Workgroup: lpwan Working Group
Internet-Draft:
draft-ietf-lpwan-schc-compound-ack-06
Updates: 8724 (if approved)
Published: 1 August 2022
Intended Status: Standards Track
Expires: 2 February 2023
Authors: JC. Zuniga C. Gomez
Cisco Universitat Politecnica de Catalunya
S. Aguilar
Universitat Politecnica de Catalunya
L. Toutain S. Cespedes
IMT-Atlantique Concordia University
D. Wistuba
NIC Labs, Universidad de Chile

SCHC Compound ACK

Abstract

The present document describes an extension to the SCHC (Static Context Header Compression and fragmentation) protocol RFC8724. It defines a SCHC Compound 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 message format and procedure are generic, so they can be used, for instance, by any of the four LWPAN technologies defined in RFC8376, being Sigfox, LoRaWAN, NB-IoT and IEEE 802.15.4w.

Status of This Memo

Table of Contents

1. Introduction

The Generic Framework for Static Context Header Compression and Fragmentation (SCHC) 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 LWPAN defined in [RFC8376], being Sigfox, LoRaWAN, NB-IoT and IEEE 802.15.4w. 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.

The present document describes an extension to the SCHC protocol for frame fragmentation and loss recovery. It defines a SCHC Compound ACK format and procedure, which is 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 SCHC ACK message format so that it can contain several bitmaps, each bitmap being identified by its corresponding window number.

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 Compound SCHC 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.

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 in [RFC8724].

3. SCHC Compound ACK

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

The SCHC Compound ACK groups the window number (W) with its corresponding bitmap. Windows do not need to be contiguous. However, the window numbers and 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 W number MUST be placed in the SCHC ACK Header.

3.1. SCHC Compound ACK Message Format

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

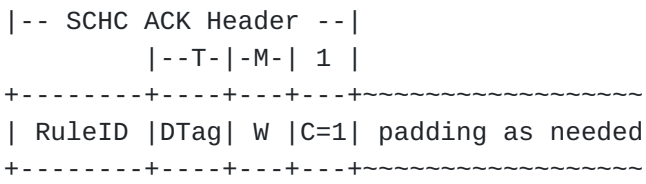

Figure 1: SCHC Success ACK message format, as defined in RFC8724

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 Figure 2 and Figure 3. If M or more padding bits would be needed after the last bitmap in the message to fill the last 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, and therefore they signal the end of the SCHC Compound ACK message.

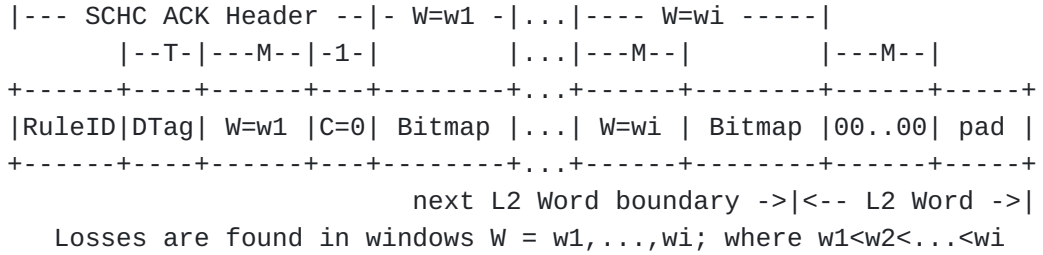

Figure 2: SCHC Compound ACK message format

Figure 3 shows the case when the required padding bits are strictly less than M bits. In this case, the length of the message learnt

from the underlying layer does not leave room for any extra window value, let alone any bitmap, thereby signaling the end of the SCHC Compound ACK message.

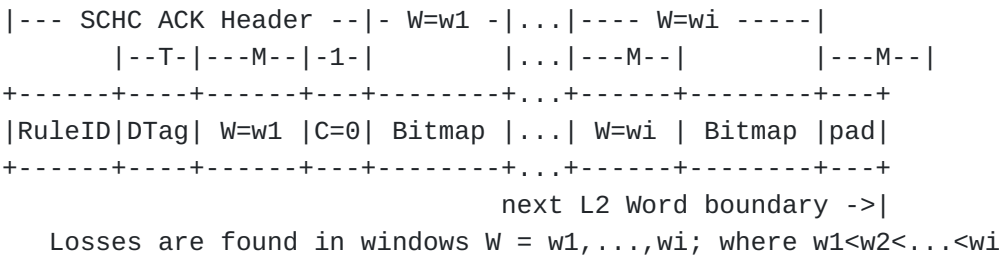

Figure 3: SCHC Compound ACK message format with less than M padding bits

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), and therefore intermediate bitmaps 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 SCHC technology-specific 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.

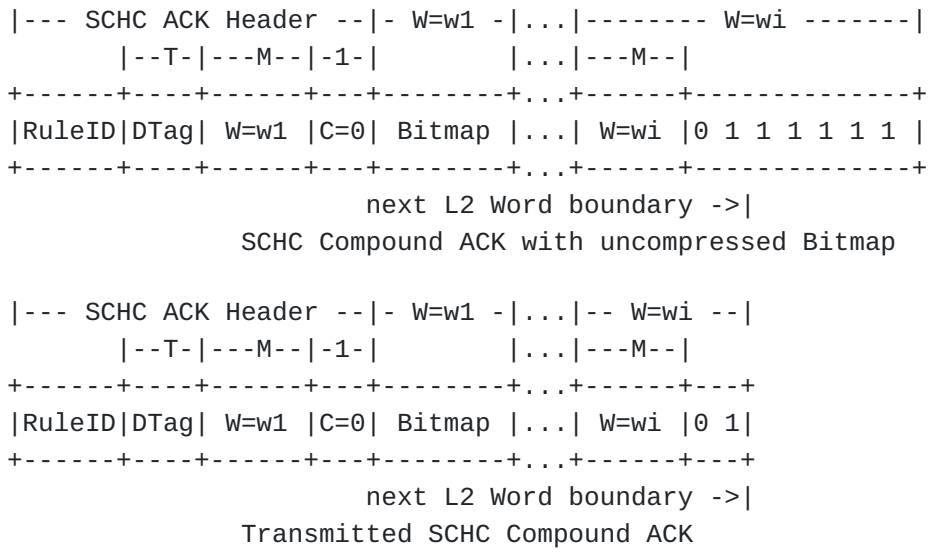

Figure 4: SCHC Compound ACK message format with compressed bitmap

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 tile 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.

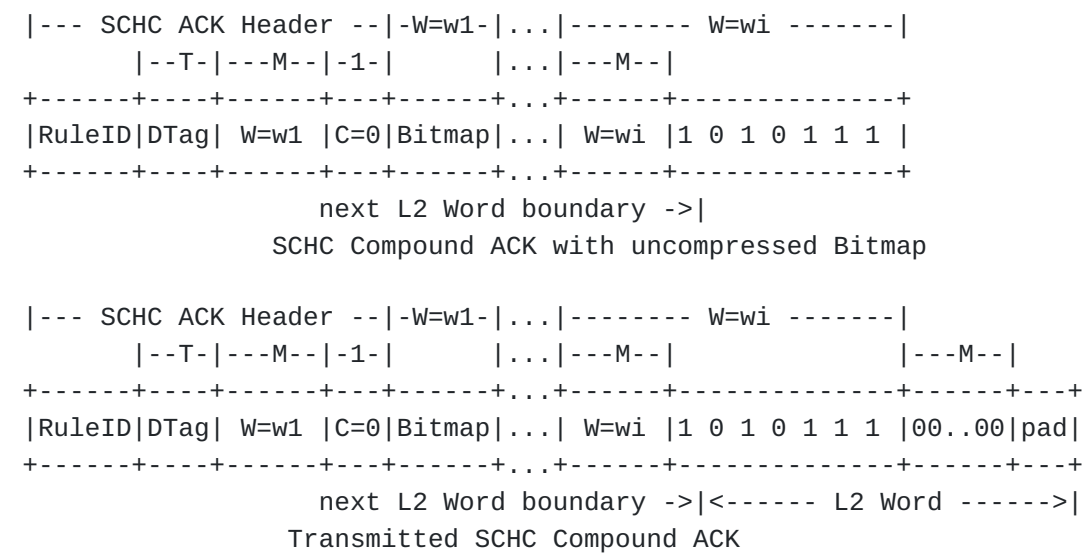

Figure 5: SCHC Compound ACK message format with compressed bitmap

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

Note: because it has a C bit reset to 0, the SCHC Compound ACK is distinguishable from the Receiver-Abort message [RFC8724], which has a C bit set to 1.

3.2. SCHC Compound ACK Behaviour

The SCHC ACK-on-Error behaviour is described in section 8.4.3 of [RFC8724]. The present document slightly modifies this behaviour, since 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, each bitmap being identified by its corresponding window number.

The SCHC ACK format, as presented in [RFC8724], 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].

Also, some flexibility is introduced with respect to [RFC8724], in that the receiver has the capability to respond to the All-0 with a SCHC Compound ACK or not, depending on certain parameters, like network conditions. Note that even though the protocol allows for

such flexibility, the actual decision criteria is not specified in this document.

The following sections describe the differences between the baseline SCHC specification and the present SCHC protocol extension specification. New text is between ** NEW TEXT **. Old text is -- OLD TEXT --. New text replaces old text.

8.4.3. ACK-on-Error Mode

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 --exactly-- one window of tiles ** or several windows of tiles, each one identified by its window number**.

See Figure 23 for an example.

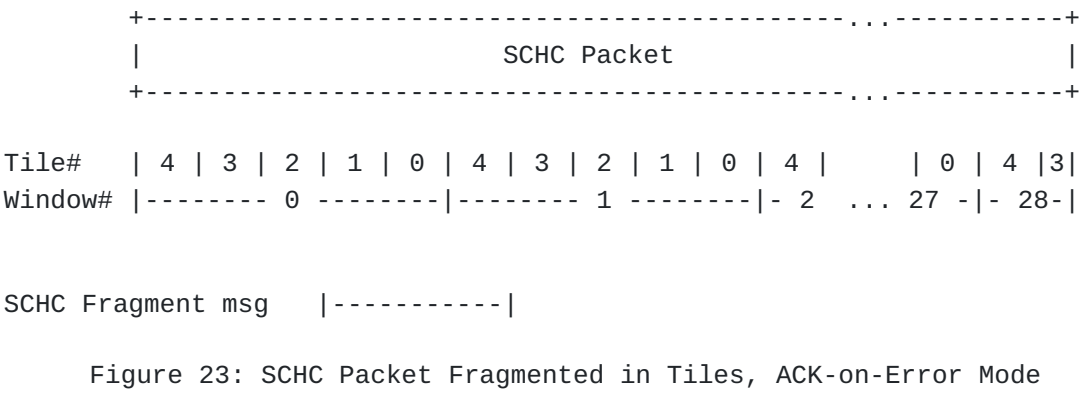

Figure 23: SCHC Packet Fragmented in Tiles, ACK-on-Error Mode

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, or
- too many retransmission attempts were 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 multiple 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 Section 8.4.3.1), and

*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.

***if the SCHC Compound ACK is used.**

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 Inactivity Timer, and

*one Attempts counter.

8.4.3.1. Sender Behavior

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 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 any 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 field 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:

oif the Profile mandates that the last tile be sent in an All-1 SCHC Fragment:

oif the SCHC **Compound** ACK shows no missing tile at the receiver, the sender:

o**MUST** send a SCHC Sender-Abort, and

o**MAY** exit with an error condition.

ootherwise:

othe fragment sender **MUST** send SCHC Fragment messages containing all the tiles **of all the windows** that are reported missing in the SCHC **Compound** ACK.

oif the last of these SCHC Fragment messages is not an All-1 SCHC Fragment, then the fragment sender --**MUST** in addition send after it a SCHC ACK REQ with the W field corresponding to the last window.-- **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 confirmation that all tiles have been correctly received.**

oin 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.

ootherwise:

oif the SCHC **Compound** ACK shows no missing tile at the receiver, the sender **MUST** send the All-1 SCHC Fragment

ootherwise:

othe fragment sender **MUST** send SCHC Fragment messages containing all the tiles that are reported missing in the SCHC **Compound** ACK.

othe 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.

8.4.3.2. Receiver Behavior

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

*the receiver **SHOULD** check if 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, 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:

othe size of the tiles is known a priori,

otiles are larger than an L2 Word, and

opadding 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 if network conditions are known to be conducive), 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.**--lowest-numbered such 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 numbered 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 for sending 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, or

*the Inactivity Timer has expired, or

*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 Examples

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 compound SCHC ACK is generated.

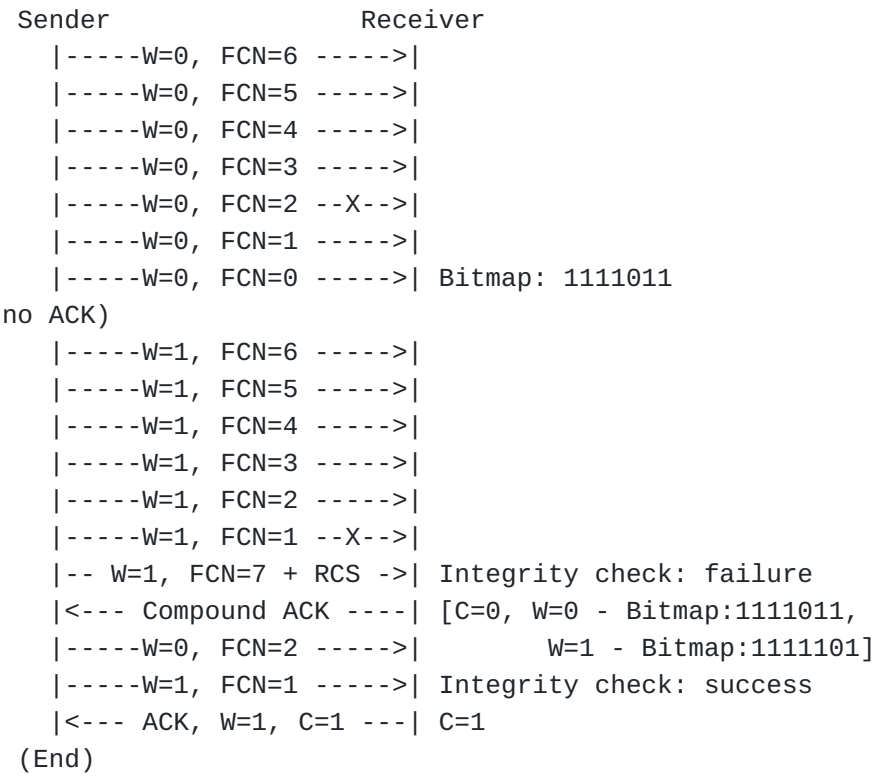

Figure 7: SCHC Compound ACK message sequence example

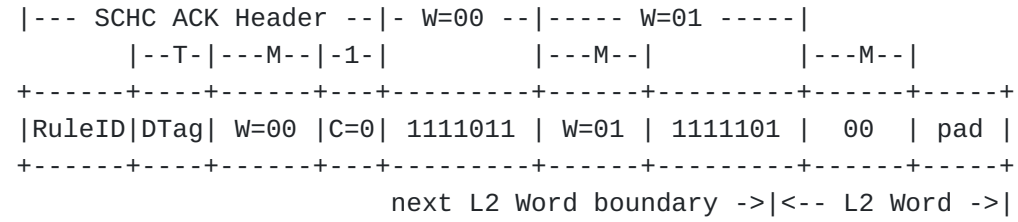

Figure 8: SCHC Compound ACK message format example: Losses are found in windows 00 and 01

5. SCHC Compound ACK YANG Data Model

The present document also extends the SCHC YANG data model defined in [I-D.ietf-lpwan-schc-yang-data-model] 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-compound-ack@2021-12-10.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>";
  description
     "
     Copyright (c) 2021 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 Simplified 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 XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.
     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 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.

     ****************************************************************

     This module extends the ietf-schc module to include the
     Compound ACK behavior for ACK-on-Error as defined in RFC YYYY.
     It introduces a new leaf for ACK-on-Error defining the format
     of the SCHC Compound ACK, adding the possibility to send
     several bitmaps in a single SCHC ACK message.";

  revision 2022-02-08 {
```

```
  description
    "Initial version for RFC YYYY ";
  reference
    "RFC YYYY: SCHC Compound 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 RFC8724.";
}

identity bitmap-compound-ack {
  base bitmap-format-base-type;
  description
    "Compound ACK.";
}

typedef bitmap-format-type {
  type identityref {
    base bitmap-format-base-type;
  }
  description
    "type used in rules";
}

augment "/schc:schc/schc:rule/schc:nature/schc:fragmentation/schc:mode/schc:ack-on-error" +
  leaf bitmap-format {
      when "derived-from(../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(../schc:fragmentation-mode, 'schc:fragmentation-mode-ack-on-error
      type boolean;
      default true;
      description
                      "when true ultimate bitmap in the SCHC ACK message can be compressed"
      }

  description
```

```
      "added to SCHC rules";
  }

}

<CODE ENDS>
```

Figure 9: SCHC YANG Data Model - Compound ACK extension

5.2. SCHC YANG Tree Extension

```
  augment /schc:schc/schc:rule/schc:nature/schc:fragmentation/schc:mode/
    +--rw bitmap-format?   schc-compound-ack:bitmap-format-type
```

Figure 10: SCHC YANG Tree - 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, in addition to the ones listed in Annex D of [RFC8724].

*Usage or not of the SCHC Compound ACK message.

*Usage or not of the compressed bitmap format in the last window of the SCHC Compound ACK message.

7. Security considerations

The current document specifies a message format extension for SCHC. Hence, the same Security Considerations defined in [RFC8724] apply.

8. IANA Considerations

This document has no IANA actions.

9. 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 2017 through grant SGR 376.

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.

10. References

10.1. Normative References

[I-D.ietf-lpwan-schc-yang-data-model] Minaburo, A. and L. Toutain, "Data Model for Static Context Header Compression (SCHC)", Work in Progress, Internet-Draft, draft-ietf-lpwan-schc-yang-data-model-14, February 2021, <http://www.ietf.org/internet-drafts/draft-ietf-lpwan-schc-yang-data-model-14.txt>.

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

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

[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>.

10.2. 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>.

Authors' Addresses

Juan Carlos Zúñiga
Cisco
Montreal QC
Canada

Email: juzuniga@cisco.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
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: wistuba@niclabs.cl