lpwan Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

A. Minaburo
Acklio
L. Toutain
IMT-Atlantique
C. Gomez
Universitat Politecnica de Catalunya
D. Barthel
Orange Labs
JC. Zuniga
SIGFOX

October 22, 2018

LPWAN Static Context Header Compression (SCHC) and fragmentation for IPv6 and UDP
draft-ietf-lpwan-ipv6-static-context-hc-17

Abstract

This document defines the Static Context Header Compression (SCHC) framework, which provides both header compression and fragmentation functionalities. SCHC has been designed for Low Power Wide Area Networks (LPWAN).

SCHC compression is based on a common static context stored in both the LPWAN device and the network side. This document defines a header compression mechanism and its application to compress IPv6/UDP headers.

This document also specifies a fragmentation and reassembly mechanism that is used to support the IPv6 MTU requirement over the LPWAN technologies. Fragmentation is needed for IPv6 datagrams that, after SCHC compression or when such compression was not possible, still exceed the layer-2 maximum payload size.

The SCHC header compression and fragmentation mechanisms are independent of the specific LPWAN technology over which they are used. This document defines generic functionalities and offers flexibility with regard to parameter settings and mechanism choices. Technology-specific and product-specific settings and choices are expected to be grouped into Profiles specified in other documents.

This Internet-Draft will expire on April 25, 2019.

Copyright Notice

Table of Contents

1. Introduction

This document defines the Static Context Header Compression (SCHC) framework, which provides both header compression and fragmentation functionalities. SCHC has been designed for Low Power Wide Area Networks (LPWAN).

Header compression is needed for efficient Internet connectivity to the node within an LPWAN network. Some LPWAN networks properties can be exploited to get an efficient header compression:

- The network topology is star-oriented, which means that all packets between the same source-destination pair follow the same path. For the needs of this document, the architecture can simply be described as Devices (Dev) exchanging information with LPWAN Application Servers (App) through a Network Gateway (NGW).

- Because devices embed built-in applications, the traffic flows to be compressed are known in advance. Indeed, new applications are less frequently installed in an LPWAN device, as they are in a computer or smartphone.

SCHC compression uses a context in which information about header fields is stored. This context is static: the values of the header fields do not change over time. This avoids complex resynchronization mechanisms. Indeed, downlink is often more restricted/expensive, perhaps completely unavailable [RFC8376]. A compression protocol that relies on feedback is not compatible with the characteristics of such LPWANs.

In most cases, a small context identifier is enough to represent the full IPv6/UDP headers. The SCHC header compression mechanism is independent of the specific LPWAN technology over which it is used.

LPWAN technologies impose some strict limitations on traffic. For instance, devices are sleeping most of the time and may receive data during short periods of time after transmission to preserve battery. LPWAN technologies are also characterized by a greatly reduced data unit and/or payload size (see [RFC8376]). However, some LPWAN technologies do not provide fragmentation functionality; to support the IPv6 MTU requirement of 1280 bytes [RFC8200], they require a fragmentation protocol at the adaptation layer below IPv6. Accordingly, this document defines an fragmentation/reassembly mechanism for LPWAN technologies to supports the IPv6 MTU. Its

implementation is optional. If not interested, the reader can safely skip its description.

This document defines generic functionality and offers flexibility with regard to parameters settings and mechanism choices. Technology-specific settings and product-specific and choices are expected to be grouped into Profiles specified in other documents.

2. Requirements Notation

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.

3. LPWAN Architecture

LPWAN technologies have similar network architectures but different terminologies. Using the terminology defined in [RFC8376], we can identify different types of entities in a typical LPWAN network, see Figure 1:

o Devices (Dev) are the end-devices or hosts (e.g. sensors, actuators, etc.). There can be a very high density of devices per radio gateway.

o The Radio Gateway (RGW), which is the end point of the constrained link.

o The Network Gateway (NGW) is the interconnection node between the Radio Gateway and the Internet.

o LPWAN-AAA Server, which controls the user authentication and the applications.

o Application Server (App)

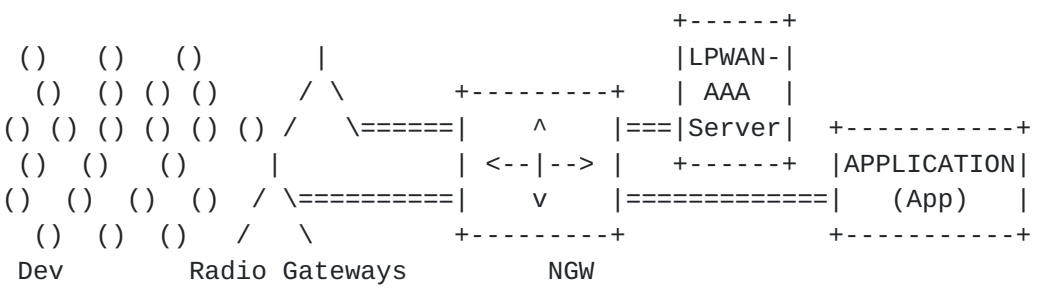


Figure 1: LPWAN Architecture

4. Terminology

This section defines the terminology and acronyms used in this document.

Note that the SCHC acronym is pronounced like "sheek" in English (or "chic" in French). Therefore, this document writes "a SCHC Packet" instead of "an SCHC Packet".

- App: LPWAN Application. An application sending/receiving IPv6 packets to/from the Device.

- AppIID: Application Interface Identifier. The IID that identifies the application server interface.

- Bi: Bidirectional. Characterises a Field Descriptor that applies to headers of packets travelling in either direction (Up and Dw, see this glossary).

- CDA: Compression/Decompression Action. Describes the reciprocal pair of actions that are performed at the compressor to compress a header field and at the decompressor to recover the original header field value.

- Compression Residue. The bits that need to be sent (beyond the Rule ID itself) after applying the SCHC compression over each header field.

- Context: A set of Rules used to compress/decompress headers.

- Dev: Device. A node connected to an LPWAN. A Dev SHOULD implement SCHC.

- DevIID: Device Interface Identifier. The IID that identifies the Dev interface.

- DI: Direction Indicator. This field tells which direction of packet travel (Up, Dw or Bi) a Rule applies to. This allows for assymmetric processing.

- Dw: Downlink direction for compression/decompression in both sides, from SCHC C/D in the network to SCHC C/D in the Dev.

- Field Description. A line in the Rule table.

- FID: Field Identifier. This is an index to describe the header fields in a Rule.

- FL: Field Length is the length of the packet header field. It is expressed in bits for header fields of fixed lengths or as a type (e.g. variable, token length, ...) for field lengths that are unknown at the time of Rule creation. The length of a header field is defined in the corresponding protocol specification (such as IPv6 or UDP).

- FP: Field Position is a value that is used to identify the position where each instance of a field appears in the header.

- IID: Interface Identifier. See the IPv6 addressing architecture [RFC7136]

- L2: Layer two. The immediate lower layer SCHC interfaces with. It is provided by an underlying LPWAN technology. It does not necessarily correspond to the OSI model definition of Layer 2.

- L2 Word: this is the minimum subdivision of payload data that the L2 will carry. In most L2 technologies, the L2 Word is an octet. In bit-oriented radio technologies, the L2 Word might be a single bit. The L2 Word size is assumed to be constant over time for each device.

- MO: Matching Operator. An operator used to match a value contained in a header field with a value contained in a Rule.

- Padding (P). Extra bits that may be appended by SCHC to a data unit that it passes to the underlying Layer 2 for transmission. SCHC itself operates on bits, not bytes, and does not have any alignment prerequisite. See Section 9.

- Profile: SCHC offers variations in the way it is operated, with a number of parameters listed in Appendix D. A Profile indicates a particular setting of all these parameters. Both ends of a SCHC session must be provisioned with the same Profile information and

with the same set of Rules before the session starts, so that there is no ambiguity in how they expect to communicate.

- Rule: A set of header field values.

- Rule ID: An identifier for a Rule. SCHC C/D on both sides share the same Rule ID for a given packet. A set of Rule IDs are used to support SCHC F/R functionality.

- SCHC C/D: Static Context Header Compression Compressor/ Decompressor. A mechanism used on both sides, at the Dev and at the network, to achieve Compression/Decompression of headers. SCHC C/D uses Rules to perform compression and decompression.

- SCHC Packet: A packet (e.g. an IPv6 packet) whose header has been compressed as per the header compression mechanism defined in this document. If the header compression process is unable to actually compress the packet header, the packet with the uncompressed header is still called a SCHC Packet (in this case, a Rule ID is used to indicate that the packet header has not been compressed). See Section 7 for more details.

- TV: Target value. A value contained in a Rule that will be matched with the value of a header field.

- Up: Uplink direction for compression/decompression in both sides, from the Dev SCHC C/D to the network SCHC C/D.

Additional terminology for the optional SCHC Fragmentation / Reassembly mechanism (SCHC F/R) is found in Section 8.2.

5. SCHC overview

SCHC can be characterized as an adaptation layer between IPv6 and the underlying LPWAN technology. SCHC comprises two sublayers (i.e. the Compression sublayer and the Fragmentation sublayer), as shown in Figure 2.

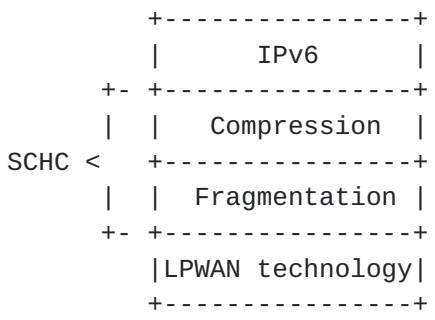
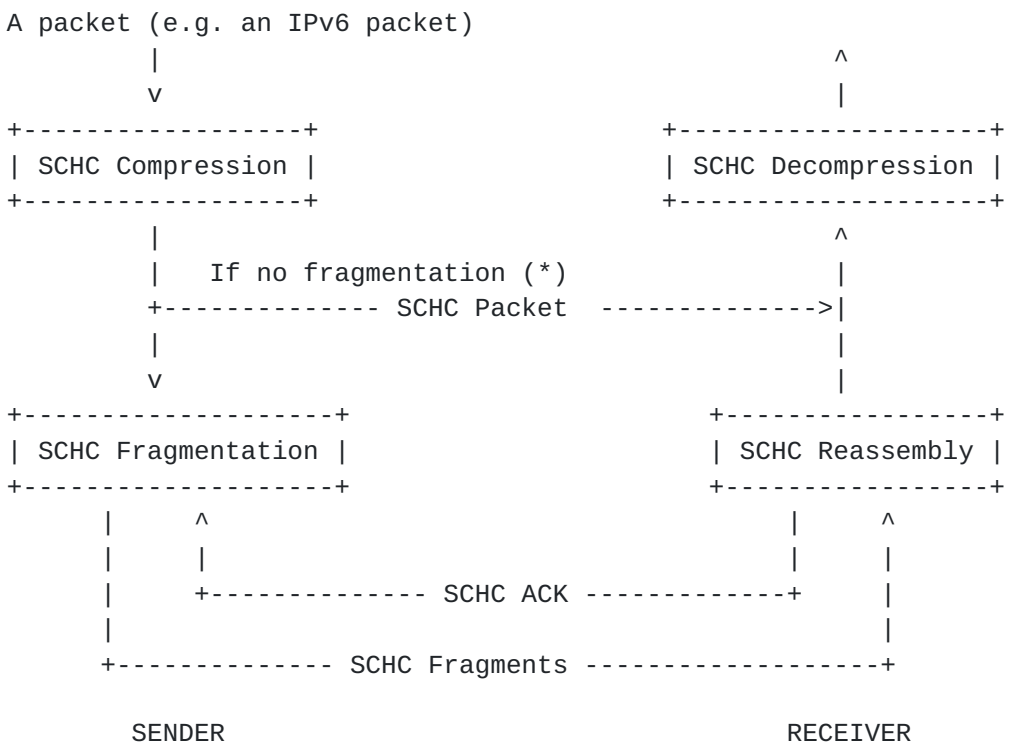


Figure 2: Protocol stack comprising IPv6, SCHC and an LPWAN technology

As per this document, when a packet (e.g. an IPv6 packet) needs to be transmitted, header compression is first applied to the packet. The resulting packet after header compression (whose header may or may not actually be smaller than that of the original packet) is called a SCHC Packet. If the SCHC Packet needs to be fragmented by the optional SCHC Fragmentation, fragmentation is then applied to the SCHC Packet. The SCHC Packet or the SCHC Fragments are then transmitted over the LPWAN. The reciprocal operations take place at the receiver. This process is illustrated in Figure 3.



*: the decision to use Fragmentation or not is left to each Profile.

Figure 3: SCHC operations at the SENDER and the RECEIVER

5.1. SCHC Packet format

The SCHC Packet is composed of the Compressed Header followed by the payload from the original packet (see Figure 4). The Compressed Header itself is composed of the Rule ID and a Compression Residue, which is the output of the compression actions of the Rule that was applied (see Section 7). The Compression Residue may be empty. Both the Rule ID and the Compression Residue potentially have a variable size, and generally are not a mutiple of bytes in size.

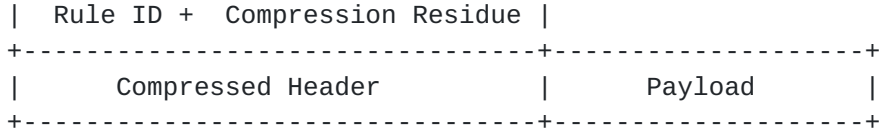


Figure 4: SCHC Packet

5.2. Functional mapping

Figure 5 below maps the functional elements of Figure 3 onto the LPWAN architecture elements of Figure 1.

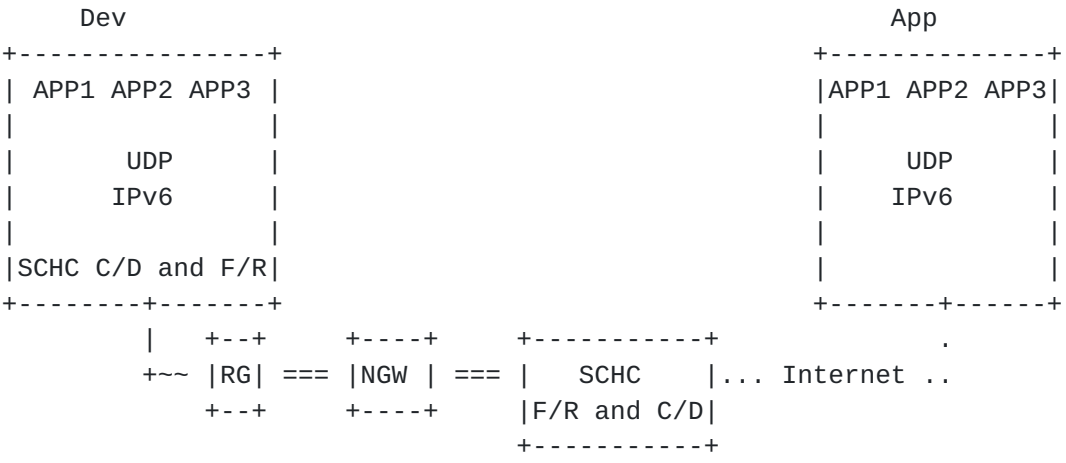


Figure 5: Architecture

SCHC C/D and SCHC F/R are located on both sides of the LPWAN transmission, i.e. on the Dev side and on the Network side.

The operation in the Uplink direction is as follows. The Device application uses IPv6 or IPv6/UDP protocols. Before sending the packets, the Dev compresses their headers using SCHC C/D and, if the SCHC Packet resulting from the compression needs to be fragmented by SCHC, SCHC F/R is performed (see Section 8). The resulting SCHC Fragments are sent to an LPWAN Radio Gateway (RG) which forwards them to a Network Gateway (NGW). The NGW sends the data to a SCHC F/R for re-assembly (if needed) and then to the SCHC C/D for decompression. After decompression, the packet can be sent over the Internet to one or several LPWAN Application Servers (App).

The SCHC F/R and C/D on the Network side can be located in the NGW, or somewhere else as long as a tunnel is established between them and the NGW. Note that, for some LPWAN technologies, it MAY be suitable to locate the SCHC F/R functionality nearer the NGW, in order to better deal with time constraints of such technologies.

The SCHC C/D and F/R on both sides MUST share the same set of Rules.

The SCHC C/D and F/R process is symmetrical, therefore the description of the Downlink direction is symmetrical to the one above.

6. Rule ID

Rule IDs are identifiers used to select the correct context either for Compression/Decompression or for Fragmentation/Reassembly.

The size of the Rule IDs is not specified in this document, as it is implementation-specific and can vary according to the LPWAN technology and the number of Rules, among others. It is defined in Profiles.

The Rule IDs are used:

- In the SCHC C/D context, to identify the Rule (i.e., the set of Field Descriptions) that is used to compress a packet header.

- At least one Rule ID MAY be allocated to tagging packets for which SCHC compression was not possible (no matching Rule was found).

- In SCHC F/R, to identify the specific modes and settings of SCHC Fragments being transmitted, and to identify the SCHC ACKs, including their modes and settings. Note that when F/R is used for both communication directions, at least two Rule ID values are therefore needed for F/R.

7. Compression/Decompression

Compression with SCHC is based on using context, i.e. a set of Rules to compress or decompress headers. SCHC avoids context synchronization, which consumes considerable bandwidth in other header compression mechanisms such as RoHC [RFC5795]. Since the content of packets is highly predictable in LPWAN networks, static contexts MAY be stored beforehand to omit transmitting some information over the air. The contexts MUST be stored at both ends, and they can be learned by a provisioning protocol or by out of band means, or they can be pre-provisioned. The way the contexts are provisioned is out of the scope of this document.

7.1. SCHC C/D Rules

The main idea of the SCHC compression scheme is to transmit the Rule ID to the other end instead of sending known field values. This Rule ID identifies a Rule that provides the closest match to the original packet values. Hence, when a value is known by both ends, it is only necessary to send the corresponding Rule ID over the LPWAN network. The manner by which Rules are generated is out of the scope of this document. The Rules MAY be changed at run-time but the mechanism is out of scope of this document.

The context contains a list of Rules (see Figure 6). Each Rule itself contains a list of Field Descriptions composed of a Field Identifier (FID), a Field Length (FL), a Field Position (FP), a Direction Indicator (DI), a Target Value (TV), a Matching Operator (MO) and a Compression/Decompression Action (CDA).

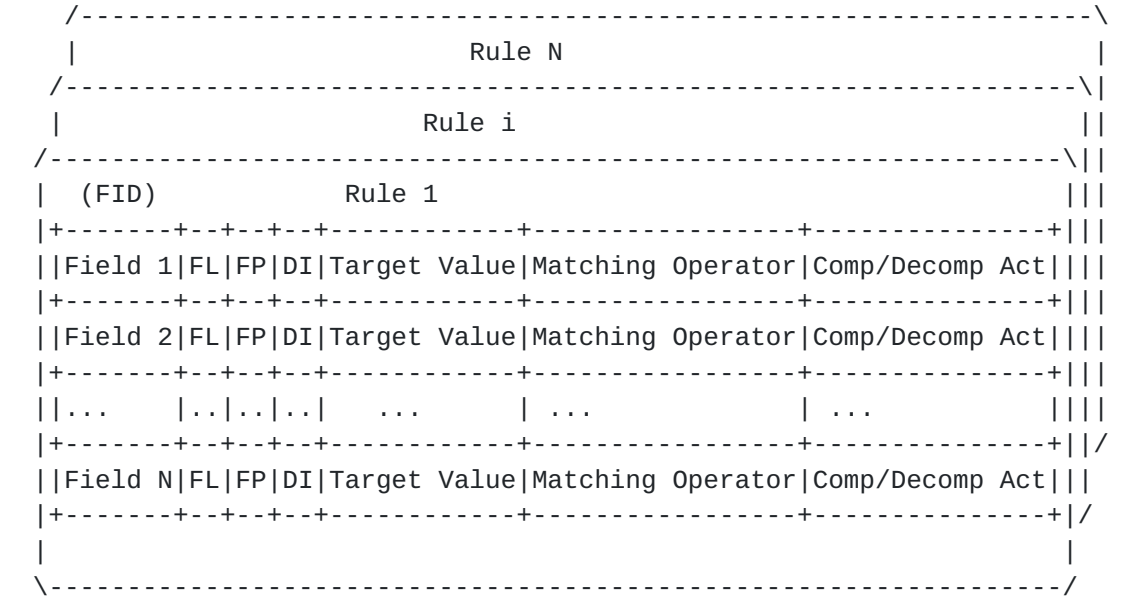


Figure 6: A Compression/Decompression Context

A Rule does not describe how to parse a packet header to find each field. This MUST be known from the compressor/decompressor. Rules only describe the compression/decompression behavior for each header field. In a Rule, the Field Descriptions are listed in the order in which the fields appear in the packet header.

A Rule also describes what is sent in the Compression Residue. The Compression Residue is assembled by concatenating the residues for each field, in the order the Field Descriptions appear in the Rule.

The Context describes the header fields and its values with the following entries:

- Field ID (FID) is a unique value to define the header field.

- Field Length (FL) represents the length of the field. It can be either a fixed value (in bits) if the length is known when the Rule is created or a type if the length is variable. The length of a header field is defined in the corresponding protocol specification. The type defines the process to compute the

length, its unit (bits, bytes,...) and the value to be sent before the Compression Residue.

- Field Position (FP): most often, a field only occurs once in a packet header. Some fields may occur multiple times in a header. FP indicates which occurrence this Field Description applies to. The default value is 1 (first occurence).

- A Direction Indicator (DI) indicates the packet direction(s) this Field Description applies to. Three values are possible:

  * UPLINK (Up): this Field Description is only applicable to packets sent by the Dev to the App,

  * DOWNLINK (Dw): this Field Description is only applicable to packets sent from the App to the Dev,

  * BIDIRECTIONAL (Bi): this Field Description is applicable to packets travelling both Up and Dw.

- Target Value (TV) is the value used to match against the packet header field. The Target Value can be of any type (integer, strings, etc.). It can be a single value or a more complex structure (array, list, etc.), such as a JSON or a CBOR structure.

- Matching Operator (MO) is the operator used to match the Field Value and the Target Value. The Matching Operator may require some parameters. MO is only used during the compression phase. The set of MOs defined in this document can be found in Section 7.4.

- Compression Decompression Action (CDA) describes the compression and decompression processes to be performed after the MO is applied. Some CDAs MAY require parameter values for their operation. CDAs are used in both the compression and the decompression functions. The set of CDAs defined in this document can be found in Section 7.5.

7.2. Rule ID for SCHC C/D

Rule IDs are sent by the compression function in one side and are received for the decompression function in the other side. In SCHC C/D, the Rule IDs are specific to a Dev. Hence, multiple Dev instances MAY use the same Rule ID to define different header compression contexts. To identify the correct Rule ID, the SCHC C/D needs to associate the Rule ID with the Dev identifier to find the appropriate Rule to be applied.

7.3. Packet processing

The compression/decompression process follows several steps:

- Compression Rule selection: The goal is to identify which Rule(s) will be used to compress the packet's headers. When performing decompression, on the network side the SCHC C/D needs to find the correct Rule based on the L2 address; in this way, it can use the DevIID and the Rule ID. On the Dev side, only the Rule ID is needed to identify the correct Rule since the Dev typically only holds Rules that apply to itself. The Rule will be selected by matching the Fields Descriptions to the packet header as described below. When the selection of a Rule is done, this Rule is used to compress the header. The detailed steps for compression Rule selection are the following:

  * The first step is to choose the Field Descriptions by their direction, using the Direction Indicator (DI). A Field Description that does not correspond to the appropriate DI will be ignored. If all the fields of the packet do not have a Field Description with the correct DI, the Rule is discarded and SCHC C/D proceeds to consider the next Rule.

  * When the DI has matched, then the next step is to identify the fields according to Field Position (FP). If FP does not correspond, the Rule is not used and the SCHC C/D proceeds to consider the next Rule.

  * Once the DI and the FP correspond to the header information, each packet field's value is then compared to the corresponding Target Value (TV) stored in the Rule for that specific field using the matching operator (MO).

    If all the fields in the packet's header satisfy all the matching operators (MO) of a Rule (i.e. all MO results are True), the fields of the header are then compressed according to the Compression/Decompression Actions (CDAs) and a compressed header (with possibly a Compression Residue) SHOULD be obtained. Otherwise, the next Rule is tested.

  * If no eligible compression Rule is found, then the header MUST be sent without compression, using a Rule ID dedicated to this purpose. Sending the header uncompressed but may require the use of the SCHC F/R process.

- Sending: The Rule ID is sent to the other end followed by the Compression Residue (which could be empty) or the uncompressed header, and directly followed by the payload. The Compression

Residue is the concatenation of the Compression Residues for each field according to the CDAs for that Rule. The way the Rule ID is sent depends on the Profile. For example, it can be either included in an L2 header or sent in the first byte of the L2 payload. (see Figure 4). This process will be specified in the Profile and is out of the scope of the present document. On LPWAN technologies that are byte-oriented, the compressed header concatenated with the original packet payload is padded to a multiple of 8 bits, if needed. See Section 9 for details.

- Decompression: When doing decompression, on the network side the SCHC C/D needs to find the correct Rule based on the L2 address and in this way, it can use the DevIID and the Rule ID. On the Dev side, only the Rule ID is needed to identify the correct Rule since the Dev only holds Rules that apply to itself.

  The receiver identifies the sender through its device-id or source identifier (e.g. MAC address, if it exists) and selects the Rule using the Rule ID. This Rule describes the compressed header format and associates the received Compression Residue to each of the header fields. For each field in the header, the receiver applies the CDA action associated to that field in order to reconstruct the original header field value. The CDA application order can be different from the order in which the fields are listed in the Rule. In particular, Compute-* MUST be applied after the application of the CDAs of all the fields it computes on.

7.4. Matching operators

Matching Operators (MOs) are functions used by both SCHC C/D endpoints involved in the header compression/decompression. They are not typed and can be applied to integer, string or any other data type. The result of the operation can either be True or False. MOs are defined as follows:

- equal: The match result is True if the field value in the packet matches the TV.

- ignore: No check is done between the field value in the packet and the TV in the Rule. The result of the matching is always true.

- MSB(x): A match is obtained if the most significant x bits of the packet header field value are equal to the TV in the Rule. The x parameter of the MSB MO indicates how many bits are involved in the comparison. If the FL is described as variable, the length must be a multiple of the unit. For example, x must be multiple of 8 if the unit of the variable length is in bytes.

- match-mapping: With match-mapping, the Target Value is a list of values. Each value of the list is identified by a short ID (or index). Compression is achieved by sending the index instead of the original header field value. This operator matches if the header field value is equal to one of the values in the target list.

7.5. Compression Decompression Actions (CDA)

The Compression Decompression Action (CDA) describes the actions taken during the compression of headers fields, and inversely, the action taken by the decompressor to restore the original value.

Action	Compression	Decompression
not-sent	elided	use value stored in context
value-sent	send	build from received value
mapping-sent	send index	value from index on a table
LSB	send LSB	TV, received value
compute-length	elided	compute length
compute-checksum	elided	compute UDP checksum
DevIID	elided	build IID from L2 Dev addr
AppIID	elided	build IID from L2 App addr

Figure 7: Compression and Decompression Actions

Figure 7 summarizes the basic actions that can be used to compress and decompress a field. The first column shows the action's name. The second and third columns show the reciprocal compression/ decompression behavior for each action.

Compression is done in the order that the Field Descriptions appear in a Rule. The result of each Compression/Decompression Action is appended to the accumulated Compression Residue in that same order. The receiver knows the size of each compressed field, which can be given by the Rule or MAY be sent with the compressed header.

7.5.1. processing variable-length fields

If the field is identified in the Field Description as being of variable size, then the size of the Compression Residue value (using the unit defined in the FL) MUST first be sent as follows:

- If the size is between 0 and 14, it is sent as a 4-bits unsigned integer.

- For values between 15 and 254, 0b1111 is transmitted and then the size is sent as an 8 bits unsigned integer.

- For larger values of the size, 0xfff is transmitted and then the next two bytes contain the size value as a 16 bits unsigned integer.

If a field is not present in the packet but exists in the Rule and its FL is specified as being variable, size 0 MUST be sent to denote its absence.

7.5.2. not-sent CDA

The not-sent action is generally used when the field value is specified in a Rule and therefore known by both the Compressor and the Decompressor. This action SHOULD be used with the "equal" MO. If MO is "ignore", there is a risk to have a decompressed field value different from the original field that was compressed.

The compressor does not send any Compression Residue for a field on which not-sent compression is applied.

The decompressor restores the field value with the Target Value stored in the matched Rule identified by the received Rule ID.

7.5.3. value-sent CDA

The value-sent action is generally used when the field value is not known by both the Compressor and the Decompressor. The value is sent as a residue in the compressed message header. Both Compressor and Decompressor MUST know the size of the field, either implicitly (the size is known by both sides) or by explicitly indicating the length in the Compression Residue, as defined in Section 7.5.1. This action is generally used with the "ignore" MO.

7.5.4. mapping-sent CDA

The mapping-sent action is used to send an index (the index into the Target Value list of values) instead of the original value. This action is used together with the "match-mapping" MO.

On the compressor side, the match-mapping Matching Operator searches the TV for a match with the header field value and the mapping-sent CDA appends the corresponding index to the Compression Residue to be

sent. On the decompressor side, the CDA uses the received index to restore the field value by looking up the list in the TV.

The number of bits sent is the minimal size for coding all the possible indices.

7.5.5. LSB CDA

The LSB action is used together with the "MSB(x)" MO to avoid sending the most significant part of the packet field if that part is already known by the receiving end. The number of bits sent is the original header field length minus the length specified in the MSB(x) MO.

The compressor sends the Least Significant Bits (e.g. LSB of the length field). The decompressor concatenates the x most significant bits of Target Value and the received residue.

If this action needs to be done on a variable length field, the size of the Compression Residue in bytes MUST be sent as described in Section 7.5.1.

7.5.6. DevIID, AppIID CDA

These actions are used to process respectively the Dev and the App Interface Identifiers (DevIID and AppIID) of the IPv6 addresses. AppIID CDA is less common since most current LPWAN technologies frames contain a single L2 address, which is the Dev's address.

The IID value MAY be computed from the Device ID present in the L2 header, or from some other stable identifier. The computation is specific to each Profile and MAY depend on the Device ID size.

In the downlink direction (Dw), at the compressor, the DevIID CDA may be used to generate the L2 addresses on the LPWAN, based on the packet's Destination Address.

7.5.7. Compute-*

Some fields may be elided during compression and reconstructed during decompression. This is the case for length and checksum, so:

- compute-length: computes the length assigned to this field. This CDA MAY be used to compute IPv6 length or UDP length.

- compute-checksum: computes a checksum from the information already received by the SCHC C/D. This field MAY be used to compute UDP checksum.

8. Fragmentation/Reassembly

8.1. Overview

In LPWAN technologies, the L2 MTU typically ranges from tens to hundreds of bytes. Some of these technologies do not have an internal fragmentation/reassembly mechanism.

The SCHC Fragmentation/Reassembly (SCHC F/R) functionality is offered as an option for such LPWAN technologies to cope with the IPv6 MTU requirement of 1280 bytes [RFC8200]. It is optional to implement. If it is not needed, its description can be safely ignored.

This specification includes several SCHC F/R modes, which allow for a range of reliability options such as optional SCHC Fragment retransmission. More modes may be defined in the future.

The same SCHC F/R mode MUST be used for all SCHC Fragments of the same fragmented SCHC Packet. This document does not make any decision with regard to which mode(s) will be used over a specific LPWAN technology. This will be defined in Profiles.

SCHC F/R uses the knowledge of the L2 Word size (see Section 4) to encode some messages. Therefore, SCHC MUST know the L2 Word size. SCHC F/R usually generates SCHC Fragments and SCHC ACKs that are multiples of L2 Words. The padding overhead is kept to the absolute minimum (see Section 9).

8.2. SCHC F/R Tools

This subsection describes the different tools that are used to enable the SCHC F/R functionality defined in this document. These tools include the SCHC F/R messages, tiles, windows, counters, timers and header fields.

The tools are described here in a generic manner. Their application to each SCHC F/R mode is found in Section 8.4.

8.2.1. Messages

The messages that can be used by SCHC F/R are the following:

- SCHC Fragment: A data unit that carries a piece of a SCHC Packet from the sender to the receiver.

- SCHC ACK: An acknowledgement for fragmentation, by the receiver to the sender. This message is used to report on the successful

reception of pieces of, or the whole of the fragmented SCHC Packet.

- SCHC ACK REQ: An explicit request for a SCHC ACK. By the sender to the receiver.

- SCHC Sender-Abort: A message by the sender telling the receiver that it has aborted the transmission of a fragmented SCHC Packet.

- SCHC Receiver-Abort: A message by the receiver to tell the sender to abort the transmission of a fragmented SCHC Packet.

8.2.2. Tiles, Windows, Bitmaps, Timers, Counters

8.2.2.1. Tiles

The SCHC Packet is fragmented into pieces, hereafter called tiles. The tiles MUST be contiguous.

See Figure 8 for an example.

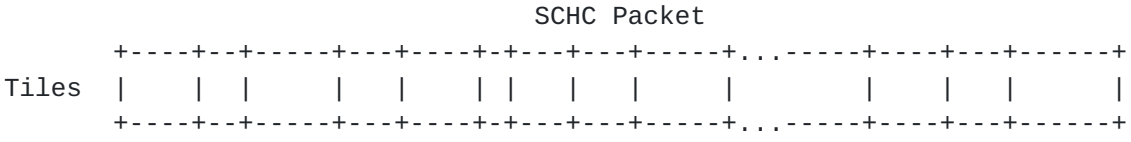


Figure 8: a SCHC Packet fragmented in tiles

Each SCHC Fragment message carries at least one tile in its Payload, if the Payload field is present.

8.2.2.2. Windows

Some SCHC F/R modes may handle successive tiles in groups, called windows.

If windows are used

- all the windows of a SCHC Packet, except the last one, MUST contain the same number of tiles. This number is WINDOW_SIZE.

- WINDOW_SIZE MUST be specified in a Profile.

- the windows are numbered.

- their numbers MUST increase from 0 upward, from the start of the SCHC Packet to its end.

- the last window MUST contain WINDOW_SIZE tiles or less.
- tiles are numbered within each window.
- the tile numbers MUST decrement from WINDOW_SIZE - 1 downward, looking from the start of the SCHC Packet toward its end.
- each tile of a SCHC Packet is therefore uniquely identified by a window number and a tile number within this window.

See Figure 9 for an example.

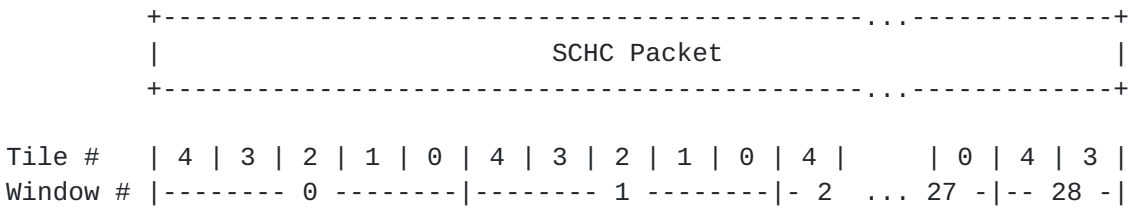


Figure 9: a SCHC Packet fragmented in tiles grouped in 28 windows, with WINDOW_SIZE = 5

When windows are used

- information on the correct reception of the tiles belonging to the same window MUST be grouped together.
- it is RECOMMENDED that this information is kept in Bitmaps.
- Bitmaps MAY be sent back to the sender in a SCHC ACK message.
- Each window has a Bitmap.

8.2.2.3. Bitmaps

Each bit in the Bitmap for a window corresponds to a tile in the window. Each Bitmap has therefore WINDOW_SIZE bits. The bit at the left-most position corresponds to the tile numbered WINDOW_SIZE - 1. Consecutive bits, going right, correspond to sequentially decreasing tile numbers. In Bitmaps for windows that are not the last one of a SCHC Packet, the bit at the right-most position corresponds to the tile numbered 0. In the Bitmap for the last window, the bit at the right-most position corresponds either to the tile numbered 0 or to a tile that is sent/received as "the last one of the SCHC Packet" without explicitely stating its number (see Section 8.3.1.2).

At the receiver

- a bit set to 1 in the Bitmap indicates that a tile associated with that bit position has been correctly received for that window.

- a bit set to 0 in the Bitmap indicates that no tile associated with that bit position has been correctly received for that window.

WINDOW_SIZE finely controls the size of the Bitmap sent in the SCHC ACK message, which may be critical to some LPWAN technologies.

8.2.2.4. Timers and counters

Some SCHC F/R modes can use the following timers and counters

- Inactivity Timer: this timer can be used to unlock a SCHC Fragment receiver that is not receiving a SCHC F/R message while it is expecting one.

- Retransmission Timer: this timer can be used by a SCHC Fragment sender to set a timeout on expecting a SCHC ACK.

- Attempts: this counter counts the requests for SCHC ACKs. MAX_ACK_REQUESTS is the threshold at which an exception is raised.

8.2.3. Integrity Checking

The reassembled SCHC Packet is checked for integrity at the receive end. Integrity checking is performed by computing a MIC at the sender side and transmitting it to the receiver for comparison with the locally computed MIC.

The MIC supports UDP checksum elision by SCHC C/D (see Section 10.11).

The CRC32 polynomial 0xEDB88320 (i.e. the reverse representation of the polynomial used e.g. in the Ethernet standard [RFC3385]) is RECOMMENDED as the default algorithm for computing the MIC. Nevertheless, other MIC lengths or other algorithms MAY be required by the Profile.

Note that the concatenation of the complete SCHC Packet and the potential padding bits of the last SCHC Fragment does not generally constitute an integer number of bytes. For implementers to be able to use byte-oriented CRC libraries, it is RECOMMENDED that the concatenation of the complete SCHC Packet and the last fragment potential padding bits be zero-extended to the next byte boundary and that the MIC be computed on that byte array. A Profile MAY specify another behaviour.

8.2.4. Header Fields

The SCHC F/R messages use the following fields (see the related formats in Section 8.3):

- Rule ID: this field is present in all the SCHC F/R messages. It is used to identify
  - that a SCHC F/R message is being carried, as opposed to an unfragmented SCHC Packet,
  - which SCHC F/R mode is used
  - and among this mode
    - if windows are used and what the value of WINDOW_SIZE is,
    - what other optional fields are present and what the field sizes are.

  Therefore, the Rule ID allows SCHC F/R interleaving non-fragmented SCHC Packets and SCHC Fragments that carry other SCHC Packets, or interleaving SCHC Fragments that use different SCHC F/R modes or different parameters.

- Datagram Tag (DTag). The DTag field is optional. Its presence and size (called T, in bits) is defined by each Profile for each Rule ID.

  When there is no DTag, there can be only one fragmented SCHC Packet in transit for a given Rule ID.

  If present, DTag
  - MUST be set to the same value for all the SCHC F/R messages related to the same fragmented SCHC Packet,
  - MUST be set to different values for SCHC F/R messages related to different SCHC Packets that are being fragmented under the same Rule ID and that may overlap during the fragmented transmission.

  A sequence counter that is incremented for each new fragmented SCHC Packet, counting from 0 to up to (2^T)-1 and wrapping back to 0 is RECOMMENDED for maximum traceability and replay avoidance.

- W: The W field is optional. It is only present if windows are used. Its presence and size (called M, in bits) is defined by each SCHC F/R mode and each Profile for each Rule ID.

  This field carries information pertaining to the window a SCHC F/R message relates to. If present, W MUST carry the same value for all the SCHC F/R messages related to the same window. Depending on the mode and Profile, W may carry the full window number, or just the least significant bit or any other partial representation of the window number.

- Fragment Compressed Number (FCN). The FCN field is present in the SCHC Fragment Header. Its size (called N, in bits) is defined by each Profile for each Rule ID.

  This field conveys information about the progress in the sequence of tiles being transmitted by SCHC Fragment messages. For example, it can contain a partial, efficient representation of a larger-sized tile number. The description of the exact use of the FCN field is left to each SCHC F/R mode. However, two values are reserved for special purposes. They help control the SCHC F/R process:

  * The FCN value with all the bits equal to 1 (called All-1) signals the very last tile of a SCHC Packet. By extension, if windows are used, the last window of a packet is called the All-1 window.

  * If windows are used, the FCN value with all the bits equal to 0 (called All-0) signals the last tile of a window that is not the last one of the SCHC packet. By extension, such a window is called an All-0 window.

  The highest value of FCN (an unsigned integer) is called MAX_WIND_FCN. Since All-1 is reserved, MAX_WIND_FCN MUST be stricly less that (2^N)-1.

- Message Integrity Check (MIC). This field only appears in the All-1 SCHC Fragments. Its size (called T, in bits) is defined by each Profile for each Rule ID.

  See Section 8.2.3 for the MIC default size, default polynomials and details on its computation.

- C (integrity Check): C is a 1-bit field. This field is used in the SCHC ACK message to report on the reassembled SCHC Packet integrity check (see Section 8.2.3).

A value of 1 tells that the integrity check was performed and is successful. A value of 0 tells that the integrity check was not performed, or that is was a failure.

- Compressed Bitmap. The Compressed Bitmap is used together with windows and Bitmaps (see Section 8.2.2.3). Its presence and size is defined for each F/R mode for each Rule ID.

  This field appears in the SCHC ACK message to report on the receiver Bitmap (see Section 8.3.2.1).

8.3. SCHC F/R Message Formats

This section defines the SCHC Fragment formats, the SCHC ACK format, the SCHC ACK REQ format and the SCHC Abort formats.

8.3.1. SCHC Fragment format

A SCHC Fragment conforms to the general format shown in Figure 10. It comprises a SCHC Fragment Header and a SCHC Fragment Payload. The SCHC Fragment Payload carries one or several tile(s).

Figure 10: SCHC Fragment general format. Presence of a padding field is optional

8.3.1.1. Regular SCHC Fragment

The Regular SCHC Fragment format is shown in Figure 11. Regular SCHC Fragments are generally used to carry tiles that are not the last one of a SCHC Packet. The DTag field and the W field are optional.

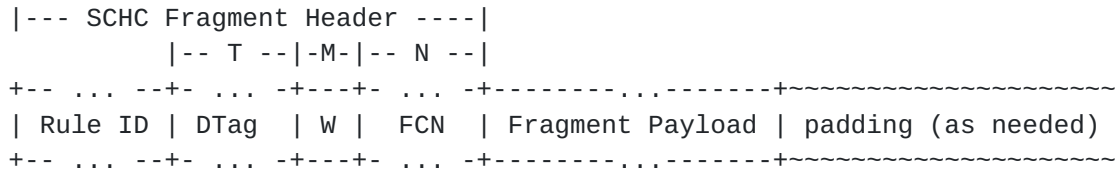


Figure 11: Detailed Header Format for Regular SCHC Fragments

The FCN field MUST NOT contain all bits set to 1.

If the size of the SCHC Fragment Payload does not nicely complement the SCHC Header size in a way that would make the SCHC Fragment a multiple of the L2 Word, then padding bits MUST be added.

The Fragment Payload of a SCHC Fragment with FCN == 0 (called an All-0 SCHC Fragment) MUST be at least the size of an L2 Word. The rationale is that, even in the presence of padding, an All-0 SCHC Fragment needs to be distinguishable from the SCHC ACK REQ message, which has the same header but has no payload (see Section 8.3.3).

8.3.1.2. All-1 SCHC Fragment

The All-1 SCHC Fragment format is shown in Figure 12. The All-1 SCHC Fragment is generally used to carry the very last tile of a SCHC Packet and a MIC, or a MIC only. The DTag field, the W field and the Payload are optional.

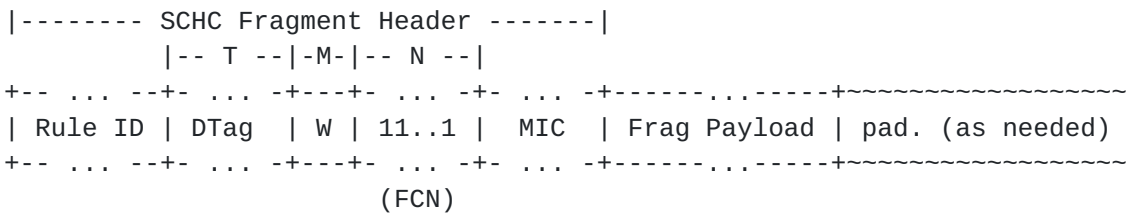


Figure 12: Detailed format for the All-1 SCHC Fragment

If the size of the SCHC Fragment Payload does not nicely complement the SCHC Header size in a way that would make the SCHC Fragment a multiple of the L2 Word, then padding bits MUST be added.

The All-1 SCHC Fragment message MUST be distinguishable by size from a SCHC Sender-Abort message (see Section 8.3.4.1) that has the same T, M and N values. This is trivially achieved by having the MIC larger than an L2 Word, or by having the Payload larger than an L2 Word. This is also naturally achieved if the SCHC Sender-Abort Header is a multiple of L2 Words.

8.3.2. SCHC ACK format

The SCHC ACK message MUST obey the format shown in Figure 13. The DTag field, the W field and the Compressed Bitmap field are optional. The Compressed Bitmap field can only be present in SCHC F/R modes that use windows.

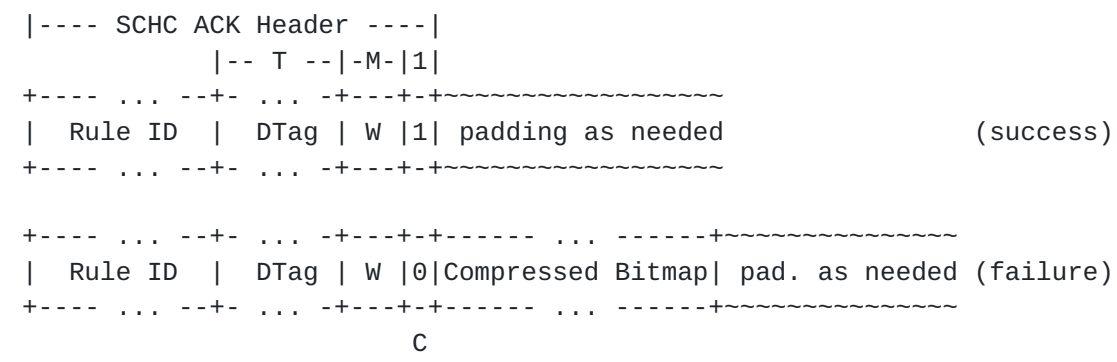


Figure 13: Format of the SCHC ACK message

The SCHC ACK Header contains a C bit (see Section 8.2.4).

If the C bit is set to 1 (integrity check successful), no Bitmap is carried and padding bits MUST be appended as needed to fill up the last L2 Word.

If the C bit is set to 0 (integrity check not performed or failed) and if windows are used,

- a representation of the Bitmap for the window referred to by the W field MUST follow the C bit

- padding bits MUST be appended as needed to fill up the last L2 Word

If the C bit is 1 or windows are not used, the C bit MUST be followed by padding bits as needed to fill up the last L2 Word.

See Section 8.2.2.3 for a description of the Bitmap.

The representation of the Bitmap that is transmitted MUST be the compressed version specified in Section 8.3.2.1, in order to reduce the SCHC ACK message size.

8.3.2.1. Bitmap Compression

For transmission, the Compressed Bitmap in the SCHC ACK message is defined by the following algorithm (see Figure 14 for a follow-along example):

- Build a temporary SCHC ACK message that contains the Header followed by the original Bitmap.

- Positioning scissors at the end of the Bitmap, after its last bit.

- While the bit on the left of the scissors is 1 and belongs to the Bitmap, keep moving left, then stop. When this is done,
- While the scissors are not on an L2 Word boundary of the SCHC ACK message and there is a Bitmap bit on the right of the scissors, keep moving right, then stop.
- At this point, cut and drop off any bits to the right of the scissors

When one or more bits have effectively been dropped off as a result of the above algorithm, the SCHC ACK message is a multiple of L2 Words, no padding bits will be appended.

Because the SCHC Fragment sender knows the size of the original Bitmap, it can reconstruct the original Bitmap from the Compressed Bitmap received in the SCH ACK message.

Figure 14 shows an example where L2 Words are actually bytes and where the original Bitmap contains 17 bits, the last 15 of which are all set to 1.

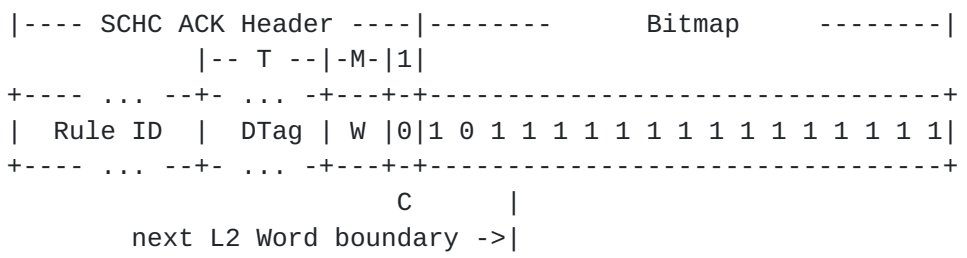


Figure 14: Tentative SCHC ACK message with Bitmap before compression

Figure 15 shows that the last 14 bits are not sent.

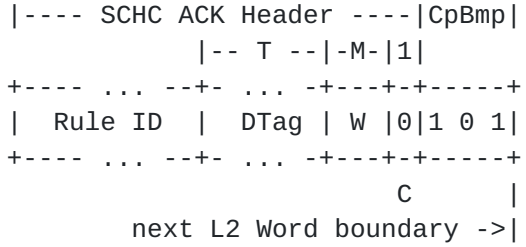


Figure 15: Actual SCHC ACK message with Compressed Bitmap, no padding

Figure 16 shows an example of a SCHC ACK with tile numbers ranging from 6 down to 0, where the Bitmap indicates that the second and the fourth tile of the window have not been correctly received.

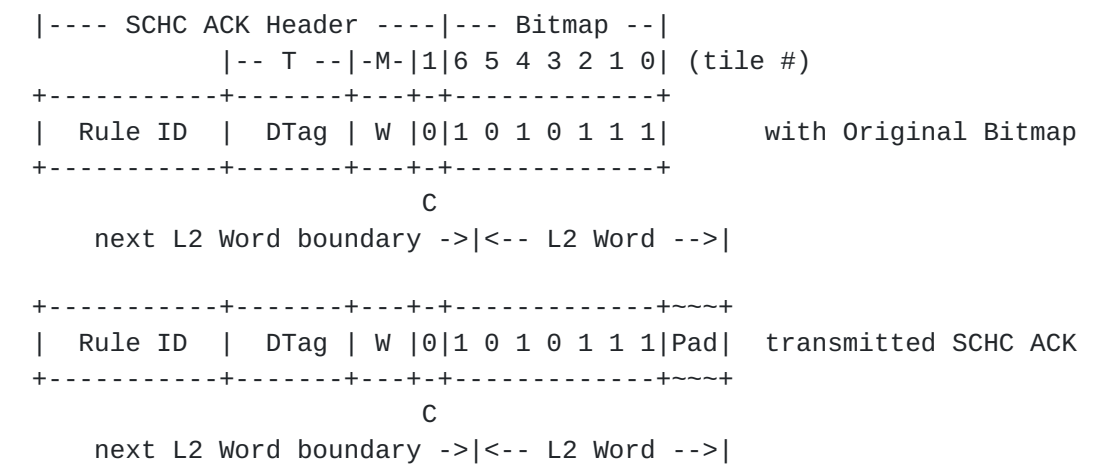


Figure 16: Example of a SCHC ACK message, missing tiles, with padding

Figure 17 shows an example of a SCHC ACK with FCN ranging from 6 down to 0, where integrity check has not been performed or has failed and the Bitmap indicates that there is no missing tile in that window.

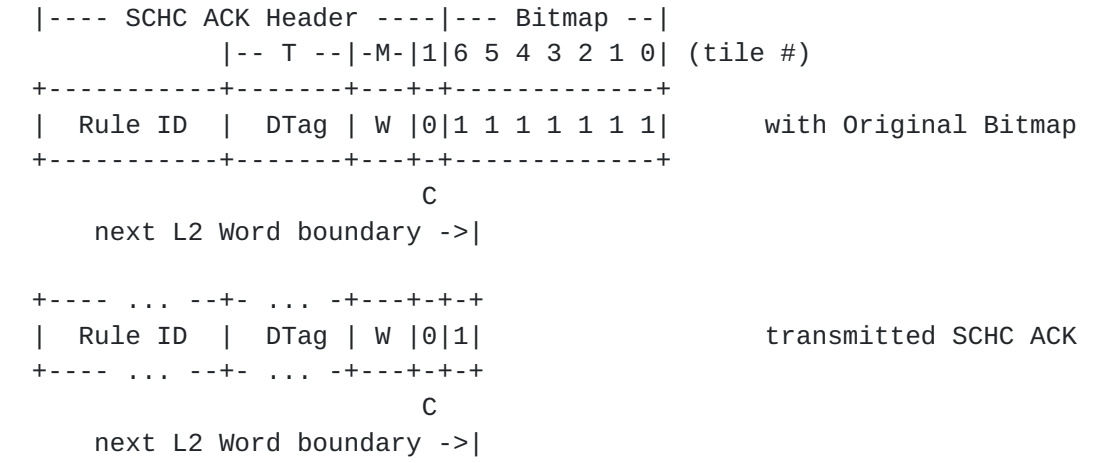


Figure 17: Example of a SCHC ACK message, no missing tile, no padding

8.3.3. SCHC ACK REQ format

The SCHC ACK REQ is used by a sender to explicitely request a SCHC ACK from the receiver. Its format is described in Figure 18. The DTag field and the W field are optional.

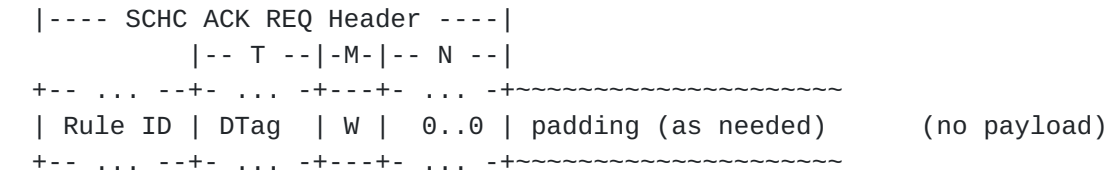


Figure 18: SCHC ACK REQ detailed format

The size of the SCHC ACK REQ header is generally not a multiple of the L2 Word size. Therefore, a SCHC ACK REQ generally needs padding bits.

Note that the SCHC ACK REQ has the same header as an All-0 SCHC Fragment (see Section 8.3.1.1) but it doesn't have a payload. A receiver can distinguish the former form the latter by the message length, even in the presence of padding. This is possible because

- the padding bits are always stricly less than an L2 Word.
- the size of an All-0 SCHC Fragment Payload is at least the size of an L2 Word,

8.3.4. SCHC Abort formats

8.3.4.1. SCHC Sender-Abort

When a SCHC Fragment sender needs to abort an on-going fragmented SCHC Packet transmission, it sends a SCHC Sender-Abort message to the SCHC Fragment receiver.

The SCHC Sender-Abort format is described in Figure 19. The DTag field and the W field are optional.

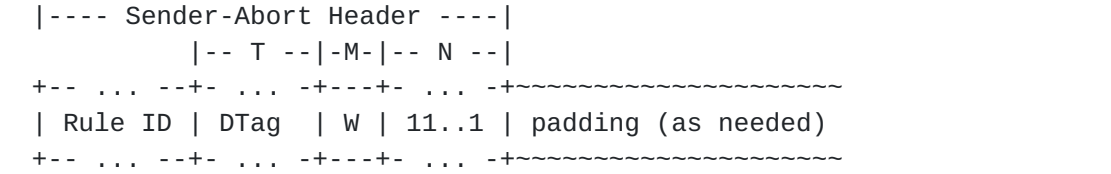


Figure 19: SCHC Sender-Abort format

If the W field is present,

- the fragment sender MUST set it to all 1's. Other values are RESERVED.
- the fragment receiver MUST check its value. If the value is different from all 1's, the message MUST be ignored.

The size of the SCHC Sender-Abort header is generally not a multiple of the L2 Word size. Therefore, a SCHC Sender-Abort generally needs padding bits.

Note that the SCHC Sender-Abort has the same header as an All-1 SCHC Fragment (see Section 8.3.1.2), but that it does not include a MIC nor a payload. The receiver distinguishes the former from the latter by the message length, even in the presence of padding. This is possible through different combinations

- the size of the Sender-Abort Header may be made such that it is not padded
- or the total size of the MIC and the Payload of an All-1 SCHC Fragment is at least the size of an L2 Word
- or through other alignment and size combinations

The SCHC Sender-Abort MUST NOT be acknowledged.

8.3.4.2. SCHC Receiver-Abort

When a SCHC Fragment receiver needs to abort an on-going fragmented SCHC Packet transmission, it transmits a SCHC Receiver-Abort message to the SCHC Fragment sender.

The SCHC Receiver-Abort format is described in Figure 20. The DTag field and the W field are optional.

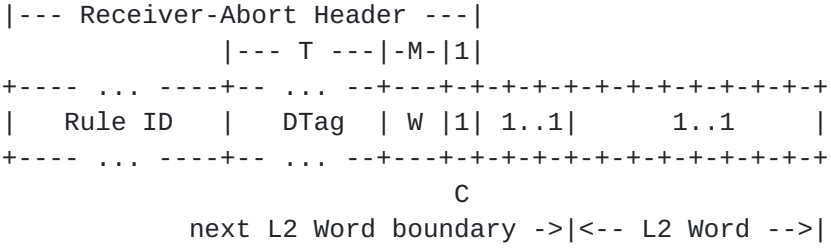


Figure 20: SCHC Receiver-Abort format

If the W field is present,

- the fragment receiver MUST set it to all 1's. Other values are RESERVED.
- the fragment sender MUST check its value. If the value is different from all 1's, the message MUST be ignored.

Note that the SCHC Receiver-Abort has the same header as a SCHC ACK message. The bits that follow the SCHC Receiver-Abort Header MUST be as follows

- if the Header does not end at an L2 Word boundary, append bits set to 1 as needed to reach the next L2 Word boundary
- append exactly one more L2 Word with bits all set to 1's

Such a bit pattern never occurs in a regular SCHC ACK. This is how the fragment sender recognizes a SCHC Receiver-Abort.

A SCHC Receiver-Abort is aligned to L2 Words, by design. Therefore, padding MUST NOT be appended.

The SCHC Receiver-Abort MUST NOT be acknowledged.

8.4. SCHC F/R modes

This specification includes several SCHC F/R modes, which allow for

- a range of reliability options, such as optional SCHC Fragment retransmission
- support of different LPWAN characteristics, such as variable MTU.

More modes may be defined in the future.

8.4.1. No-ACK mode

The No-ACK mode has been designed under the assumption that data unit out-of-sequence delivery does not occur between the entity performing fragmentation and the entity performing reassembly. This mode supports LPWAN technologies that have a variable MTU.

In No-ACK mode, there is no feedback communication from the fragment receiver to the fragment sender. The sender just transmits all the SCHC Fragments blindly.

Padding is kept to a minimum: only the last SCHC Fragment is padded as needed.

The tile sizes are not required to be uniform. Windows are not used. The Retransmission Timer is not used. The Attempts counter is not used.

Each Profile MUST specify which Rule ID value(s) is (are) allocated to this mode. For brevity, the rest of Section 8.4.1 only refers to Rule ID values that are allocated to this mode.

The W field MUST NOT be present in the SCHC F/R messages. SCHC ACK MUST NOT be sent. SCHC ACK REQ MUST NOT be sent. SCHC Sender-Abort MAY be sent. SCHC Receiver-Abort MUST NOT be sent.

The value of N (size of the FCN field) is RECOMMENDED to be 1.

Each Profile, for each Rule ID value, MUST define

- the presence or absence of the DTag field in the SCHC F/R messages, as well as its size if it is present,
- the size and algorithm for the MIC field in the SCHC F/R messages, if different from the default,
- the expiration time of the Inactivity Timer

Each Profile, for each Rule ID value, MAY define

- a value of N different from the recommend one,
- what values will be sent in the FCN field, for values different from the All-1 value.

The receiver, for each pair of Rule ID and optional DTag values, MUST maintain

- one Inactivity Timer

8.4.1.1. Sender behaviour

At the beginning of the fragmentation of a new SCHC Packet, the fragment sender MUST select a Rule ID and optional DTag value pair for this SCHC Packet. For brevity, the rest of Section 8.4.1 only refers to SCHC F/R messages bearing the Rule ID and optional DTag values hereby selected.

Each SCHC Fragment MUST contain exactly one tile in its Payload. The tile MUST be at least the size of an L2 Word. The sender MUST transmit the SCHC Fragments messages in the order that the tiles appear in the SCHC Packet. Except for the last tile of a SCHC Packet, each tile MUST be of a size that complements the SCHC Fragment Header so that the SCHC Fragment is a multiple of L2 Words without the need for padding bits. Except for the last one, the SCHC Fragments MUST use the Regular SCHC Fragment format specified in

Section 8.3.1.1. The last SCHC Fragment MUST use the All-1 format specified in Section 8.3.1.2.

The MIC MUST be computed on the reassembled SCHC Packet concatenated with the padding bits of the last SCHC Fragment. The rationale is that the SCHC Reassembler has no way of knowing where the payload of the last SCHC Fragment ends. Indeed, this requires decompressing the SCHC Packet, which is out of the scope of the SCHC Reassembler.

The sender MAY transmit a SCHC Sender-Abort.

Figure 35 shows an example of a corresponding state machine.

8.4.1.2. Receiver behaviour

On receiving Regular SCHC Fragments,

- the receiver MUST reset the Inactivity Timer,
- the receiver assembles the payloads of the SCHC Fragments

On receiving an All-1 SCHC Fragment,

- the receiver MUST append the All-1 SCHC Fragment Payload and the padding bits to the previously received SCHC Fragment Payloads for this SCHC Packet
- if an integrity checking is specified in the Profile,
  - the receiver MUST perform the integrity check
  - if integrity checking fails, the receiver MUST drop the reassembled SCHC Packet and it MUST release all resources associated with this Rule ID and optional DTag values.
- the reassembly operation concludes.

On expiration of the Inactivity Timer, the receiver MUST drop the SCHC Packet being reassembled and it MUST release all resources associated with this Rule ID and optional DTag values.

On receiving a SCHC Sender-Abort, the receiver MAY release all resources associated with this Rule ID and optional DTag values.

The MIC computed at the receiver MUST be computed over the reassembled SCHC Packet and over the padding bits that were received in the SCHC Fragment carrying the last tile.

Figure 36 shows an example of a corresponding state machine.

8.4.2. ACK-Always

The ACK-Always mode has been designed under the following assumptions

- Data unit out-of-sequence delivery does not occur between the entity performing fragmentation and the entity performing reassembly

- The L2 MTU value does not change while a fragmented SCHC Packet is being transmitted.

In ACK-Always mode, windows are used. An acknowledgement, positive or negative, is fed by the fragment receiver back to the fragment sender at the end of the transmission of each window of SCHC Fragments.

The tiles are not required to be of uniform size. Padding is kept to a minimum: only the last SCHC Fragment is padded as needed.

In a nutshell, the algorithm is the following: after a first blind transmission of all the tiles of a window, the fragment sender iterates retransmitting the tiles that are reported missing until the fragment receiver reports that all the tiles belonging to the window have been correctly received, or until too many attempts were made. The fragment sender only advances to the next window of tiles when it has ascertained that all the tiles belonging to the current window have been fully and correctly received. This results in a lock-step behaviour between the sender and the receiver, at the window granularity.

Each Profile MUST specify which Rule ID value(s) is (are) allocated to this mode. For brevity, the rest of Section 8.4.1 only refers to Rule ID values that are allocated to this mode.

The W field MUST be present and its size M MUST be 1 bit. WINDOW_SIZE MUST be equal to MAX_WIND_FCN + 1.

Each Profile, for each Rule ID value, MUST define

- the value of N (size of the FCN field),

- the value of MAX_WIND_FCN

- the size and algorithm for the MIC field in the SCHC F/R messages, if different from the default,

- the presence or absence of the DTag field in the SCHC F/R messages, as well as its size if it is present,
- the value of MAX_ACK_REQUESTS,
- the expiration time of the Retransmission Timer
- the expiration time of the Inactivity Timer

The sender, for each active pair of Rule ID and optional DTag values, MUST maintain

- one Attempts counter
- one Retransmission Timer

The receiver, for each pair of Rule ID and optional DTag values, MUST maintain

- one Inactivity Timer

8.4.2.1. Sender behaviour

At the beginning of the fragmentation of a new SCHC Packet, the fragment sender MUST select a Rule ID and DTag value pair for this SCHC Packet. For brevity, the rest of Section 8.4.2 only refers to SCHC F/R messages bearing the Rule ID and optional DTag values hereby selected.

Each SCHC Fragment MUST contain exactly one tile in its Payload. All tiles with the number 0 in their window, as well as the last tile, MUST be at least the size of an L2 Word.

In all SCHC Fragment messages, the W field MUST be filled with the least significant bit of the window number that the sender is currently processing.

If a SCHC Fragment carries a tile that is not the last one of the SCHC Packet,

- it MUST be of the Regular type specified in Section 8.3.1.1
- the FCN field MUST contain the tile number
- each tile MUST be of a size that complements the SCHC Fragment Header so that the SCHC Fragment is a multiple of L2 Words without the need for padding bits.

The SCHC Fragment that carries the last tile MUST be an All-1 SCHC Fragment, described in Section 8.3.1.2.

The bits on which the MIC is computed MUST be the SCHC Packet concatenated with the potential padding bits that are appended to the Payload of the SCHC Fragment that carries the last tile.

The fragment sender MUST start by processing the window numbered 0.

In a "blind transmission" phase, it MUST transmit all the tiles composing the window, in decreasing tile number.

Then, it enters an "equalization phase" in which it MUST initialize an Attempts counter to 0, it MUST start a Retransmission Timer and it MUST expect to receive a SCHC ACK. Then,

- on receiving a SCHC ACK,
  - if the SCHC ACK indicates that some tiles are missing at the receiver, then the sender MUST transmit all the tiles that have been reported missing, it MUST increment Attempts, it MUST reset the Retransmission Timer and MUST expect to receive a SCHC ACK again.
  - if the current window is not the last one and the SCHC ACK indicates that all tiles were correctly received, the sender MUST stop the Retransmission Timer, it MUST advance to the next fragmentation window and it MUST start a blind transmission phase as described above.
  - if the current window is the last one and the SCHC ACK indicates that more tiles were received than the sender actually sent, the fragment sender MUST send a SCHC Sender-Abort, it MUST release all resource associated with this SCHC Packet and it MAY exit with an error condition.
  - if the current window is the last one and the SCHC ACK indicates that all tiles were correctly received yet integrity check was a failure, the fragment sender MUST send a SCHC Sender-Abort, it MUST release all resource associated with this SCHC Packet and it MAY exit with an error condition.
  - if the current window is the last one and the SCHC ACK indicates that integrity checking was successful, the sender exits successfully.
- on Retransmission Timer expiration,

* if Attempts is strictly less that MAX_ACK_REQUESTS, the fragment sender MUST send a SCHC ACK REQ and MUST increment the Attempts counter.

* otherwise the fragment sender MUST send a SCHC Sender-Abort, it MUST release all resource associated with this SCHC Packet and it MAY exit with an error condition.

At any time,

- on receiving a SCHC Receiver-Abort, the fragment sender MUST release all resource associated with this SCHC Packet and it MAY exit with an error condition.

- on receiving a SCHC ACK that bears a W value different from the W value that it currently uses, the fragment sender MUST silently discard and ignore that SCHC ACK.

Figure 37 shows an example of a corresponding state machine.

8.4.2.2. Receiver behaviour

On receiving a SCHC Fragment with a Rule ID and optional DTag pair not being processed at that time

- the receiver MAY check if the optional DTag value has not recently been used for that Rule ID value, thereby ensuring that the received SCHC Fragment is not a remnant of a prior fragmented SCHC Packet transmission. If the SCHC Fragment is determined to be such a remant, the receiver MAY silently ignore it and discard it.

- the receiver MUST start a process to assemble a new SCHC Packet with that Rule ID and DTag value pair. That process MUST only examine received SCHC F/R messages with that Rule ID and DTag value pair and MUST only transmit SCHC F/R messages with that Rule ID and DTag value pair.

- the receiver MUST start an Inactivity Timer. It MUST initialise an Attempts counter to 0. It MUST initialise a window counter to 0.

In the rest of this section, "local W bit" means the least significant bit of the window counter of the receiver.

On reception of any SCHC F/R message, the receiver MUST reset the Inactivity Timer.

Entering an "acceptance phase", the receiver MUST first initialise an empty Bitmap for this window, then

- on receiving a SCHC Fragment or SCHC ACK REQ with the W bit different from the local W bit, the receiver MUST silently ignore and discard that message.

- on receiving a SCHC Fragment with the W bit equal to the local W bit, the receiver MUST assemble the received tile based on the window counter and on the FCN field in the SCHC Fragment and it MUST update the Bitmap.

  * if the SCHC Fragment received is an All-0 SCHC Fragment, the current window is determined to be a not-last window, and the receiver MUST send a SCHC ACK for this window. Then,

    + If the Bitmap indicates that all the tiles of the current window have been correctly received, the receiver MUST increment its window counter and it enters the "acceptance phase" for that new window.

    + If the Bitmap indicates that at least one tile is missing in the current window, the receiver enters the "equalization phase" for this window.

  * if the SCHC Fragment received is an All-1 SCHC Fragment, the padding bits of the All-1 SCHC Fragment MUST be assembled after the received tile, the current window is determined to be the last window, the receiver MUST perform the integrity check and it MUST send a SCHC ACK for this window. Then,

    + If the integrity check indicates that the full SCHC Packet has been correctly reassembled, the receiver MUST enter the "clean-up phase".

    + If the integrity check indicates that the full SCHC Packet has not been correctly reassembled, the receiver enters the "equalization phase" for this window.

- on receiving a SCHC ACK REQ with the W bit equal to the local W bit, the receiver has not yet determined if the current window is a not-last one or the last one, the receiver MUST send a SCHC ACK for this window, and it keeps accepting incoming messages.

In the "equalization phase":

- if the window is a not-last window

* on receiving a SCHC Fragment or SCHC ACK REQ with a W bit different from the local W bit the receiver MUST silently ignore and discard that message.

* on receiving a SCHC ACK REQ with a W bit equal to the local W bit, the receiver MUST send a SCHC ACK for this window.

* on receiving a SCHC Fragment with a W bit equal to the local W bit,

  + if the SCHC Fragment received is an All-1 SCHC Fragment, the receiver MUST silently ignore it and discard it.

  + otherwise, the receiver MUST update the Bitmap and it MUST assemble the tile received.

* on the Bitmap becoming fully populated with 1's, the receiver MUST send a SCHC ACK for this window, it MUST increment its window counter and it enters the "acceptance phase" for the new window.

o if the window is the last window

* on receiving a SCHC Fragment or SCHC ACK REQ with a W bit different from the local W bit the receiver MUST silently ignore and discard that message.

* on receiving a SCHC ACK REQ with a W bit equal to the local W bit, the receiver MUST send a SCHC ACK for this window.

* on receiving a SCHC Fragment with a W bit equal to the local W bit,

  + if the SCHC Fragment received is an All-0 SCHC Fragment, the receiver MUST silently ignore it and discard it.

  + otherwise, the receiver MUST update the Bitmap and it MUST assemble the tile received. If the SCHC Fragment received is an All-1 SCHC Fragment, the receiver MUST assemble the padding bits of the All-1 SCHC Fragment after the received tile. It MUST perform the integrity check. Then

    - if the integrity check indicates that the full SCHC Packet has been correctly reassembled, the receiver MUST send a SCHC ACK and it enters the "clean-up phase".

    - if the integrity check indicates that the full SCHC Packet has not been correctly reassembled,

- if the SCHC Fragment received was an All-1 SCHC Fragment, the receiver MUST send a SCHC ACK for this window

- it keeps accepting incoming messages.

In the "clean-up phase":

- Any received SCHC F/R message with a W bit different from the local W bit MUST be silently ignored and discarded.

- Any received SCHC F/R message different from an All-1 SCHC Fragment or a SCHC ACK REQ MUST be silently ignored and discarded.

- On receiving an All-1 SCHC Fragment or a SCHC ACK REQ, the receiver MUST send a SCHC ACK.

- On expiration of the Inactivity Timer, the receive process for that SCHC Packet MAY exit

At any time, on expiration of the Inactivity Timer, on receiving a SCHC Sender-Abort or when Attempts reaches MAX_ACK_REQUESTS, the receiver MUST send a SCHC Receiver-Abort, it MUST release all resource associated with this SCHC Packet and it MAY exit the receive process for that SCHC Packet.

The MIC computed at the receiver MUST be computed over the reassembled SCHC Packet and over the padding bits that were received in the SCHC Fragment carrying the last tile.

Figure 38 shows an example of a corresponding state machine.

8.4.3. ACK-on-Error

The ACK-on-Error mode supports LPWAN technologies that have variable MTU and out-of-order delivery.

In ACK-on-Error mode, windows are used. All tiles MUST be of equal size, except for the last one, which MUST be of the same size or smaller than the preceding ones. WINDOW_SIZE MUST be equal to MAX_WIND_FCN + 1.

A SCHC Fragment message carries one or more tiles, which may span multiple windows. A SCHC ACK reports on the reception of exactly one window of tiles.

See Figure 21 for an example.

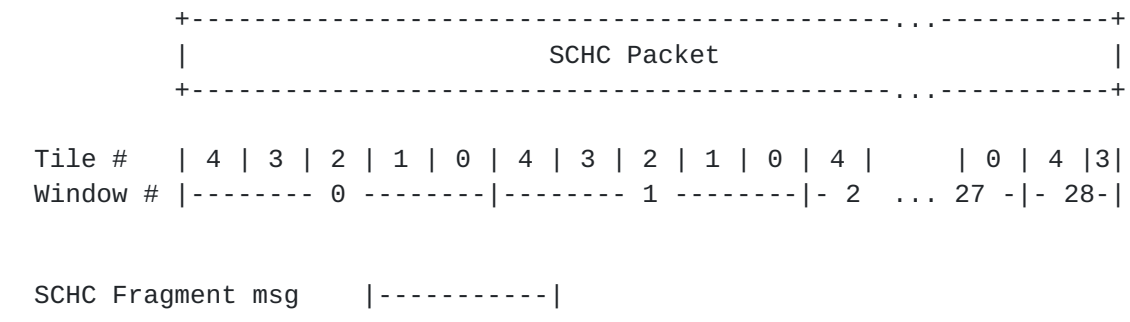


Figure 21: a SCHC Packet fragmented in tiles, Ack-on-Error mode

The W field is wide enough that it unambiguously represents an absolute window number. The fragment receiver feeds SCHC ACKs back to the fragment sender about windows that it misses tiles of. No SCHC ACK is fed back 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 can still later retransmit SCHC Fragments with tiles belonging to previous windows. Therefore, the sender and the receiver may operate in a fully 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 Rule ID value(s) is (are) allocated to this ACK-on-Error mode. For brevity, the rest of Section 8.4.3 only refers to SCHC F/R messages with Rule ID values that are allocated to this mode.

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

Each Profile, for each Rule ID 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 (size of the W field),

- the value of N (size of the FCN field),
- the value of MAX_WIND_FCN
- the size and algorithm for the MIC field in the SCHC F/R messages, if different from the default,
- the presence or absence of the DTag field in the SCHC F/R messages, as well as its size if it is present,
- the value of MAX_ACK_REQUESTS,
- the expiration time of the Retransmission Timer
- the expiration time of the Inactivity Timer

The sender, for each active pair of Rule ID and optional DTag values, MUST maintain

- one Attempts counter
- one Retransmission Timer

The receiver, for each pair of Rule ID and optional DTag values, MUST maintain

- one Inactivity Timer

8.4.3.1. Sender behaviour

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

- the fragment sender MUST select a Rule ID and DTag value pair for this SCHC Packet. A Rule MUST NOT be selected if the values of M and MAX_WIND_FCN for that Rule are such that the SCHC Packet cannot be fragmented in (2ˆM) * (MAX_WIND_FCN+1) tiles or less.
- the fragment sender MUST initialize the Attempts counter to 0 for that Rule ID and DTag value pair.

For brevity, the rest of Section 8.4.3 only refers to SCHC F/R messages bearing the Rule ID and optional DTag values hereby selected.

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

- the selected tiles MUST be consecutive in the original SCHC Packet
- 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.

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

If a SCHC Fragment carries more than one tile, or carries one tile that is not the last one of the SCHC Packet,

- it MUST be of the Regular type specified in Section 8.3.1.1
- the FCN field MUST contain the tile number of the first tile sent in that SCHC Fragment
- padding bits are appended to the tiles as needed to fit the Payload size constraint of Regular SCHC Fragments

The bits on which the MIC is computed MUST be the SCHC Packet concatenated with the padding bits that are appended to the Payload of the SCHC Fragment that carries the last tile.

The fragment sender MAY send the last tile as the Payload of an All-1 SCHC Fragment.

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.

Note that the last tile of a SCHC Packet can be sent in different ways, depending on Profiles and implementations

- in a Regular SCHC Fragment, either alone or as part of multiple tiles Payload
- in an All-1 SCHC Fragment

However, the last tile MUST NOT have ever been sent both in a Regular SCHC Fragment and in a All-1 SCHC Fragment.

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

- an All-1 SCHC Fragment

- or a SCHC ACK REQ with the W field corresponding to the last window.

A Profile MAY specify other times at which the fragment sender MUST listen for SCHC ACK messages.

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

- it MUST increment the Attempts counter
- it MUST reset the Retransmission Timer

On Retransmission Timer expiration

- if Attempts is strictly less than MAX_ACK_REQUESTS, the fragment sender MUST send a SCHC ACK REQ with the W field corresponding to the last window and it MUST increment the Attempts counter
- otherwise the fragment sender MUST send a SCHC Sender-Abort and it MUST release all resource associated with this SCHC Packet.

On receiving a SCHC ACK,

- if the W field in the SCHC ACK corresponds to the last window of the SCHC Packet,
  - if the C bit is set, the sender MAY release all resource associated with this SCHC Packet and MAY exit successfully
  - otherwise,
    - if the SCHC ACK shows no missing tile at the receiver, the sender
      - MUST send a SCHC Sender-Abort
      - MUST release all resource associated with this SCHC Packet
      - MAY exit with an error condition
    - otherwise
      - the fragment sender MUST send SCHC Fragment messages containing all the tiles that are reported missing in the SCHC ACK.

- if the last message in this sequence of SCHC Fragment messages is not an All-1 SCHC Fragment, then the fragment sender MUST send a SCHC ACK REQ with the W field corresponding to the last window after the sequence.

o otherwise, the fragment sender

  * MUST send SCHC Fragment messages containing the tiles that are reported missing in the SCHC ACK

  * then it MAY send a SCHC ACK REQ with the W field corresponding to the last window

See Figure 39 for one among several possible examples of a Finite State Machine implementing a sender behaviour obeying this specification.

8.4.3.2. Receiver behaviour

On receiving a SCHC Fragment with a Rule ID and optional DTag pair not being processed at that time

o the receiver MAY check if the optional DTag value has not recently been used for that Rule ID value, thereby ensuring that the received SCHC Fragment is not a remnant of a prior fragmented SCHC Packet transmission. If the SCHC Fragment is determined to be such a remant, the receiver MAY silently ignore it and discard it.

o the receiver MUST start a process to assemble a new SCHC Packet with that Rule ID and DTag value pair. That process MUST only examine received SCHC F/R messages with that Rule ID and DTag value pair and MUST only transmit SCHC F/R messages with that Rule ID and DTag value pair.

o the receiver MUST start an Inactivity Timer. It MUST initialise an Attempts counter to 0.

On reception of any SCHC F/R message, the receiver MUST reset the Inactivity Timer.

On reception of a SCHC Fragment message, the receiver MUST assemble the received tiles based on the W and FCN fields of the SCHC Fragment.

o 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 size and padding bits MUST be discarded. The latter is possible because
  - the size of the tiles is known a priori,
  - tiles are larger than an L2 Word
  - padding bits are always strictly less than an L2 Word

On reception of a SCHC ACK REQ or of an All-1 SCHC Fragment,

- if the receiver has at least one window that it knows has tiles missing, it MUST return a SCHC ACK for the lowest-numbered such window,
- otherwise,
  - if it has received at least one tile, it MUST return a SCHC ACK for the highest-numbered window it currently has tiles for
  - otherwise it MUST return a SCHC ACK for window numbered 0

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

On sending a SCHC ACK, the receiver MUST increase the Attempts counter.

From reception of an All-1 SCHC Fragment onward, a receiver MUST check the integrity of the reassembled SCHC Packet at least every time it prepares for sending a SCHC ACK for the last window.

On reception of a SCHC Sender-Abort, the receiver MUST release all resource associated with this SCHC Packet.

On expiration of the Inactivity Timer, the receiver MUST send a SCHC Receiver-Abort and it MUST release all resource associated with this SCHC Packet.

On the Attempts counter exceeding MAX_ACK_REQUESTS, the receiver MUST send a SCHC Receiver-Abort and it MUST release all resource associated with this SCHC Packet.

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 when at least an All-1 SCHC Fragment has been received and integrity checking of the reassembled SCHC Packet is successful.

The MIC computed at the receiver MUST be computed over the reassembled SCHC Packet and over the padding bits that were received in the SCHC Fragment carrying the last tile.

See Figure 40 for one among several possible examples of a Finite State Machine implementing a receiver behaviour obeying this specification, and that is meant to match the sender Finite State Machine of Figure 39.

9. Padding management

SCHC C/D and SCHC F/R operate on bits, not bytes. SCHC itself does not have any alignment prerequisite. The size of SCHC Packets can be any number of bits. If the layer below SCHC constrains the payload to align to some boundary, called L2 Words (for example, bytes), SCHC will meet that constraint and produce messages with the correct alignement. This may entail adding extra bits, called padding bits.

When padding occurs, the number of appended bits MUST be strictly less than the L2 Word size.

Padding happens at most once for each Packet during SCHC Compression and optional SCHC Fragmentation (see Figure 2). If a SCHC Packet is sent unfragmented (see Figure 22), it is padded as needed for transmission. If a SCHC Packet is fragmented, it is not padded in itself, only the SCHC Fragments are padded as needed for transmission. Some SCHC F/R modes only pad the very last SCHC Fragment.

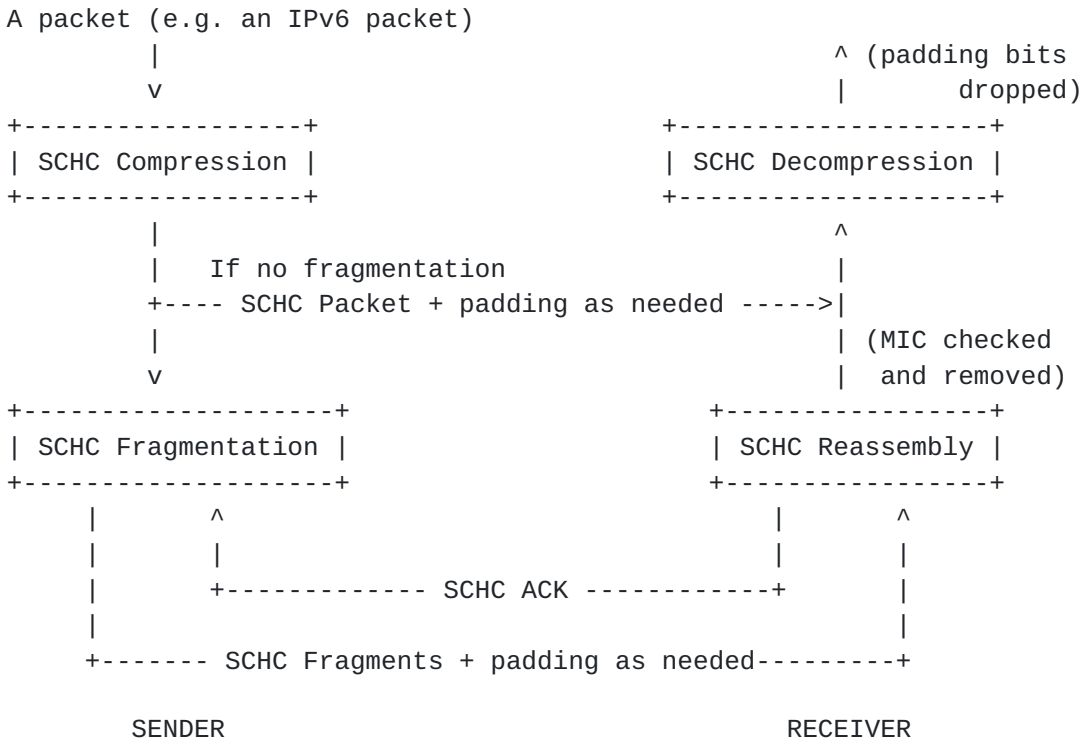


Figure 22: SCHC operations, including padding as needed

Each Profile MUST specify the size of the L2 Word. The L2 Word might actually be a single bit, in which case at most zero bits of padding will be appended to any message, i.e. no padding will take place at all.

A Profile MAY define the value of the padding bits. The RECOMMENDED value is 0.

10. SCHC Compression for IPv6 and UDP headers

This section lists the different IPv6 and UDP header fields and how they can be compressed.

10.1. IPv6 version field

This field always holds the same value. Therefore, in the Rule, TV is set to 6, MO to "equal" and CDA to "not-sent".

10.2. IPv6 Traffic class field

If the DiffServ field does not vary and is known by both sides, the Field Descriptor in the Rule SHOULD contain a TV with this well-known value, an "equal" MO and a "not-sent" CDA.

Otherwise (e.g. ECN bits are to be transmitted), two possibilities can be considered depending on the variability of the value:

- One possibility is to not compress the field and send the original value. In the Rule, TV is not set to any particular value, MO is set to "ignore" and CDA is set to "value-sent".

- If some upper bits in the field are constant and known, a better option is to only send the LSBs. In the Rule, TV is set to a value with the stable known upper part, MO is set to MSB(x) and CDA to LSB.

10.3. Flow label field

If the Flow Label field does not vary and is known by both sides, the Field Descriptor in the Rule SHOULD contain a TV with this well-known value, an "equal" MO and a "not-sent" CDA.

Otherwise, two possibilities can be considered:

- One possibility is to not compress the field and send the original value. In the Rule, TV is not set to any particular value, MO is set to "ignore" and CDA is set to "value-sent".

- If some upper bits in the field are constant and known, a better option is to only send the LSBs. In the Rule, TV is set to a value with the stable known upper part, MO is set to MSB(x) and CDA to LSB.

10.4. Payload Length field

This field can be elided for the transmission on the LPWAN network. The SCHC C/D recomputes the original payload length value. In the Field Descriptor, TV is not set, MO is set to "ignore" and CDA is "compute-IPv6-length".

If the payload length needs to be sent and does not need to be coded in 16 bits, the TV can be set to 0x0000, the MO set to MSB(16-s) where 's' is the number of bits to code the maximum length, and CDA is set to LSB.

10.5. Next Header field

If the Next Header field does not vary and is known by both sides, the Field Descriptor in the Rule SHOULD contain a TV with this Next Header value, the MO SHOULD be "equal" and the CDA SHOULD be "not-sent".

Otherwise, TV is not set in the Field Descriptor, MO is set to "ignore" and CDA is set to "value-sent". Alternatively, a matching-list MAY also be used.

10.6. Hop Limit field

The field behavior for this field is different for Uplink and Downlink. In Uplink, since there is no IP forwarding between the Dev and the SCHC C/D, the value is relatively constant. On the other hand, the Downlink value depends of Internet routing and MAY change more frequently. One neat way of processing this field is to use the Direction Indicator (DI) to distinguish both directions:

- in the Uplink, elide the field: the TV in the Field Descriptor is set to the known constant value, the MO is set to "equal" and the CDA is set to "not-sent".

- in the Downlink, send the value: TV is not set, MO is set to "ignore" and CDA is set to "value-sent".

10.7. IPv6 addresses fields

As in 6LoWPAN [RFC4944], IPv6 addresses are split into two 64-bit long fields; one for the prefix and one for the Interface Identifier (IID). These fields SHOULD be compressed. To allow for a single Rule being used for both directions, these values are identified by their role (DEV or APP) and not by their position in the header (source or destination).

10.7.1. IPv6 source and destination prefixes

Both ends MUST be synchronized with the appropriate prefixes. For a specific flow, the source and destination prefixes can be unique and stored in the context. It can be either a link-local prefix or a global prefix. In that case, the TV for the source and destination prefixes contain the values, the MO is set to "equal" and the CDA is set to "not-sent".

If the Rule is intended to compress packets with different prefix values, match-mapping SHOULD be used. The different prefixes are

listed in the TV, the MO is set to "match-mapping" and the CDA is set to "mapping-sent". See Figure 24

Otherwise, the TV contains the prefix, the MO is set to "equal" and the CDA is set to "value-sent".

10.7.2. IPv6 source and destination IID

If the DEV or APP IID are based on an LPWAN address, then the IID can be reconstructed with information coming from the LPWAN header. In that case, the TV is not set, the MO is set to "ignore" and the CDA is set to "DevIID" or "AppIID". Note that the LPWAN technology generally carries a single identifier corresponding to the DEV. Therefore AppIID cannot be used.

For privacy reasons or if the DEV address is changing over time, a static value that is not equal to the DEV address SHOULD be used. In that case, the TV contains the static value, the MO operator is set to "equal" and the CDA is set to "not-sent". [RFC7217] provides some methods that MAY be used to derive this static identifier.

If several IIDs are possible, then the TV contains the list of possible IIDs, the MO is set to "match-mapping" and the CDA is set to "mapping-sent".

It MAY also happen that the IID variability only expresses itself on a few bytes. In that case, the TV is set to the stable part of the IID, the MO is set to "MSB" and the CDA is set to "LSB".

Finally, the IID can be sent in extenso on the LPWAN. In that case, the TV is not set, the MO is set to "ignore" and the CDA is set to "value-sent".

10.8. IPv6 extensions

No Rule is currently defined that processes IPv6 extensions. If such extensions are needed, their compression/decompression Rules can be based on the MOs and CDAs described above.

10.9. UDP source and destination port

To allow for a single Rule being used for both directions, the UDP port values are identified by their role (DEV or APP) and not by their position in the header (source or destination). The SCHC C/D MUST be aware of the traffic direction (Uplink, Downlink) to select the appropriate field. The following Rules apply for DEV and APP port numbers.

If both ends know the port number, it can be elided. The TV contains the port number, the MO is set to "equal" and the CDA is set to "not-sent".

If the port variation is on few bits, the TV contains the stable part of the port number, the MO is set to "MSB" and the CDA is set to "LSB".

If some well-known values are used, the TV can contain the list of these values, the MO is set to "match-mapping" and the CDA is set to "mapping-sent".

Otherwise the port numbers are sent over the LPWAN. The TV is not set, the MO is set to "ignore" and the CDA is set to "value-sent".

10.10. UDP length field

The UDP length can be computed from the received data. In that case, the TV is not set, the MO is set to "ignore" and the CDA is set to "compute-length".

If the payload is small, the TV can be set to 0x0000, the MO set to "MSB" and the CDA to "LSB".

In other cases, the length SHOULD be sent and the CDA is replaced by "value-sent".

10.11. UDP Checksum field

The UDP checksum operation is mandatory with IPv6 [RFC8200] for most packets but recognizes that there are exceptions to that default behavior.

For instance, protocols that use UDP as a tunnel encapsulation may enable zero-checksum mode for a specific port (or set of ports) for sending and/or receiving. [RFC8200] also stipulates that any node implementing zero-checksum mode must follow the requirements specified in "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums" [RFC6936].

6LoWPAN Header Compression [RFC6282] also authorizes to send UDP datagram that are deprived of the checksum protection when an upper layer guarantees the integrity of the UDP payload and pseudo-header all the way between the compressor that elides the UDP checksum and the decompressor that computes again it. A specific example of this is when a Message Integrity Check (MIC) protects the compressed message all along that path with a strength that is identical or better to the UDP checksum.

In a similar fashion, this specification allows a SCHC compressor to elide the UDP checks when another layer guarantees an identical or better integrity protection for the UDP payload and the pseudo-header. In this case, the TV is not set, the MO is set to "ignore" and the CDA is set to "compute-checksum".

In particular, when SCHC fragmentation is used, a fragmentation MIC of 2 bytes or more provides equal or better protection than the UDP checksum; in that case, if the compressor is collocated with the fragmentation point and the decompressor is collocated with the packet reassembly point, then compressor MAY elide the UDP checksum. Whether and when the UDP Checksum is elided is to be specified in the Profile.

Since the compression happens before the fragmentation, implementors should understand the risks when dealing with unprotected data below the transport layer and take special care when manipulating that data.

In other cases, the checksum SHOULD be explicitly sent. The TV is not set, the MO is set to "ignore" and the CDA is set to "value-sent".

11. IANA Considerations

This document has no request to IANA.

12. Security considerations

12.1. Security considerations for SCHC Compression/Decompression

A malicious header compression could cause the reconstruction of a wrong packet that does not match with the original one. Such a corruption MAY be detected with end-to-end authentication and integrity mechanisms. Header Compression does not add more security problem than what is already needed in a transmission. For instance, to avoid an attack, never re-construct a packet bigger than some configured size (with 1500 bytes as generic default).

12.2. Security considerations for SCHC Fragmentation/Reassembly

This subsection describes potential attacks to LPWAN SCHC F/R and suggests possible countermeasures.

A node can perform a buffer reservation attack by sending a first SCHC Fragment to a target. Then, the receiver will reserve buffer space for the IPv6 packet. Other incoming fragmented SCHC Packets will be dropped while the reassembly buffer is occupied during the

reassembly timeout. Once that timeout expires, the attacker can repeat the same procedure, and iterate, thus creating a denial of service attack. The (low) cost to mount this attack is linear with the number of buffers at the target node. However, the cost for an attacker can be increased if individual SCHC Fragments of multiple packets can be stored in the reassembly buffer. To further increase the attack cost, the reassembly buffer can be split into SCHC Fragment-sized buffer slots. Once a packet is complete, it is processed normally. If buffer overload occurs, a receiver can discard packets based on the sender behavior, which MAY help identify which SCHC Fragments have been sent by an attacker.

In another type of attack, the malicious node is required to have overhearing capabilities. If an attacker can overhear a SCHC Fragment, it can send a spoofed duplicate (e.g. with random payload) to the destination. If the LPWAN technology does not support suitable protection (e.g. source authentication and frame counters to prevent replay attacks), a receiver cannot distinguish legitimate from spoofed SCHC Fragments. Therefore, the original IPv6 packet will be considered corrupt and will be dropped. To protect resource-constrained nodes from this attack, it has been proposed to establish a binding among the SCHC Fragments to be transmitted by a node, by applying content-chaining to the different SCHC Fragments, based on cryptographic hash functionality. The aim of this technique is to allow a receiver to identify illegitimate SCHC Fragments.

Further attacks MAY involve sending overlapped fragments (i.e. comprising some overlapping parts of the original IPv6 datagram). Implementers SHOULD make sure that the correct operation is not affected by such event.

In ACK-on-Error, a malicious node MAY force a SCHC Fragment sender to resend a SCHC Fragment a number of times, with the aim to increase consumption of the SCHC Fragment sender's resources. To this end, the malicious node MAY repeatedly send a fake ACK to the SCHC Fragment sender, with a Bitmap that reports that one or more SCHC Fragments have been lost. In order to mitigate this possible attack, MAX_ACK_RETRIES MAY be set to a safe value which allows to limit the maximum damage of the attack to an acceptable extent. However, note that a high setting for MAX_ACK_RETRIES benefits SCHC Fragment reliability modes, therefore the trade-off needs to be carefully considered.

13. Acknowledgements

Thanks to Carsten Bormann, Philippe Clavier, Diego Dujovne, Eduardo Ingles Sanchez, Arunprabhu Kandasamy, Rahul Jadhav, Sergio Lopez Bernal, Antony Markovski, Alexander Pelov, Charles Perkins, Edgar

Ramos, Shoichi Sakane, and Pascal Thubert for useful design consideration and comments.

14. References

14.1. 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>.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <https://www.rfc-editor.org/info/rfc7217>.

[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>.

14.2. Informative References

[RFC3385] Sheinwald, D., Satran, J., Thaler, P., and V. Cavanna, "Internet Protocol Small Computer System Interface (iSCSI) Cyclic Redundancy Check (CRC)/Checksum Considerations", RFC 3385, DOI 10.17487/RFC3385, September 2002, <https://www.rfc-editor.org/info/rfc3385>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, DOI 10.17487/RFC5795, March 2010, <https://www.rfc-editor.org/info/rfc5795>.

[RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <https://www.rfc-editor.org/info/rfc6282>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <https://www.rfc-editor.org/info/rfc6936>.

[RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136, February 2014, <https://www.rfc-editor.org/info/rfc7136>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[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>.

Appendix A. SCHC Compression Examples

This section gives some scenarios of the compression mechanism for IPv6/UDP. The goal is to illustrate the behavior of SCHC.

The most common case using the mechanisms defined in this document will be a LPWAN Dev that embeds some applications running over CoAP. In this example, three flows are considered. The first flow is for the device management based on CoAP using Link Local IPv6 addresses and UDP ports 123 and 124 for Dev and App, respectively. The second flow will be a CoAP server for measurements done by the Device (using ports 5683) and Global IPv6 Address prefixes alpha::IID/64 to beta::1/64. The last flow is for legacy applications using different ports numbers, the destination IPv6 address prefix is gamma::1/64.

Figure 23 presents the protocol stack for this Device. IPv6 and UDP are represented with dotted lines since these protocols are compressed on the radio link.

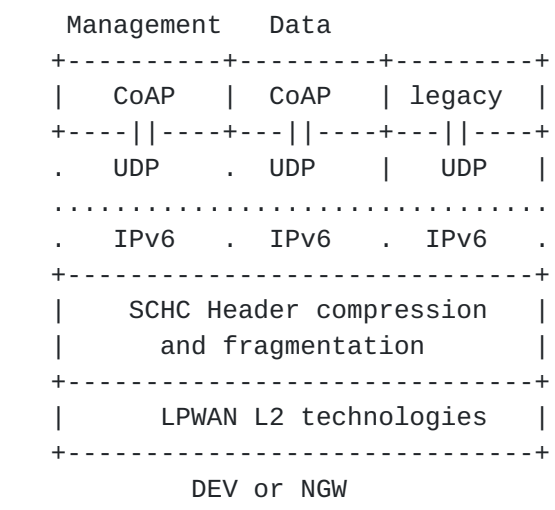


Figure 23: Simplified Protocol Stack for LP-WAN

Note that in some LPWAN technologies, only the Devs have a device ID. Therefore, when such technologies are used, it is necessary to statically define an IID for the Link Local address for the SCHC C/D.

Rule 0

Field	FL	FP	DI	Value	Match Opera.	Comp Decomp Action	Sent [bits]
IPv6 version	4	1	Bi	6	equal	not-sent	
IPv6 DiffServ	8	1	Bi	0	equal	not-sent	
IPv6 Flow Label	20	1	Bi	0	equal	not-sent	
IPv6 Length	16	1	Bi		ignore	comp-length	
IPv6 Next Header	8	1	Bi	17	equal	not-sent	
IPv6 Hop Limit	8	1	Bi	255	ignore	not-sent	
IPv6 DEVprefix	64	1	Bi	FE80::/64	equal	not-sent	
IPv6 DevIID	64	1	Bi		ignore	DevIID	
IPv6 APPprefix	64	1	Bi	FE80::/64	equal	not-sent	
IPv6 AppIID	64	1	Bi	::1	equal	not-sent	
UDP DEVport	16	1	Bi	123	equal	not-sent	
UDP APPport	16	1	Bi	124	equal	not-sent	
UDP Length	16	1	Bi		ignore	comp-length	
UDP checksum	16	1	Bi		ignore	comp-chk	

Rule 1

Field	FL	FP	DI	Value	Match Opera.	Action Action	Sent [bits]

IPv6 version	4	1	Bi	6	equal	not-sent		
IPv6 DiffServ	8	1	Bi	0	equal	not-sent		
IPv6 Flow Label	20	1	Bi	0	equal	not-sent		
IPv6 Length	16	1	Bi		ignore	comp-length		
IPv6 Next Header	8	1	Bi	17	equal	not-sent		
IPv6 Hop Limit	8	1	Bi	255	ignore	not-sent		
IPv6 DEVprefix	64	1	Bi	[alpha/64, fe80::/64]	match-mapping	mapping-sent		1
IPv6 DevIID	64	1	Bi		ignore	DevIID		
IPv6 APPprefix	64	1	Bi	[beta/64, alpha/64, fe80::64]	match-mapping	mapping-sent		2
IPv6 AppIID	64	1	Bi	::1000	equal	not-sent		
UDP DEVport	16	1	Bi	5683	equal	not-sent		
UDP APPport	16	1	Bi	5683	equal	not-sent		
UDP Length	16	1	Bi		ignore	comp-length		
UDP checksum	16	1	Bi		ignore	comp-chk		

Rule 2

Field	FL	FP	DI	Value	Match Opera.	Action	Action	Sent [bits]
IPv6 version	4	1	Bi	6	equal	not-sent		
IPv6 DiffServ	8	1	Bi	0	equal	not-sent		
IPv6 Flow Label	20	1	Bi	0	equal	not-sent		
IPv6 Length	16	1	Bi		ignore	comp-length		
IPv6 Next Header	8	1	Bi	17	equal	not-sent		
IPv6 Hop Limit	8	1	Up	255	ignore	not-sent		
IPv6 Hop Limit	8	1	Dw		ignore	value-sent		8
IPv6 DEVprefix	64	1	Bi	alpha/64	equal	not-sent		
IPv6 DevIID	64	1	Bi		ignore	DevIID		
IPv6 APPprefix	64	1	Bi	gamma/64	equal	not-sent		
IPv6 AppIID	64	1	Bi	::1000	equal	not-sent		
UDP DEVport	16	1	Bi	8720	MSB(12)	LSB		4
UDP APPport	16	1	Bi	8720	MSB(12)	LSB		4
UDP Length	16	1	Bi		ignore	comp-length		
UDP checksum	16	1	Bi		ignore	comp-chk		

Figure 24: Context Rules

All the fields described in the three Rules depicted on Figure 24 are present in the IPv6 and UDP headers. The DevIID-DID value is found in the L2 header.

The second and third Rules use global addresses. The way the Dev learns the prefix is not in the scope of the document.

The third Rule compresses port numbers to 4 bits.

Appendix B. Fragmentation Examples

This section provides examples for the different fragment reliability modes specified in this document.

Figure 25 illustrates the transmission in No-ACK mode of a SCHC Packet that needs 11 SCHC Fragments. FCN is 1 bit wide.

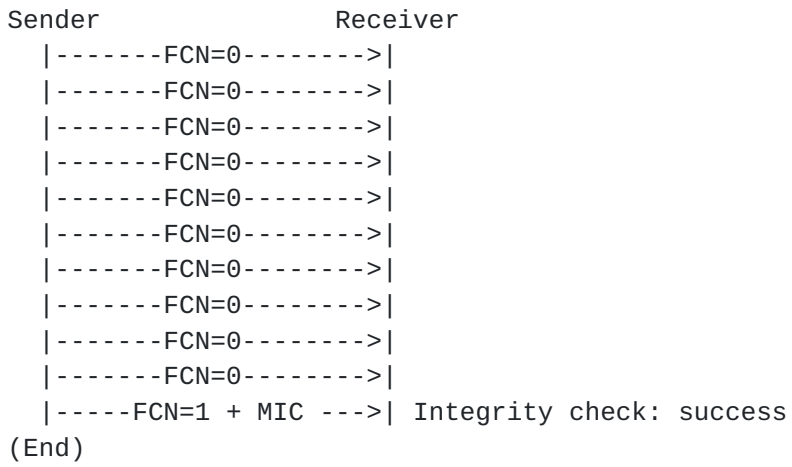


Figure 25: Transmission in No-ACK mode of a SCHC Packet carried by 11 SCHC Fragments

In the following examples, N (the size of the FCN field) is 3 bits. Therefore, the All-1 FCN value is 7.

Figure 26 illustrates the transmission in ACK-on-Error mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, MAX_WIND_FCN=6 and no lost SCHC Fragment.

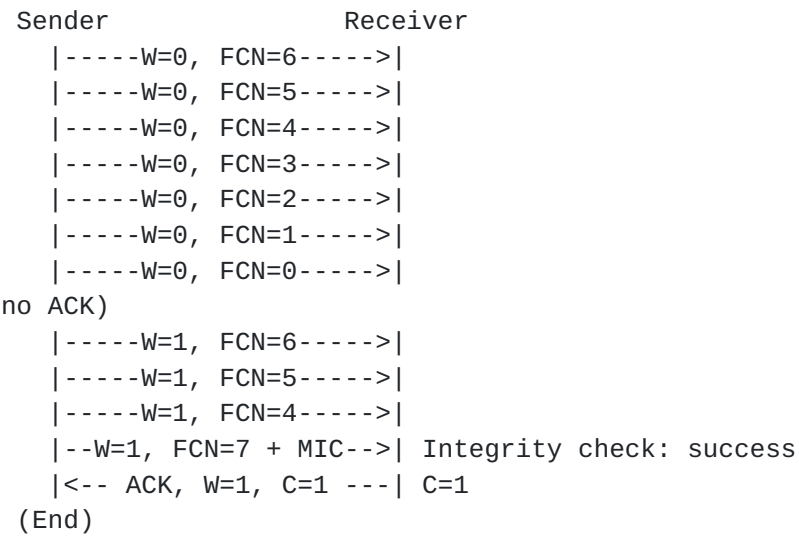


Figure 26: Transmission in ACK-on-Error mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, MAX_WIND_FCN=6 and no lost SCHC Fragment.

Figure 27 illustrates the transmission in ACK-on-Error mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, MAX_WIND_FCN=6 and three lost SCHC Fragments.

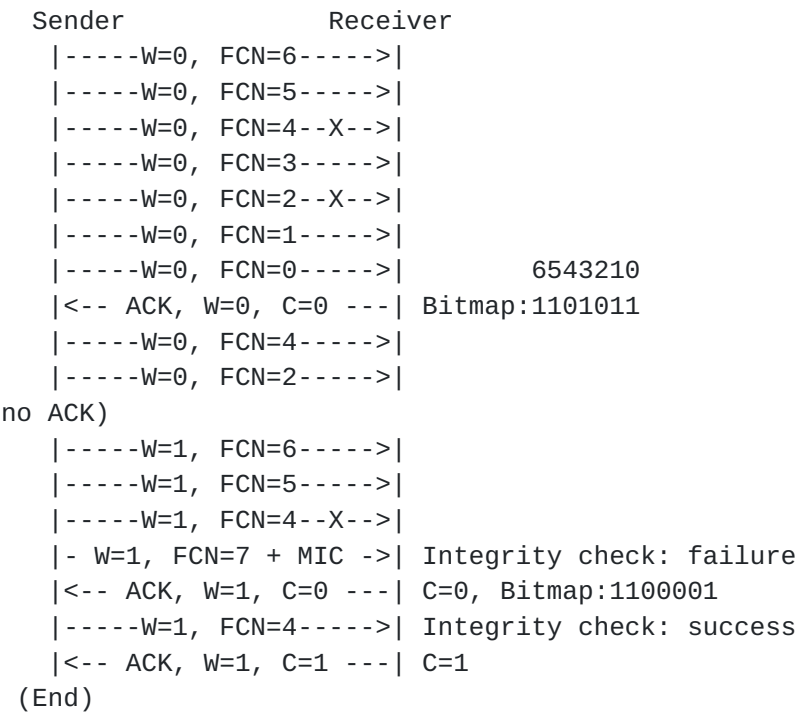


Figure 27: Transmission in ACK-on-Error mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, MAX_WIND_FCN=6 and three lost SCHC Fragments.

Figure 28 shows an example of a transmission in ACK-on-Error mode of a SCHC Packet fragmented in 73 tiles, with N=5, MAX_WIND_FCN=27, M=2 and 3 lost SCHC Fragments.

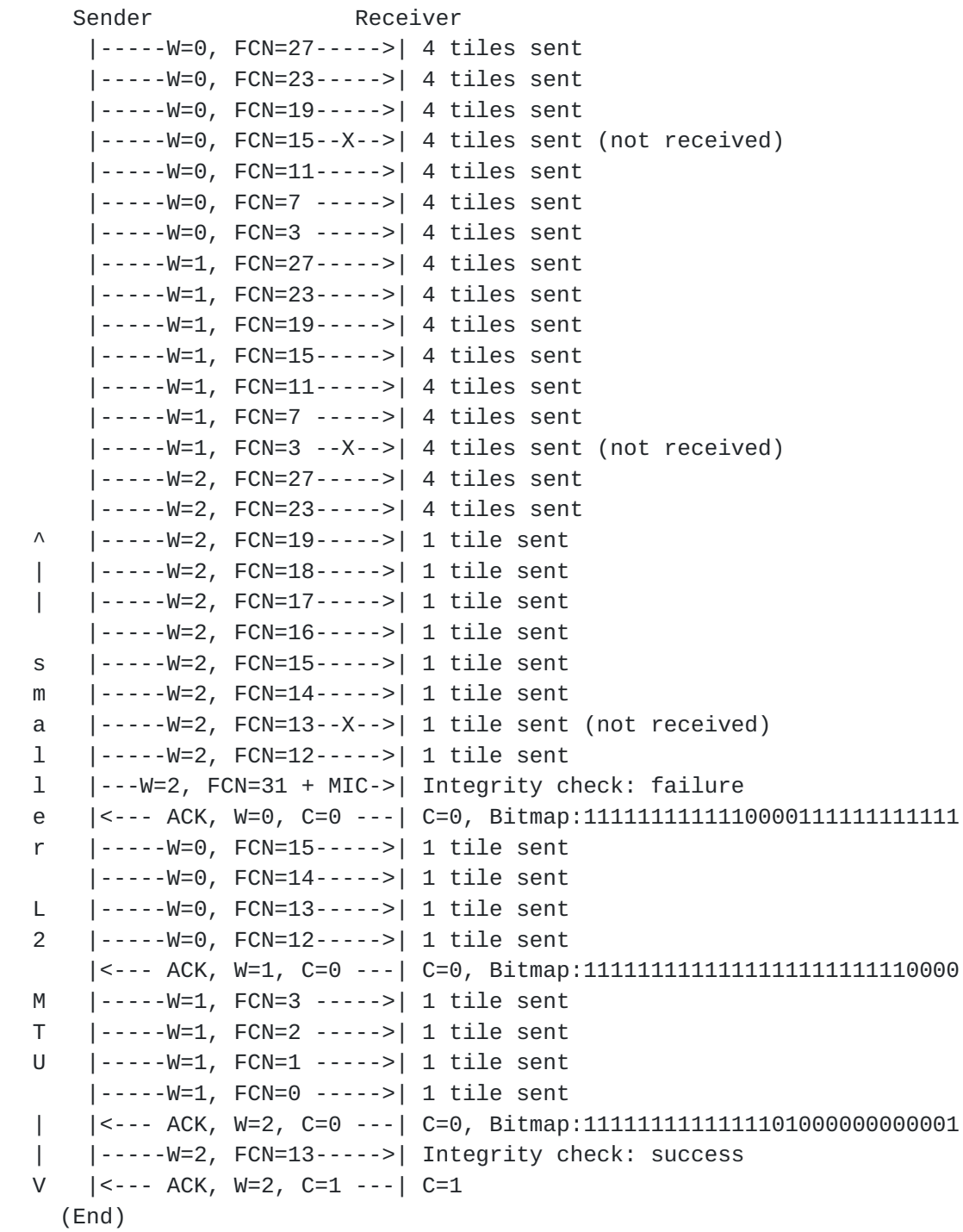


Figure 28: ACK-on-Error mode with variable MTU.

In this example, the L2 MTU becomes reduced just before sending the "W=2, FCN=19" fragment, leaving space for only 1 tile in each forthcoming SCHC Fragment. Before retransmissions, the 73 tiles are carried by a total of 25 SCHC Fragments, the last 9 being of smaller size.

Note 1: Bitmaps are shown prior to compression for transmission

Note 2: other sequences of events (e.g. regarding when ACKs are sent by the Receiver) are also allowed by this specification. Profiles may restrict this flexibility.

Figure 29 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, with N=3, MAX_WIND_FCN=6 and no loss.

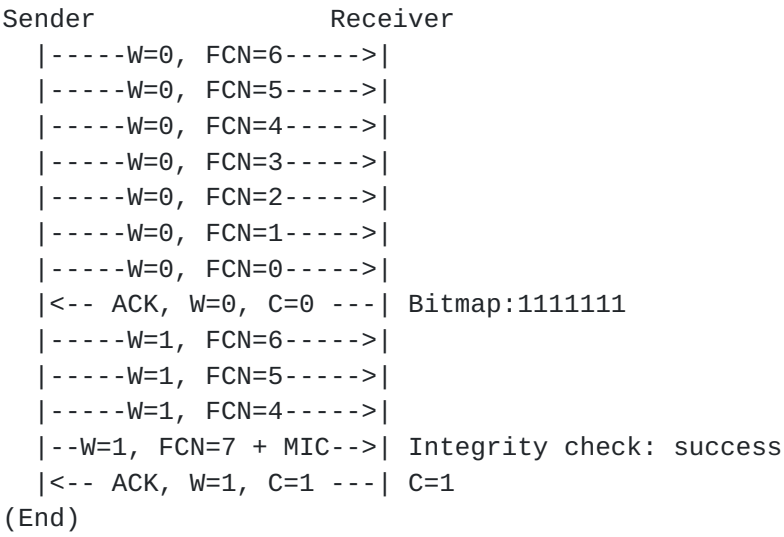


Figure 29: Transmission in ACK-Always mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, with N=3, MAX_WIND_FCN=6 and no loss.

Figure 30 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, N=3, MAX_WIND_FCN=6 and three lost SCHC Fragments.

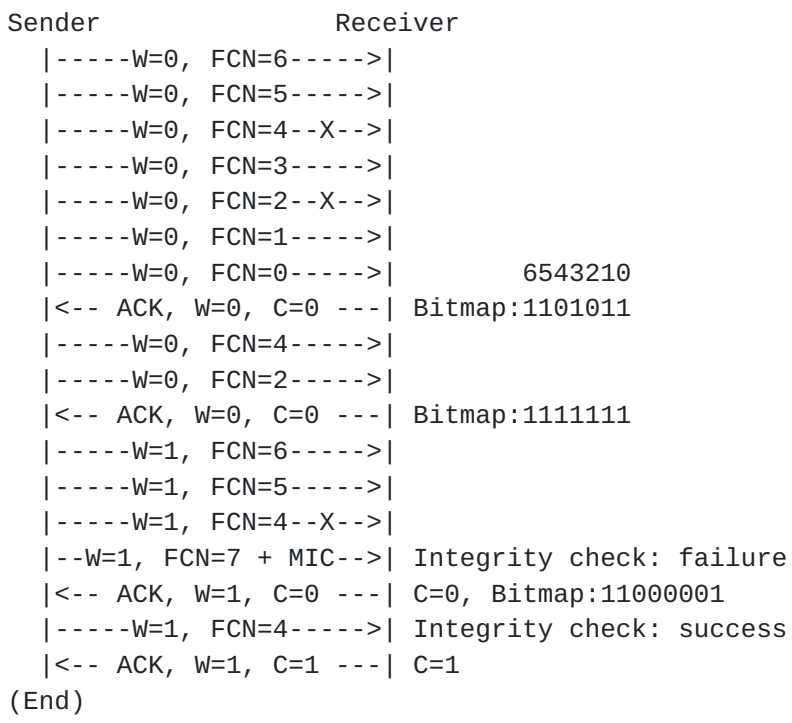


Figure 30: Transmission in ACK-Always mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, N=3, MAX_WIND_FCN=6 and three lost SCHC Fragments.

Figure 31 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with one tile per SCHC Fragment, N=3, MAX_WIND_FCN=6, three lost SCHC Fragments and only one retry needed to recover each lost SCHC Fragment.

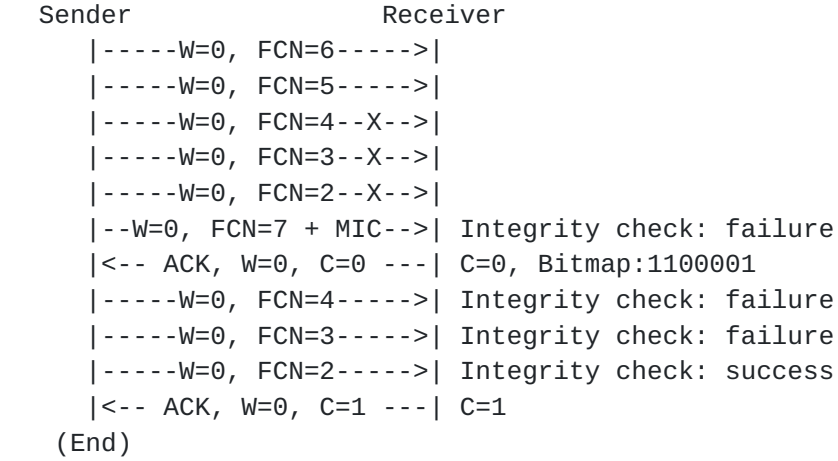


Figure 31: Transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with one tile per SCHC Fragment, N=3, MAX_WIND_FCN=6, three lost SCHC Fragments.

Figure 32 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with one tile per SCHC Fragment, N=3, MAX_WIND_FCN=6, three lost SCHC Fragments, and the second SCHC ACK lost.

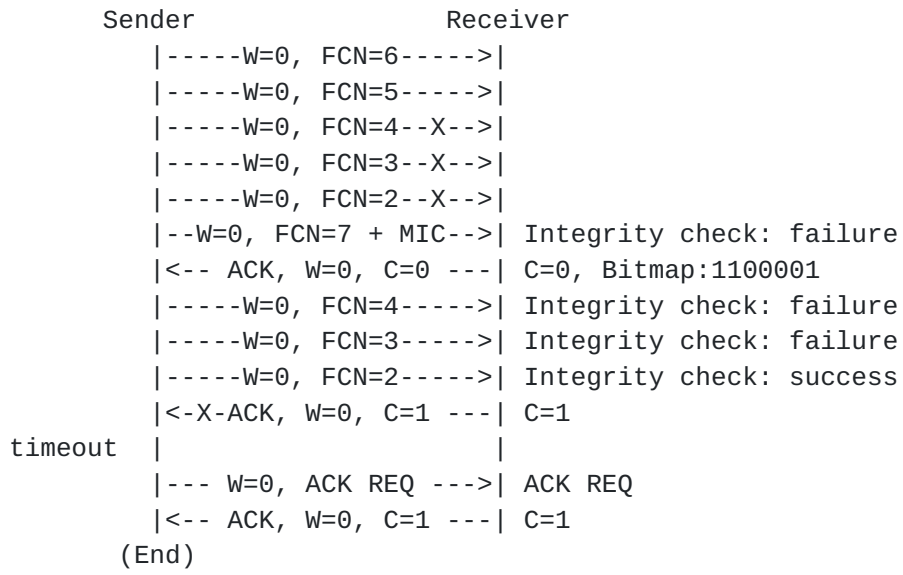


Figure 32: Transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with one tile per SCHC Fragment, N=3, MAX_WIND_FCN=6, three lost SCHC Fragments, and the second SCHC ACK lost.

Figure 33 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with N=3, MAX_WIND_FCN=6, with three lost SCHC Fragments, and one retransmitted SCHC Fragment lost again.

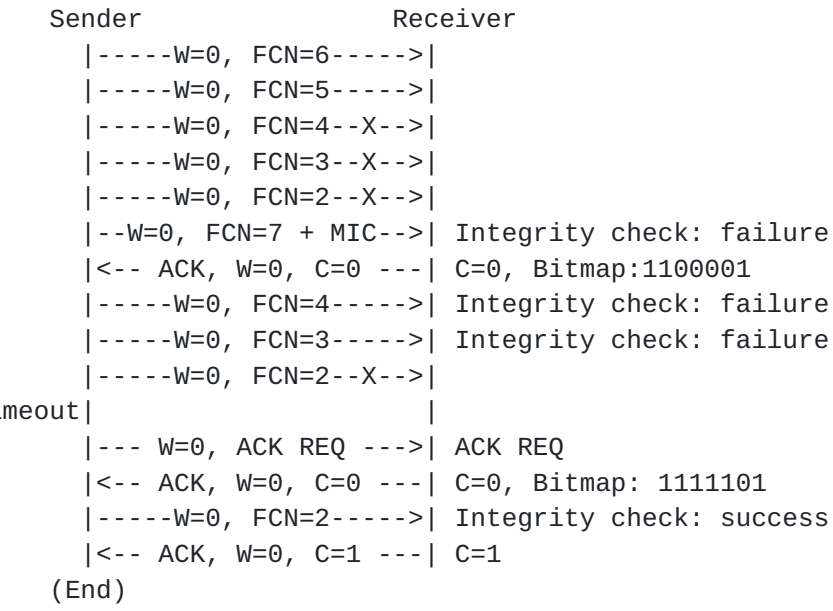


Figure 33: Transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with N=3, MAX_WIND_FCN=6, with three lost SCHC Fragments, and one retransmitted SCHC Fragment lost again.

Figure 34 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 28 tiles, with one tile per SCHC Fragment, N=5, MAX_WIND_FCN=23 and two lost SCHC Fragments.

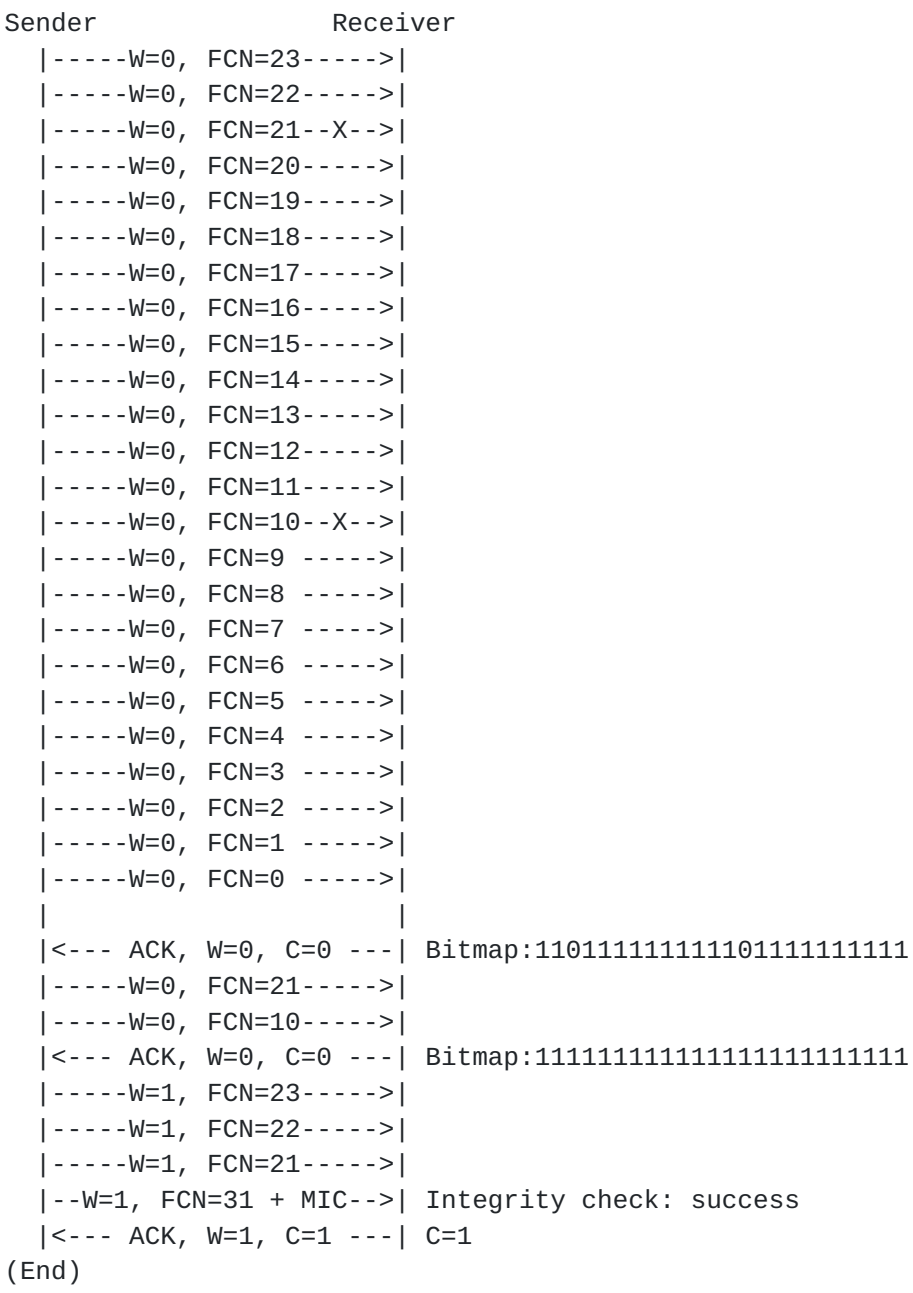


Figure 34: Transmission in ACK-Always mode of a SCHC Packet fragmented in 28 tiles, with one tile per SCHC Fragment, N=5, MAX_WIND_FCN=23 and two lost SCHC Fragments.

Appendix C. Fragmentation State Machines

The fragmentation state machines of the sender and the receiver, one for each of the different reliability modes, are described in the following figures:

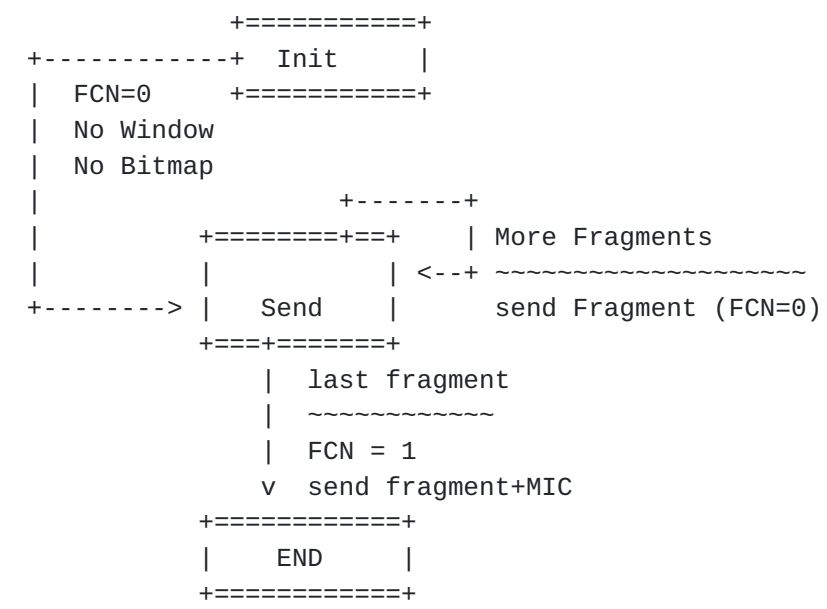


Figure 35: Sender State Machine for the No-ACK Mode

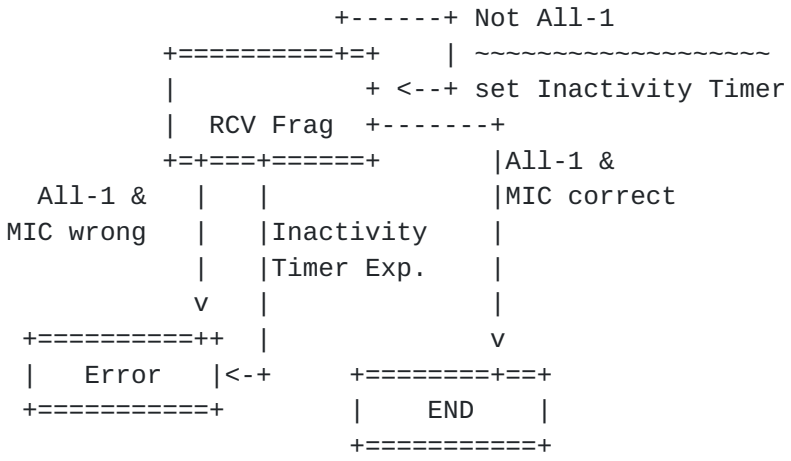


Figure 36: Receiver State Machine for the No-ACK Mode

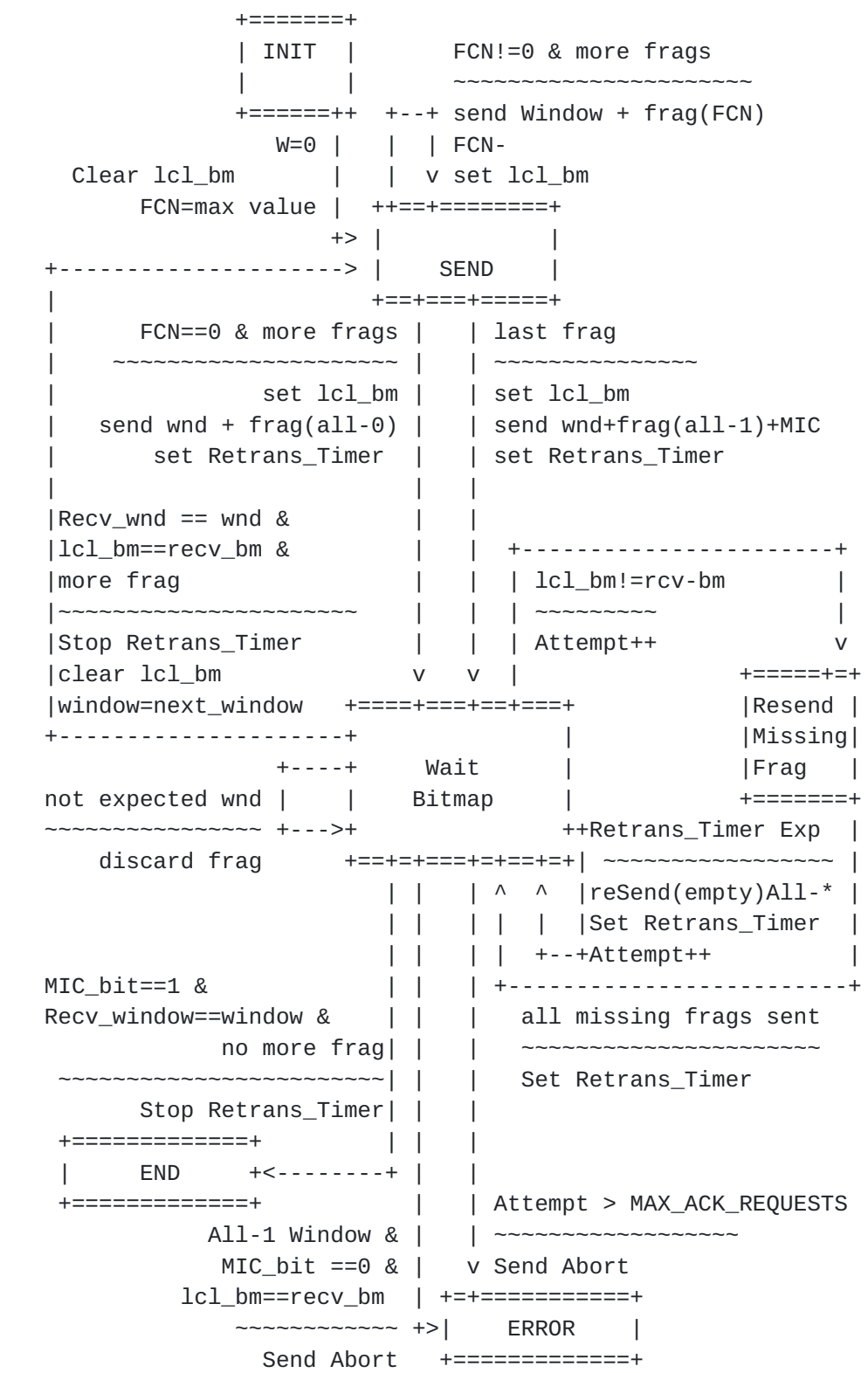
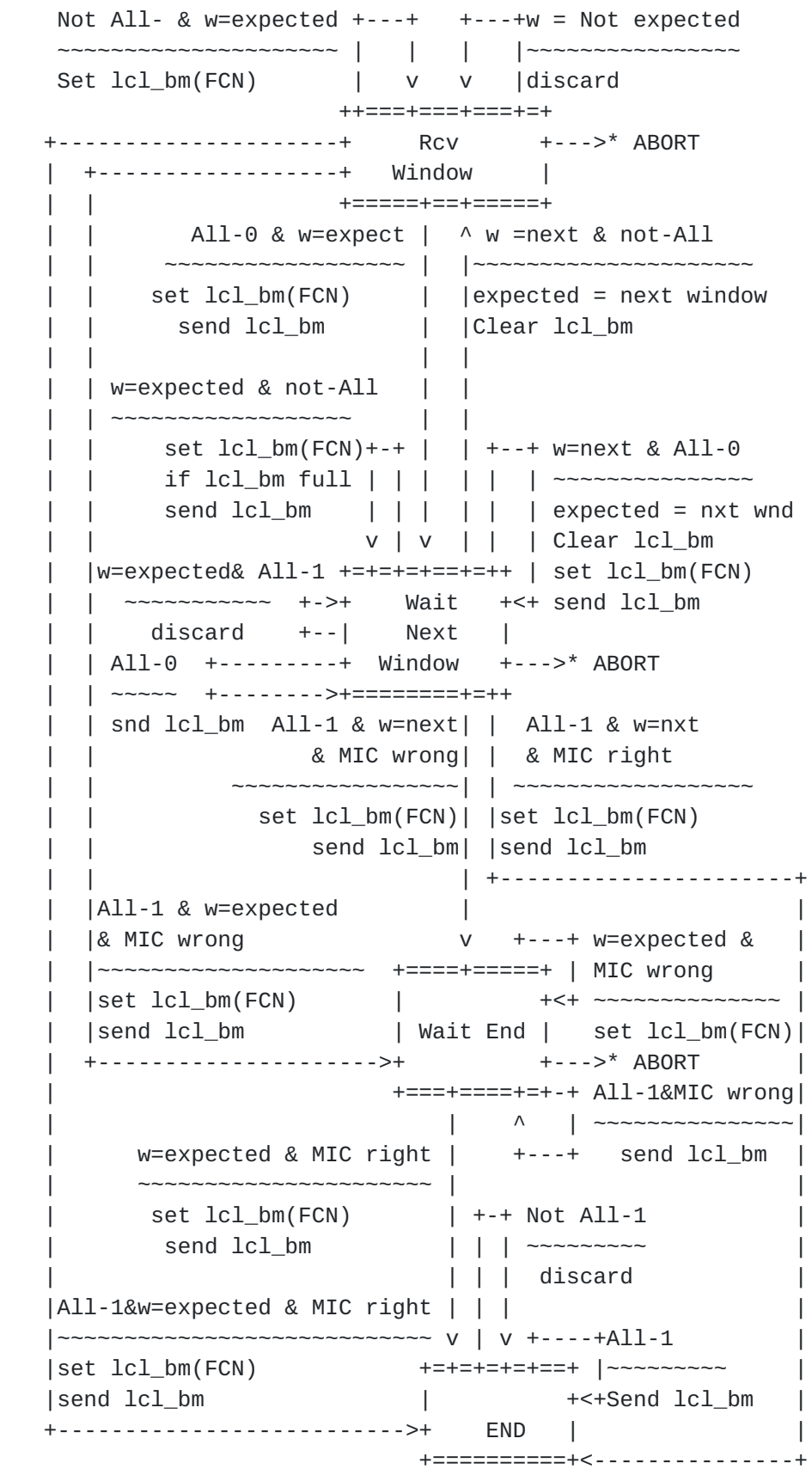
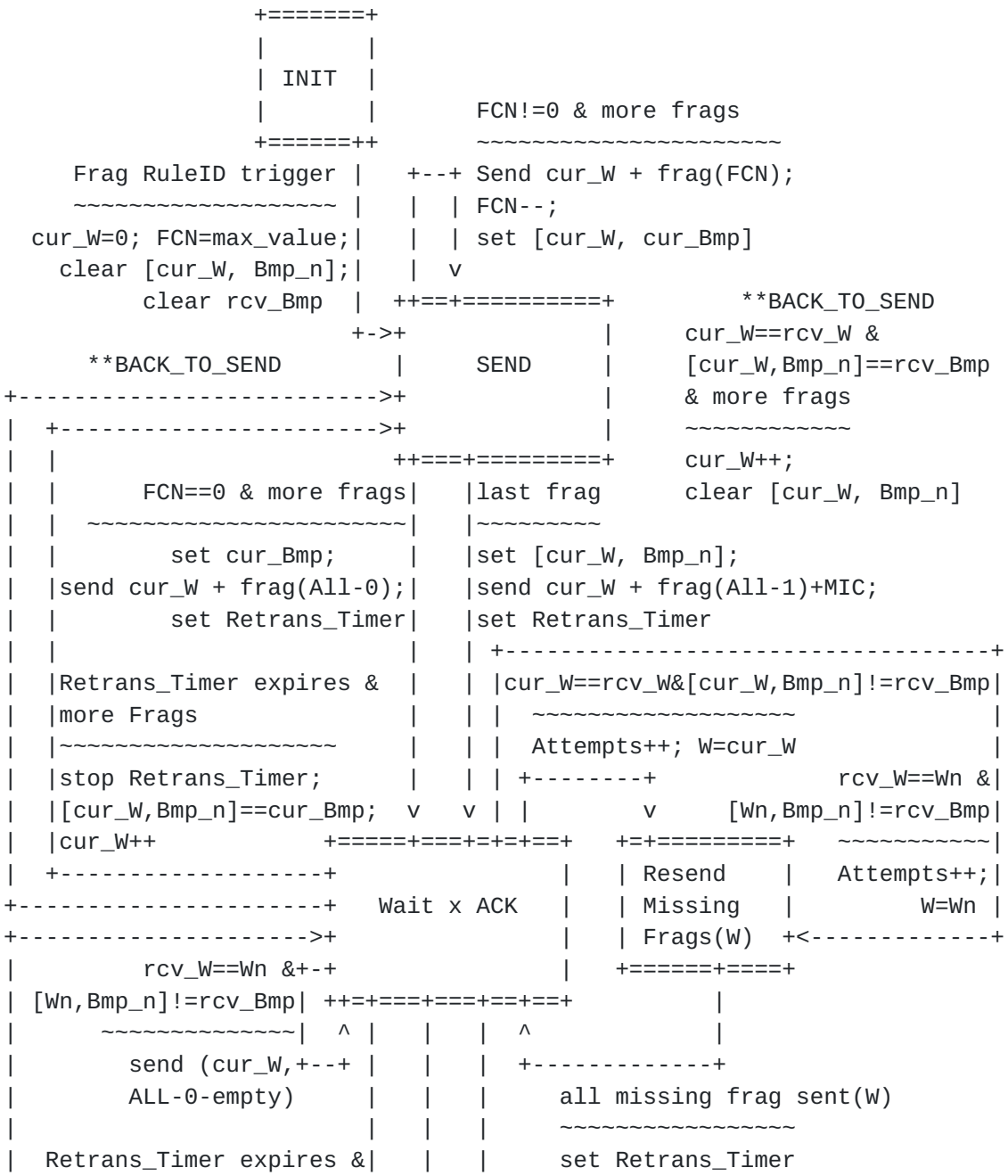


Figure 37: Sender State Machine for the ACK-Always Mode


Not All- & w=expected
Set lcl_bm(FCN)
w = Not expected
discard
Rcv Window
* ABORT
All-0 & w=expect
set lcl_bm(FCN)
send lcl_bm
^ w =next & not-All
expected = next window
Clear lcl_bm
w=expected & not-All
set lcl_bm(FCN)
if lcl_bm full
send lcl_bm
w=next & All-0
expected = nxt wnd
Clear lcl_bm
w=expected& All-1
set lcl_bm(FCN)
discard
Wait
Next
send lcl_bm
All-0
Window
* ABORT
snd lcl_bm
All-1 & w=next
& MIC wrong
All-1 & w=nxt
& MIC right
set lcl_bm(FCN)
send lcl_bm
All-1 & w=expected
& MIC wrong
w=expected &
MIC wrong
set lcl_bm(FCN)
send lcl_bm
Wait End
set lcl_bm(FCN)
* ABORT
All-1&MIC wrong
w=expected & MIC right
send lcl_bm
set lcl_bm(FCN)
send lcl_bm
Not All-1
discard
All-1&w=expected & MIC right
All-1
set lcl_bm(FCN)
send lcl_bm
Send lcl_bm
END

```
          --->* ABORT
               ~~~~~~~
               Inactivity_Timer = expires
           When DWL
             IF Inactivity_Timer expires
                Send DWL Request
                Attempt++
```

Figure 38: Receiver State Machine for the ACK-Always Mode



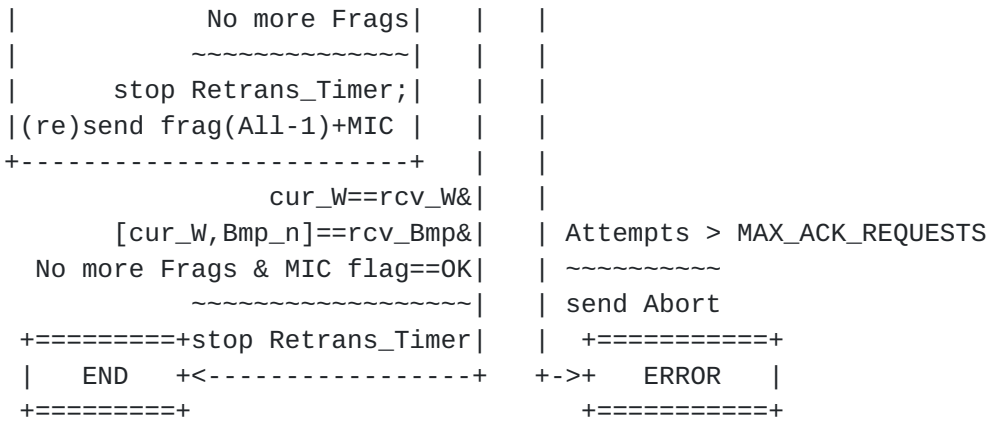


Figure 39: Sender State Machine for the ACK-on-Error Mode

This is an example only. The specification in Section 8.4.3.1 is open to very different sequencing of operations.

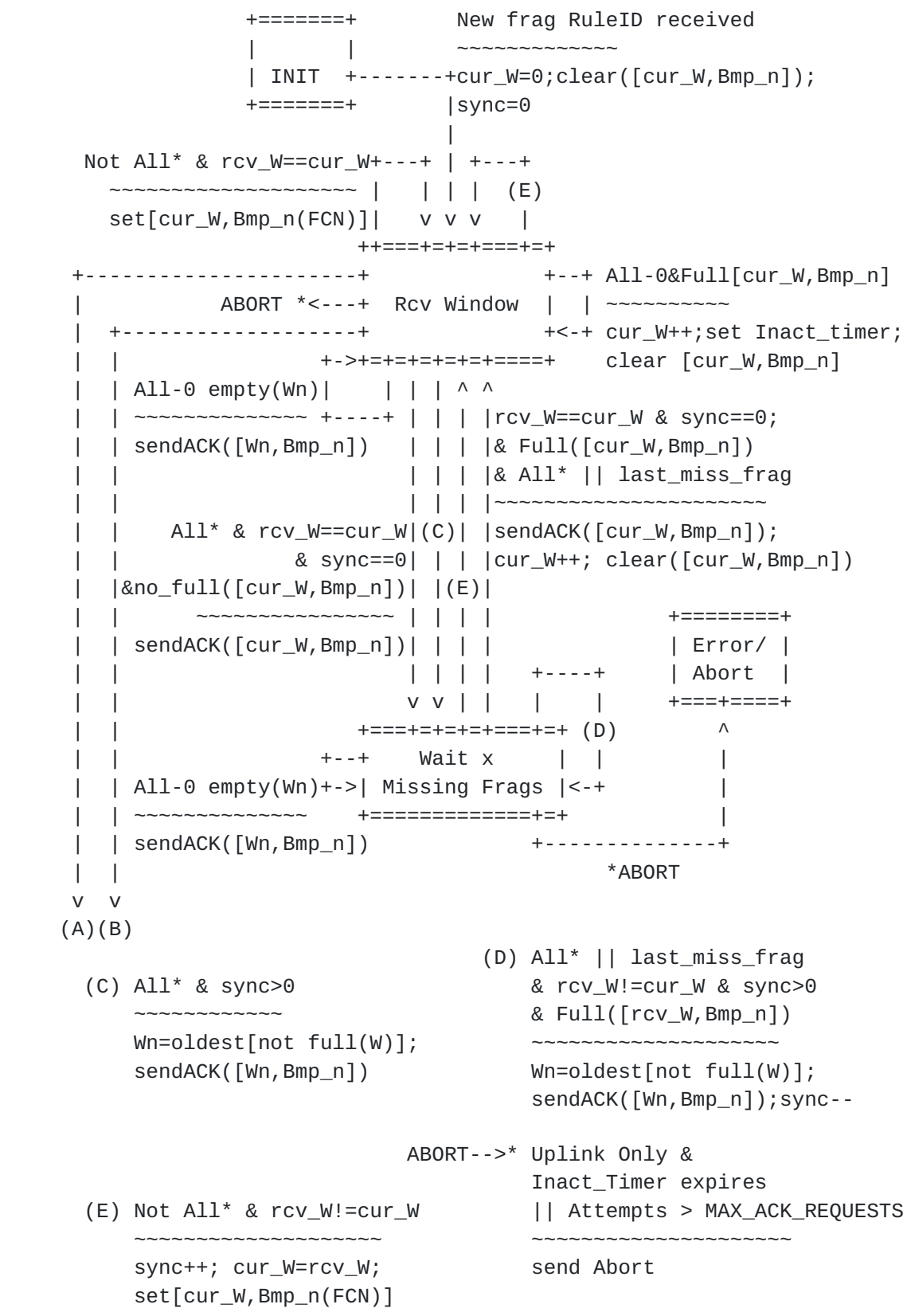

New frag RuleID received
INIT
cur_W=0;clear([cur_W,Bmp_n]);
sync=0
Not All* & rcv_W==cur_W
set[cur_W,Bmp_n(FCN)]
ABORT *<---+ Rcv Window
All-0&Full[cur_W,Bmp_n]
cur_W++;set Inact_timer;
clear [cur_W,Bmp_n]
All-0 empty(Wn)
sendACK([Wn,Bmp_n])
rcv_W==cur_W & sync==0;
& Full([cur_W,Bmp_n])
& All* || last_miss_frag
All* & rcv_W==cur_W
& sync==0
(C)
sendACK([cur_W,Bmp_n]);
cur_W++; clear([cur_W,Bmp_n])
&no_full([cur_W,Bmp_n])
(E)
sendACK([cur_W,Bmp_n])
Error/ Abort
(D)
Wait x
Missing Frags
All-0 empty(Wn)
sendACK([Wn,Bmp_n])
*ABORT
(A)(B)
(C) All* & sync>0
Wn=oldest[not full(W)];
sendACK([Wn,Bmp_n])
(D) All* || last_miss_frag
& rcv_W!=cur_W & sync>0
& Full([rcv_W,Bmp_n])
Wn=oldest[not full(W)];
sendACK([Wn,Bmp_n]);sync--
ABORT-->* Uplink Only &
Inact_Timer expires
|| Attempts > MAX_ACK_REQUESTS
(E) Not All* & rcv_W!=cur_W
sync++; cur_W=rcv_W;
set[cur_W,Bmp_n(FCN)]
send Abort

Figure 40: Receiver State Machine for the ACK-on-Error Mode

Appendix D. SCHC Parameters

This section lists the information that need to be provided in the LPWAN technology-specific documents.

- Most common uses cases, deployment scenarios
- Mapping of the SCHC architectural elements onto the LPWAN architecture
- Assessment of LPWAN integrity checking
- Various potential channel conditions for the technology and the corresponding recommended use of SCHC C/D and F/R

This section lists the parameters that need to be defined in the Profile.

- Rule ID numbering scheme, fixed-sized or variable-sized Rule IDs, number of Rules, the way the Rule ID is transmitted

- Padding: size of the L2 Word (for most LPWAN technologies, this would be a byte; for some technologies, a bit)

- Decision to use SCHC fragmentation mechanism or not. If yes:

  * reliability mode(s) used, in which cases (e.g. based on link channel condition)

  * Rule ID values assigned to each mode in use

  * presence and number of bits for DTag (T) for each Rule ID value

  * support for interleaved packet transmission, to what extent

  * WINDOW_SIZE, for modes that use windows

  * number of bits for W (M) for each Rule ID value, for modes that use windows

  * number of bits for FCN (N) for each Rule ID value

  * value of MAX_WIND_FCN and use of FCN values, if applicable to the SCHC F/R mode.

  * size of MIC and algorithm for its computation, for each Rule ID, if different from the default CRC32. Byte fill-up with zeroes or other mechanism, to be specified.

  * Retransmission Timer duration for each Rule ID value, if applicable to the SCHC F/R mode

  * Inactivity Timer duration for each Rule ID value, if applicable to the SCHC F/R mode

  * MAX_ACK_REQUEST value for each Rule ID value, if applicable to the SCHC F/R mode

- if L2 Word is wider than a bit and SCHC fragmentation is used, value of the padding bits (0 or 1). This is needed because the padding bits of the last fragment are included in the MIC computation.

A Profile MAY define a delay to be added between each SCHC message transmission to respect local regulations or other constraints imposed by the applications.

- Note on soliciting downlink transmissions: In some LPWAN technologies, as part of energy-saving techniques, downlink transmission is only possible immediately after an uplink transmission. In order to avoid potentially high delay in the downlink transmission of a fragmented SCHC Packet, the SCHC Fragment receiver may want to perform an uplink transmission as soon as possible after reception of a SCHC Fragment that is not the last one. Such uplink transmission may be triggered by the L2 (e.g. an L2 ACK sent in response to a SCHC Fragment encapsulated in a L2 PDU that requires an L2 ACK) or it may be triggered from an upper layer.

- the following parameters need to be addressed in documents other than this one but not forcely in the LPWAN technology-specific documents:

  * The way the contexts are provisioned

  * The way the Rules as generated

Appendix E. Supporting multiple window sizes for fragmentation

For ACK-Always or ACK-on-Error, implementers MAY opt to support a single window size or multiple window sizes. The latter, when feasible, may provide performance optimizations. For example, a large window size SHOULD be used for packets that need to be carried by a large number of SCHC Fragments. However, when the number of SCHC Fragments required to carry a packet is low, a smaller window size, and thus a shorter Bitmap, MAY be sufficient to provide feedback on all SCHC Fragments. If multiple window sizes are supported, the Rule ID MAY be used to signal the window size in use for a specific packet transmission.

Note that the same window size MUST be used for the transmission of all SCHC Fragments that belong to the same SCHC Packet.

Appendix F. Downlink SCHC Fragment transmission

For downlink transmission of a fragmented SCHC Packet in ACK-Always mode, the SCHC Fragment receiver MAY support timer-based SCHC ACK retransmission. In this mechanism, the SCHC Fragment receiver initializes and starts a timer (the Inactivity Timer is used) after the transmission of a SCHC ACK, except when the SCHC ACK is sent in response to the last SCHC Fragment of a packet (All-1 fragment). In the latter case, the SCHC Fragment receiver does not start a timer after transmission of the SCHC ACK.

If, after transmission of a SCHC ACK that is not an All-1 fragment, and before expiration of the corresponding Inactivity timer, the SCHC Fragment receiver receives a SCHC Fragment that belongs to the current window (e.g. a missing SCHC Fragment from the current window) or to the next window, the Inactivity timer for the SCHC ACK is stopped. However, if the Inactivity timer expires, the SCHC ACK is resent and the Inactivity timer is reinitialized and restarted.

The default initial value for the Inactivity timer, as well as the maximum number of retries for a specific SCHC ACK, denoted MAX_ACK_RETRIES, are not defined in this document, and need to be defined in a Profile. The initial value of the Inactivity timer is expected to be greater than that of the Retransmission timer, in order to make sure that a (buffered) SCHC Fragment to be retransmitted can find an opportunity for that transmission.

When the SCHC Fragment sender transmits the All-1 fragment, it starts its Retransmission Timer with a large timeout value (e.g. several times that of the initial Inactivity timer). If a SCHC ACK is received before expiration of this timer, the SCHC Fragment sender retransmits any lost SCHC Fragments reported by the SCHC ACK, or if the SCHC ACK confirms successful reception of all SCHC Fragments of the last window, the transmission of the fragmented SCHC Packet is considered complete. If the timer expires, and no SCHC ACK has been received since the start of the timer, the SCHC Fragment sender assumes that the All-1 fragment has been successfully received (and possibly, the last SCHC ACK has been lost: this mechanism assumes that the retransmission timer for the All-1 fragment is long enough to allow several SCHC ACK retries if the All-1 fragment has not;been received by the SCHC Fragment receiver, and it also assumes that it is unlikely that several ACKs become all lost).

Appendix G. Note

Carles Gomez has been funded in part by the Spanish Government (Ministerio de Educacion, Cultura y Deporte) through the Jose Castillejo grant CAS15/00336, and by the ERDF and the Spanish Government through project TEC2016-79988-P. Part of his contribution to this work has been carried out during his stay as a visiting scholar at the Computer Laboratory of the University of Cambridge.

Authors' Addresses

Ana Minaburo
Acklio
1137A avenue des Champs Blancs
35510 Cesson-Sevigne Cedex
France

Email: ana@ackl.io

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

Email: Laurent.Toutain@imt-atlantique.fr

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

Email: carlesgo@entel.upc.edu

Dominique Barthel
Orange Labs
28 chemin du Vieux Chene
38243 Meylan
France

Email: dominique.barthel@orange.com

Juan Carlos Zuniga
SIGFOX
425 rue Jean Rostand
Labege 31670
France

Email: JuanCarlos.Zuniga@sigfox.com