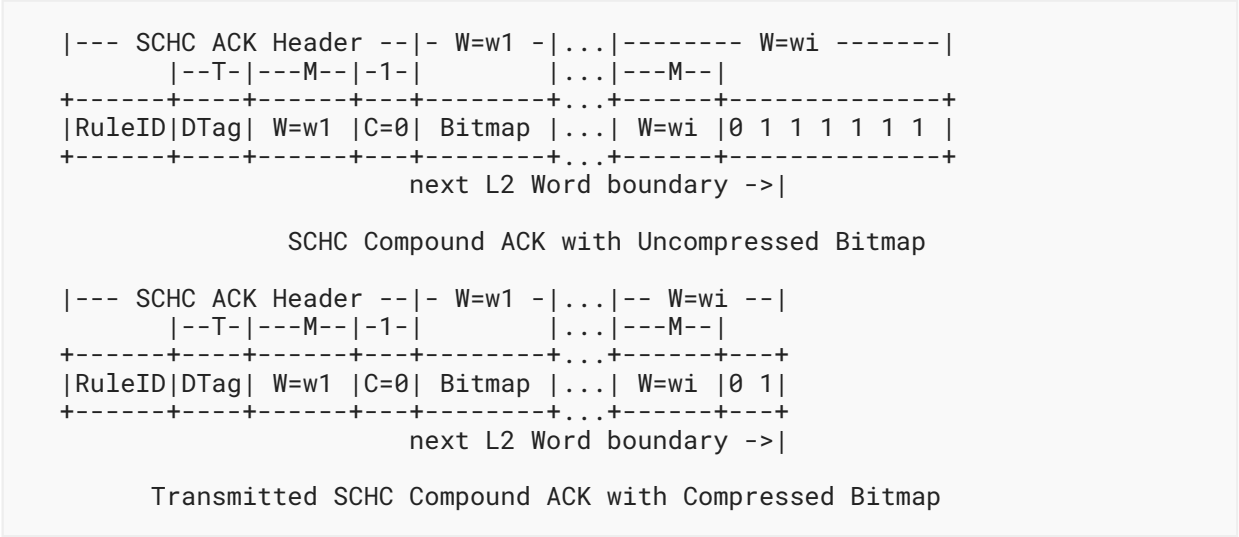

Figure 4: SCHC Compound ACK Message Format with Compressed Bitmap and No Padding Added. Losses are found in windows W = w1,...,wi, where w1 < w2 <...< wi.

Figure 5 illustrates another bitmap compression example of a SCHC Compound ACK, where the bitmap of the last window (wi) indicates that the second and the fourth tiles have not been correctly received. In this example, the compression algorithm does not result in effective compression of the last bitmap. Besides, because more than M bits of padding would be needed to fill the last L2 Word, M bits at 0 are appended to the message before padding is applied.

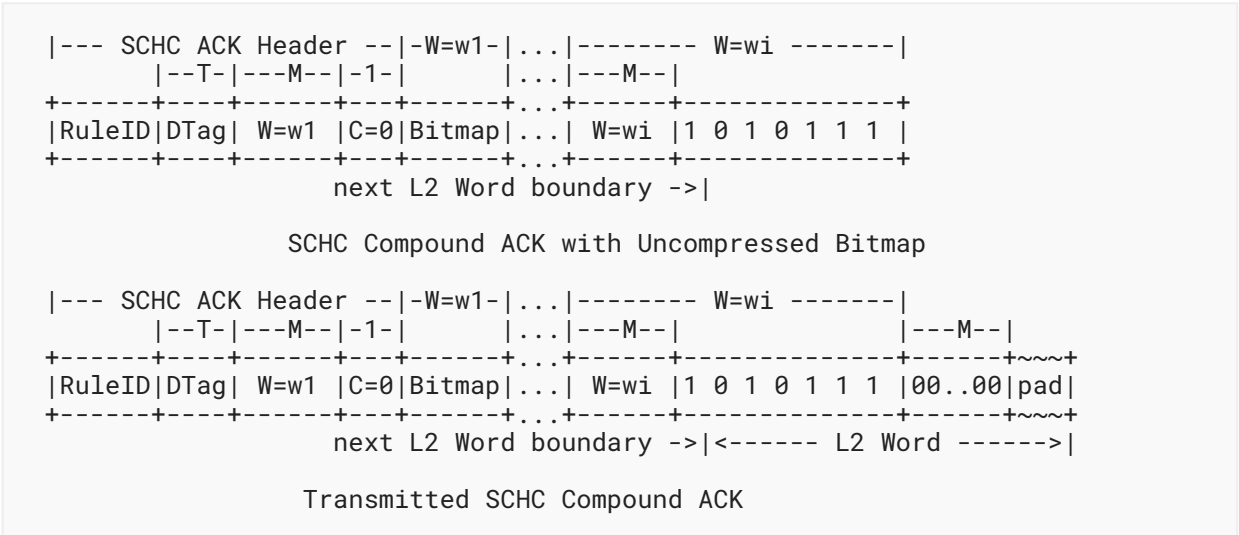

Figure 5: SCHC Compound ACK Message Format with Compressed Bitmap and Padding Added to Reach the L2 Boundary. Losses are found in windows W = w1,...,wi, where w1 < w2 <...<wi.

If a SCHC sender gets a SCHC Compound ACK with invalid window numbers, such as duplicate W values or W values not sent yet, it **MUST** discard the whole SCHC Compound ACK message.

> Note that SCHC Compound ACKs are distinguishable from the Receiver-Abort message in the same way that regular SCHC ACKs are distinguishable, since the Receiver-Abort pattern never occurs in a legitimate SCHC Compound ACK [RFC8724].

3.2. SCHC Compound ACK Behavior

The SCHC ACK-on-Error behavior is described in Section 8.4.3 of [RFC8724]. The present document slightly modifies this behavior. In the baseline SCHC specification, a SCHC ACK reports only one bitmap for the reception of exactly one window of tiles. The present SCHC Compound ACK specification extends the SCHC ACK message format so that it can contain several bitmaps, with each bitmap being identified by its corresponding window number.

As presented in [RFC8724], the SCHC ACK format can be considered a special SCHC Compound ACK case in which it reports only the tiles of one window. Therefore, the SCHC Compound ACK is backwards compatible with the SCHC ACK format presented in [RFC8724]. The receiver can assume that the sender does not support the SCHC Compound ACK if, although the SCHC Compound ACK sent by the receiver reports losses in more than one window, the sender does not resend any tiles from windows other than the first window reported in the SCHC Compound ACK. In that case, the receiver can send SCHC Compound ACKs with only one window of tiles.

Also, some flexibility is introduced with respect to [RFC8724] in that the receiver has the capability to respond (or not) to the All-0 with a SCHC Compound ACK, depending on certain parameters, like network conditions, sender buffer/cache size, and supported application delay. Note that even though the protocol allows for such flexibility, the actual decision criteria is not specified in this document. The application **MUST** set expiration timer values according to when the feedback is expected to be received, e.g., after the All-0 or after the All-1.

Section 3.2.1 (and its subsections) replaces the complete Section 8.4.3 (and its subsections) of [RFC8724].

3.2.1. ACK-on-Error Mode (Replaces Section 8.4.3, RFC 8724)

The ACK-on-Error mode supports L2 technologies that have variable MTU and out-of-order delivery. It requires an L2 that provides a feedback path from the reassembler to the fragmenter. See Appendix F for a discussion on using ACK-on-Error mode on quasi-bidirectional links.

In ACK-on-Error mode, windows are used.

All tiles except the last one and the penultimate one **MUST** be of equal size, hereafter called "regular". The size of the last tile **MUST** be smaller than or equal to the regular tile size. Regarding the penultimate tile, a Profile **MUST** pick one of the following two options:

- The penultimate tile size **MUST** be the regular tile size, or
- the penultimate tile size **MUST** be either the regular tile size or the regular tile size minus one L2 Word.

A SCHC Fragment message carries one or several contiguous tiles, which may span multiple windows. A SCHC Compound ACK reports on the reception of one window of tiles or several windows of tiles, each one identified by its window number.

See Figure 6 (see Figure 23 of RFC 8724) for an example.

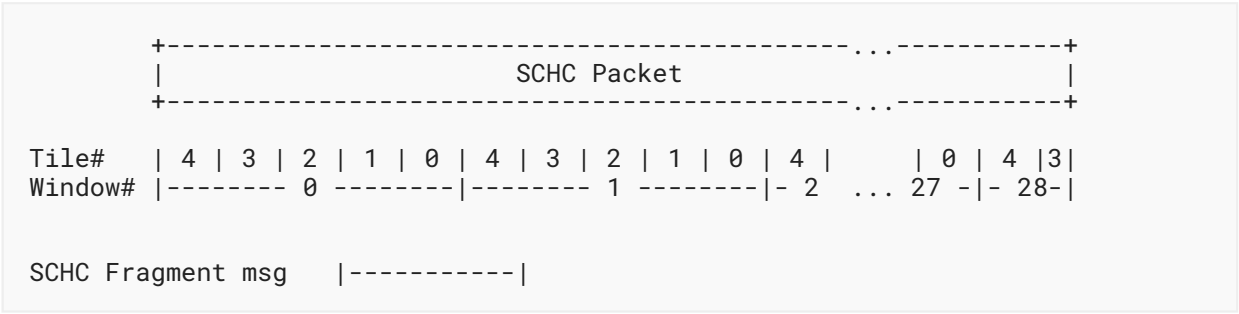

Figure 6: SCHC Packet Fragmented in Tiles, ACK-on-Error Mode (Figure 23 in RFC 8724)

The W field is wide enough that it unambiguously represents an absolute window number. The fragment receiver sends SCHC Compound ACKs to the fragment sender about windows for which tiles are missing. No SCHC Compound ACK is sent by the fragment receiver for windows that it knows have been fully received.

The fragment sender retransmits SCHC Fragments for tiles that are reported missing. It can advance to next windows even before it has ascertained that all tiles belonging to previous windows have been correctly received, and it can still later retransmit SCHC Fragments with tiles belonging to previous windows. Therefore, the sender and the receiver may operate in a decoupled fashion. The fragmented SCHC Packet transmission concludes when:

- integrity checking shows that the fragmented SCHC Packet has been correctly reassembled at the receive end, and this information has been conveyed back to the sender,
- too many retransmission attempts have been made, or
- the receiver determines that the transmission of this fragmented SCHC Packet has been inactive for too long.

Each Profile **MUST** specify which RuleID value(s) corresponds to SCHC F/R messages operating in this mode.

The W field **MUST** be present in the SCHC F/R messages.

Each Profile, for each RuleID value, **MUST** define:

- the tile size (a tile does not need to be a duplicate of an L2 Word, but it **MUST** be at least the size of an L2 Word),
- the value of M,
- the value of N,
- the value of WINDOW_SIZE, which **MUST** be strictly less than 2^N,

- the size and algorithm for the RCS field,
- the value of T,
- the value of MAX_ACK_REQUESTS,
- the expiration time of the Retransmission Timer,
- the expiration time of the Inactivity Timer,
- if the last tile is carried in a Regular SCHC Fragment or an All-1 SCHC Fragment (see Section 3.2.1.1 (Section 8.4.3.1 in [RFC8724]),
- if the penultimate tile **MAY** be one L2 Word smaller than the regular tile size (in this case, the regular tile size **MUST** be at least twice the L2 Word size),
- usage or not of the SCHC Compound ACK message, and
- usage or not of the Compressed Bitmap format in the last window of the SCHC Compound ACK message.

For each active pair of RuleID and DTag values, the sender **MUST** maintain:

- one Attempts counter and
- one Retransmission Timer.

For each active pair of RuleID and DTag values, the receiver **MUST** maintain:

- one Attempts counter and
- one Inactivity Timer.

3.2.1.1. Sender Behavior (Replaces Section 8.4.3.1, RFC 8724)

At the beginning of the fragmentation of a new SCHC Packet:

- the fragment sender **MUST** select a RuleID and DTag value pair for this SCHC Packet. A Rule **MUST NOT** be selected if the values of M and WINDOW_SIZE for that Rule are such that the SCHC Packet cannot be fragmented in (2^M) * WINDOW_SIZE tiles or less.
- the fragment sender **MUST** initialize the Attempts counter to 0 for that RuleID and DTag value pair.

A Regular SCHC Fragment message carries in its payload one or more tiles. If more than one tile is carried in one Regular SCHC Fragment:

- the selected tiles **MUST** be contiguous in the original SCHC Packet, and
- they **MUST** be placed in the SCHC Fragment Payload adjacent to one another, in the order they appear in the SCHC Packet, from the start of the SCHC Packet toward its end.

Tiles that are not the last one **MUST** be sent in Regular SCHC Fragments as specified in Section 8.3.1.1. The FCN field **MUST** contain the tile index of the first tile sent in that SCHC Fragment.

In a Regular SCHC Fragment message, the sender **MUST** fill the W field with the window number of the first tile sent in that SCHC Fragment.

A Profile **MUST** define if the last tile of a SCHC Packet is sent:

- in a Regular SCHC Fragment, alone or as part of a multi-tiles Payload,
- alone in an All-1 SCHC Fragment, or
- with either one of the above two methods.

In an All-1 SCHC Fragment message, the sender **MUST** fill the W field with the window number of the last tile of the SCHC Packet.

The fragment sender **MUST** send SCHC Fragments such that, all together, they contain all the tiles of the fragmented SCHC Packet.

The fragment sender **MUST** send at least one All-1 SCHC Fragment.

In doing the two items above, the sender **MUST** ascertain that the receiver will not receive the last tile through both a Regular SCHC Fragment and an All-1 SCHC Fragment.

The fragment sender **MUST** listen for SCHC Compound ACK messages after having sent:

- an All-1 SCHC Fragment or
- a SCHC ACK REQ.

A Profile **MAY** specify other times at which the fragment sender **MUST** listen for SCHC Compound ACK messages. For example, this could be after sending a complete window of tiles.

Each time a fragment sender sends an All-1 SCHC Fragment or a SCHC ACK REQ:

- it **MUST** increment the Attempts counter, and
- it **MUST** reset the Retransmission Timer.

On Retransmission Timer expiration:

- if the Attempts counter is strictly less than MAX_ACK_REQUESTS, the fragment sender **MUST** send either the All-1 SCHC Fragment or a SCHC ACK REQ with the W field corresponding to the last window,
- otherwise, the fragment sender **MUST** send a SCHC Sender-Abort, and it **MAY** exit with an error condition.

All message receptions being discussed in the rest of this section are to be understood as "matching the RuleID and DTag pair being processed", even if not spelled out, for brevity.

On receiving a SCHC Compound ACK:

- if one of the W fields in the SCHC Compound ACK corresponds to the last window of the SCHC Packet:
  - if the C bit is set, the sender **MAY** exit successfully.

- otherwise:
  - if the Profile mandates that the last tile be sent in an All-1 SCHC Fragment:
    - if the SCHC Compound ACK shows no missing tile at the receiver, the sender:
      - **MUST** send a SCHC Sender-Abort and
      - **MAY** exit with an error condition.
    - otherwise:
      - the fragment sender **MUST** send SCHC Fragment messages containing all the tiles of all the windows that are reported missing in the SCHC Compound ACK.
      - if the last of these SCHC Fragment messages is not an All-1 SCHC Fragment, then the fragment sender **MAY** either send, in addition, a SCHC ACK REQ with the W field corresponding to the last window or repeat the All-1 SCHC Fragment to ask the receiver to confirm that all tiles have been correctly received.
      - in doing the two items above, the sender **MUST** ascertain that the receiver will not receive the last tile through both a Regular SCHC Fragment and an All-1 SCHC Fragment.
    - otherwise:
      - if the SCHC Compound ACK shows no missing tile at the receiver, the sender **MUST** send the All-1 SCHC Fragment
      - otherwise:
        - the fragment sender **MUST** send SCHC Fragment messages containing all the tiles that are reported missing in the SCHC Compound ACK.
        - the fragment sender **MUST** then send either the All-1 SCHC Fragment or a SCHC ACK REQ with the W field corresponding to the last window.
- otherwise, the fragment sender:
  - **MUST** send SCHC Fragment messages containing the tiles that are reported missing in the SCHC Compound ACK.
  - then, it **MAY** send a SCHC ACK REQ with the W field corresponding to the last window.

See Figure 43 for one among several possible examples of a Finite State Machine implementing a sender behavior obeying this specification.

3.2.1.2. Receiver Behavior (Replaces Section 8.4.3.2, RFC 8724)

On receiving a SCHC Fragment with a RuleID and DTag pair not being processed at that time:

- the receiver **SHOULD** check that the DTag value has not recently been used for that RuleID value, thereby ensuring that the received SCHC Fragment is not a remnant of a prior fragmented SCHC Packet transmission. The initial value of the Inactivity Timer is the **RECOMMENDED** lifetime for the DTag value at the receiver. If the SCHC Fragment is determined to be such a remnant, the receiver **MAY** silently ignore it and discard it.

- the receiver **MUST** start a process to assemble a new SCHC Packet with that RuleID and DTag value pair. The receiver **MUST** start an Inactivity Timer for that RuleID and DTag value pair. It **MUST** initialize an Attempts counter to 0 for that RuleID and DTag value pair. If the receiver is under-resourced to do this, it **MUST** respond to the sender with a SCHC Receiver-Abort.

On reception of any SCHC F/R message for the RuleID and DTag pair being processed, the receiver **MUST** reset the Inactivity Timer pertaining to that RuleID and DTag pair.

All message receptions being discussed in the rest of this section are to be understood as "matching the RuleID and DTag pair being processed", even if not spelled out, for brevity.

On receiving a SCHC Fragment message, the receiver determines what tiles were received, based on the payload length and on the W and FCN fields of the SCHC Fragment.

- if the FCN is All-1 and if a Payload is present, the full SCHC Fragment Payload **MUST** be assembled including the padding bits. This is because the size of the last tile is not known by the receiver; therefore, padding bits are indistinguishable from the tile data bits, at this stage. They will be removed by the SCHC C/D sublayer. If the size of the SCHC Fragment Payload exceeds or equals the size of one regular tile plus the size of an L2 Word, this **SHOULD** raise an error flag.
- otherwise, tiles **MUST** be assembled based on the a priori known tile size.
  - If allowed by the Profile, the end of the payload **MAY** contain the last tile, which may be shorter. Padding bits are indistinguishable from the tile data bits, at this stage.
  - The payload may contain the penultimate tile that, if allowed by the Profile, **MAY** be exactly one L2 Word shorter than the regular tile size.
  - Otherwise, padding bits **MUST** be discarded. This is possible because:
    - the size of the tiles is known a priori,
    - tiles are larger than an L2 Word, and
    - padding bits are always strictly less than an L2 Word.

On receiving a SCHC All-0 SCHC Fragment:

- if the receiver knows of any windows with missing tiles for the packet being reassembled (and depending on certain parameters, like network conditions, sender buffer/cache size, and supported application delay, among others), it **MAY** return a SCHC Compound ACK for the missing tiles, starting from the lowest-numbered window.

On receiving a SCHC ACK REQ or an All-1 SCHC Fragment:

- if the receiver knows of any windows with missing tiles for the packet being reassembled, it **MUST** return a SCHC Compound ACK for the missing tiles, starting from the lowest-numbered window.

- otherwise:
  - if it has received at least one tile, it **MUST** return a SCHC Compound ACK for the highest-numbered window it currently has tiles for,
  - otherwise, it **MUST** return a SCHC Compound ACK for window number 0.

A Profile **MAY** specify other times and circumstances at which a receiver sends a SCHC Compound ACK and which window the SCHC Compound ACK reports about in these circumstances.

Upon sending a SCHC Compound ACK, the receiver **MUST** increase the Attempts counter.

After receiving an All-1 SCHC Fragment, a receiver **MUST** check the integrity of the reassembled SCHC Packet at least every time it prepares to send a SCHC Compound ACK for the last window.

Upon receiving a SCHC Sender-Abort, the receiver **MAY** exit with an error condition.

Upon expiration of the Inactivity Timer, the receiver **MUST** send a SCHC Receiver-Abort, and it **MAY** exit with an error condition.

On the Attempts counter exceeding MAX_ACK_REQUESTS, the receiver **MUST** send a SCHC Receiver-Abort, and it **MAY** exit with an error condition.

Reassembly of the SCHC Packet concludes when:

- a Sender-Abort has been received,
- the Inactivity Timer has expired,
- the Attempts counter has exceeded MAX_ACK_REQUESTS, or
- at least an All-1 SCHC Fragment has been received and integrity checking of the reassembled SCHC Packet is successful.

See Figure 44 for one among several possible examples of a Finite State Machine implementing a receiver behavior obeying this specification. The example provided is meant to match the sender Finite State Machine of Figure 43.

4. SCHC Compound ACK Example

Figure 7 shows an example transmission of a SCHC Packet in ACK-on-Error mode using the SCHC Compound ACK. In the example, the SCHC Packet is fragmented in 14 tiles, with N=3, WINDOW_SIZE=7, M=2, and two lost SCHC fragments. Only 1 SCHC Compound ACK is generated.

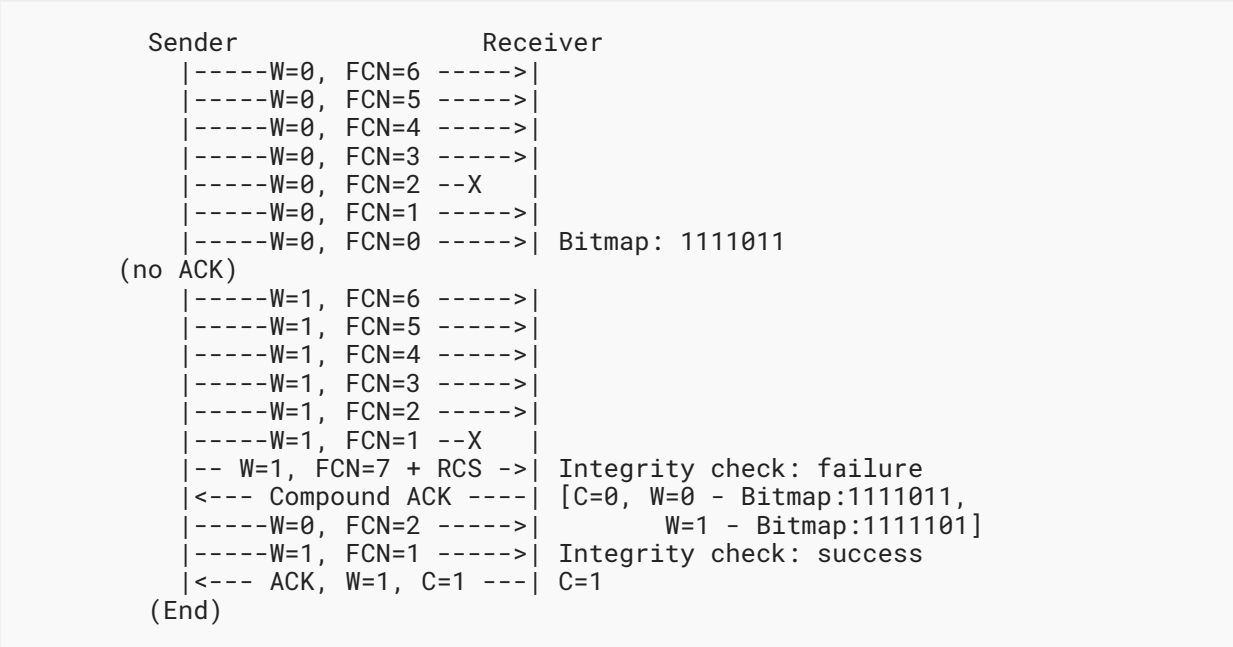

Figure 7: SCHC Compound ACK Message Sequence Example

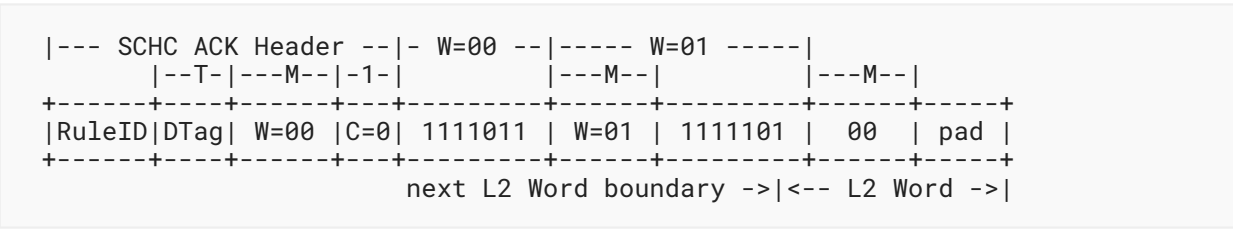

Figure 8: SCHC Compound ACK Message Format Example: Losses are Found in Windows 00 and 01

5. SCHC Compound ACK YANG Data Model

This document also extends the SCHC YANG data model defined in [RFC9363] by including a new leaf in the Ack-on-Error fragmentation mode to describe both the option to use the SCHC Compound ACK, as well as its bitmap format.

5.1. SCHC YANG Data Model Extension

```
<CODE BEGINS> file "ietf-schc-compound-ack@2023-07-26.yang"

module ietf-schc-compound-ack {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-schc-compound-ack";
  prefix schc-compound-ack;

  import ietf-schc {
    prefix schc;
  }

  organization
    "IETF IPv6 over Low Power Wide-Area Networks (lpwan)
     Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/lpwan/about/>
     WG List:  <mailto:lp-wan@ietf.org>
     Editor:   Laurent Toutain
       <mailto:laurent.toutain@imt-atlantique.fr>
     Editor:   Juan Carlos Zuniga
       <mailto:j.c.zuniga@ieee.org>
     Editor:   Sergio Aguilar
       <mailto:sergio.aguilar.romero@upc.edu>";
  description
    "Copyright (c) 2023 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.
     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Revised BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).
     This version of this YANG module is part of RFC 9363
     (https://www.rfc-editor.org/info/rfc9363); see the RFC itself
     for full legal notices.
     ****************************************************************
     Generic data model for the Static Context Header Compression
     Rule for SCHC, based on RFCs 8724 and 8824.  Including
     compression, no-compression, and fragmentation Rules.";

  revision 2023-07-26 {
    description
      "Initial version for RFC 9441.";
    reference
      "RFC 9441 Static Context Header Compression (SCHC) Compound
                Acknowledgement (ACK)";
  }

  identity bitmap-format-base-type {
    description
      "Define how the bitmap is formed in ACK messages.";
  }

  identity bitmap-RFC8724 {
    base bitmap-format-base-type;
    description
      "Bitmap by default as defined in RFC 8724.";
```

```
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context Header
                Compression and Fragmentation";
  }

  identity bitmap-compound-ack {
    base bitmap-format-base-type;
    description
      "Compound ACK allows several bitmaps in an ACK message.";
  }

  typedef bitmap-format-type {
    type identityref {
      base bitmap-format-base-type;
    }
    description
      "Type of bitmap used in Rules.";
  }

  augment "/schc:schc/schc:rule/schc:nature/"
        + "schc:fragmentation/schc:mode/schc:ack-on-error" {
    leaf bitmap-format {
      when "derived-from-or-self(../schc:fragmentation-mode,
                        'schc:fragmentation-mode-ack-on-error')";
      type schc-compound-ack:bitmap-format-type;
      default "schc-compound-ack:bitmap-RFC8724";
      description
        "How the bitmaps are included in the SCHC ACK message.";
    }
    leaf last-bitmap-compression {
      when "derived-from-or-self(../schc:fragmentation-mode,
                        'schc:fragmentation-mode-ack-on-error')";
      type boolean;
      default "true";
      description
        "When true, the ultimate bitmap in the SCHC ACK message
         can be compressed.  Default behavior from RFC 8724.";
      reference
        "RFC 8724 SCHC: Generic Framework for Static Context Header
                  Compression and Fragmentation";
    }
    description
      "Augment the SCHC Rules to manage Compound ACK.";
  }
}

<CODE ENDS>
```

Figure 9: SCHC YANG Data Model - Compound ACK Extension

5.2. SCHC YANG Tree Extension

```
module: ietf-schc-compound-ack
  augment /schc:schc/schc:rule/schc:nature/schc:fragmentation/
        schc:mode/schc:ack-on-error:
    +--rw bitmap-format?        schc-compound-ack:bitmap-format-type
    +--rw last-bitmap-compression?   boolean
```

Figure 10: Tree Diagram - Compound ACK Extension

6. SCHC Compound ACK Parameters

This section lists the parameters related to the SCHC Compound ACK usage that need to be defined in the Profile. This list **MUST** be appended to the list of SCHC parameters under "Decision to use SCHC fragmentation mechanism or not. If yes, the document must describe:" as defined in Appendix D of [RFC8724].

- whether the SCHC Compound ACK message is used or not, and
- whether the compressed bitmap format in the last window of the SCHC Compound ACK message is used or not.

7. Security Considerations

This document specifies a message format extension for SCHC. Hence, the same security considerations defined in [RFC8724] and [RFC9363] apply.

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/schc:schc/schc:rule/schc:nature/schc:fragmentation/schc:mode/schc:ack-on-error:
: All the data nodes may be modified. The Rule contains sensitive information, such as the SCHC F/R mode configuration and usage and SCHC Compound ACK configuration. An attacker may try to modify other devices' Rules by changing the F/R mode or the usage of the SCHC

Compound ACK and may block communication or create extra ACKs. Therefore, a device must be allowed to modify only its own Rules on the remote SCHC instance. The identity of the requester must be validated. This can be done through certificates or access lists.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

/schc:schc/schc:rule/schc:nature/schc:fragmentation/schc:mode/schc:ack-on-error:
By reading this module, an attacker may learn the F/R mode used by the device, how the device manages the bitmap creation, the buffer sizes, and when the device will request an ACK.

8. IANA Considerations

This document registers one URI and one YANG data model.

8.1. URI Registration

IANA registered the following URI in the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-schc-compound-ack

Registrant Contact: The IESG.

XML: N/A; the requested URI is an XML namespace.

8.2. YANG Module Name Registration

IANA has registered the following YANG data model in the "YANG Module Names" registry [RFC6020].

name: ietf-schc-compound-ack

namespace: urn:ietf:params:xml:ns:yang:ietf-schc-compound-ack

prefix: schc-compound-ack

reference: RFC 9441

9. References

9.1. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[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>.

[RFC9363] Minaburo, A. and L. Toutain, "A YANG Data Model for Static Context Header Compression (SCHC)", RFC 9363, DOI 10.17487/RFC9363, March 2023, <https://www.rfc-editor.org/info/rfc9363>.

9.2. Informative 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>.

Acknowledgements

Carles Gomez has been funded in part by the Spanish Government through the TEC2016-79988-P grant and the PID2019-106808RA-I00 grant (funded by MCIN / AEI / 10.13039/501100011033) and by Secretaria d'Universitats i Recerca del Departament d'Empresa i Coneixement de la Generalitat de Catalunya through 2017 grant SGR 376 and 2021 grant SGR 00330.

Sergio Aguilar has been funded by the ERDF and the Spanish Government through project TEC2016-79988-P and project PID2019-106808RA-I00, AEI/FEDER, EU (funded by MCIN / AEI / 10.13039/501100011033).

Sandra Cespedes has been funded in part by the ANID Chile Project FONDECYT Regular 1201893 and Basal Project FB0008.

Diego Wistuba has been funded by the ANID Chile Project FONDECYT Regular 1201893.

The authors would like to thank Rafael Vidal, Julien Boite, Renaud Marty, Antonis Platis, Dominique Barthel, and Pascal Thubert for their very useful comments, reviews, and implementation design considerations.

Authors' Addresses

Juan Carlos Zúñiga
Cisco
Montreal QC
Canada
Email: juzuniga@cisco.com

Carles Gomez
Universitat Politècnica de Catalunya
C/Esteve Terradas, 7
08860 Castelldefels
Spain
Email: carles.gomez@upc.edu

Sergio Aguilar
Universitat Politècnica 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 Céspedes
Concordia University
1455 De Maisonneuve Blvd. W.
Montreal QC, H3G 1M8
Canada
Email: sandra.cespedes@concordia.ca

Diego Wistuba
NIC Labs, Universidad de Chile
Av. Almte. Blanco Encalada 1975
Santiago
Chile
Email: research@witu.cl