

Internet Area  
Internet-Draft  
Intended status: Standards Track  
Expires: January 17, 2019

J. Robert  
FAU Erlangen-Nuernberg  
C. Perkins  
Futurewei  
July 16, 2018

SCHC for 802.15.4 lpwan applications  
draft-authors-lpwan-schc-802154-00

## Abstract

This document provides guidelines for creating Rules for Static Context Header Compression for IEEE 802.15.4. Since 802.15.4 provides layer-2 acknowledgements, some complexities that were designed for more general systems can be avoided.

## Status of This Memo

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

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

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

This Internet-Draft will expire on January 17, 2019.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. SCHC parameters . . . . .	4
3.1. Size of the Rule ID . . . . .	4
3.2. Use of Padding . . . . .	4
3.3. Fragmentation Delivery Reliability Option . . . . .	4
3.4. MAX_ACK_REQUEST . . . . .	4
3.5. FCN . . . . .	4
3.6. DTag . . . . .	4
3.7. L2 CRC . . . . .	5
3.8. Fragmentation ACK Parameters (not used) . . . . .	5
4. Security Considerations . . . . .	5
5. IANA Considerations . . . . .	5
6. Acknowledgements . . . . .	5
7. References . . . . .	5
7.1. Normative References . . . . .	5
7.2. Informative References . . . . .	6
Authors' Addresses . . . . .	6

## 1. Introduction

## Static Context Header Compression (SCHC)

[I-D.ietf-lpwan-ipv6-static-context-hc] is a solution for header compression, highly specialized for very predictable IPv6 packets to and from an lpwan node with significant resource constraints (especially power). This document provides guidelines for creating Rules for Static Context Header Compression (SCHC) for IEEE 802.15.4 [dot4]. Since 802.15.4 provides layer-2 acknowledgements, some complexities that were designed for more general systems can be avoided.

The Low-Power, Wide-Area IEEE 802.15.4w task group (LPWA) has been chartered to specify modifications to 802.15.4 MAC and PHY parameters that would be needed to make the technology more suitable for lpwan applications [lpwa\_par], [lpwa\_csd]. Although 801.15.4g [dot4g] and 802.15.4k [dot4k] were previously designed for such systems, recent experiments and further experience with new use cases have indicated the need for additional specification and wider applicability.

LPWA has listed different use-cases that may be relevant for LPWAN in a study group document [lpwa\_use\_cases]. The LPWAN use-cases discussed in that document are characterized as follows:

- o Focusing on uplink data
- o Typical Payload data length less than 16 bytes
- o No strict latency requirements

The LPWA also determined that it would be useful to produce a document for the IETF lp-wan Working Group to suggest parameters for the use cases. The discussion so far in LPWA has resulted in the document [lpwa\_schc].

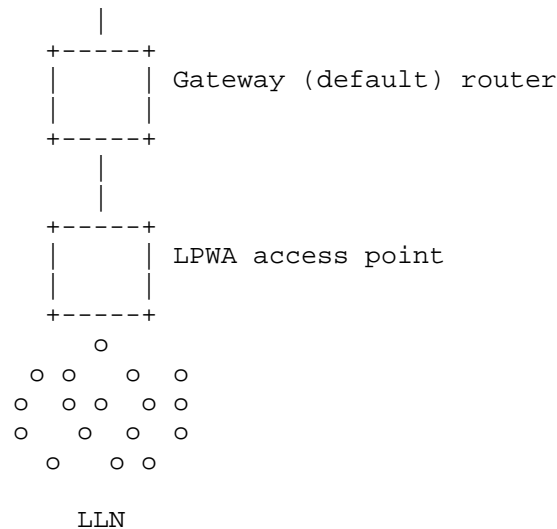


Figure 1: Representative Architecture for 802.15.4w Use Cases

A typical 802.15.4w use case is illustrated in Figure 1. The header compression context is statically configured for the transmission and reception of packets between the LPWA access point and the individual low-power devices (indicated as 'o'). Most of the rules follow the recommended practice in [I-D.ietf-lpwan-ipv6-static-context-hc] for compressing the IPv6 addresses and UDP ports; the same rulesets can be used for the possibly thousands of low-power devices, only changing the IPv6 address for the particular device relevant to the context.

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document uses the following definitions:

LPWA  
Low-Power Wide Area  
PAN

Personal Area Network  
802.15.4w  
Low-Power Wide Area (LPWA) task group within IEEE 802.15

### 3. SCHC parameters

In this section we provide details about parameter selection for a static compression context to be used over 802.15.4, according to the guidelines in [minaburo\_email]. The method by which the context is agreed upon by sender and receiver is left unspecified. For the purposes of this document, the rule-ID, rule parameters, and other uncompressed information is to be considered as a normal L2 payload that will be decompressed before delivery to L3.

#### 3.1. Size of the Rule ID

Size of the Rule ID should be 3, to allow for up to 8 rules.

#### 3.2. Use of Padding

Pad to a multiple of 8 bits in the L2 payload.

#### 3.3. Fragmentation Delivery Reliability Option

802.15.4 link acknowledgement should be used, since the static context as defined should be decompressed after delivery over a single link.

#### 3.4. MAX\_ACK\_REQUEST

MAX\_ACK\_REQUEST SHOULD be set to 3, following usual practice in 802.15.4.

#### 3.5. FCN

FCN SHOULD be set to 0, since unfragmented traffic is expected for most use cases under consideration in 802.15.4w.

#### 3.6. DTag

Similarly, DTag SHOULD be set to 0, since unfragmented traffic is expected for most use cases under consideration in 802.15.4w.

### 3.7. L2 CRC

Either CRC-16 or CRC-32 as defined in 802.15.4 could be used.

### 3.8. Fragmentation ACK Parameters (not used)

Since acknowledgments SHOULD be handled at Layer 2, no specification is made here for the following:

- o The timer duration for Fragmentation ACK Always
- o When to abort in ACK Always
- o MAX\_ATTEMPTS counter size
- o The timer size between windows in ACK On Error.
- o The inactivity timer.

## 4. Security Considerations

This document does not introduce any security mechanisms, and does not have affect existing security mechanisms or vulnerabilities already present in the base SCHC document.

## 5. IANA Considerations

This document does not specify any IANA actions.

## 6. Acknowledgements

This document has benefitted from discussions with the following people, in alphabetical order: Pat Kinney

## 7. References

### 7.1. Normative References

- [I-D.ietf-lpwan-ipv6-static-context-hc]  
Minaburo, A., Toutain, L., Gomez, C., and D. Barthel,  
"LPWAN Static Context Header Compression (SCHC) and  
fragmentation for IPv6 and UDP", draft-ietf-lpwan-ipv6-  
static-context-hc-16 (work in progress), June 2018.
- [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>>.

## 7.2. Informative References

- [dot4] P802.15, "Part 15: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", March 2012.
- [dot4g] P802.15, "Part 15: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", March 2012.
- [dot4k] P802.15, "Part 15: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", March 2012.
- [lpwa\_csd] P802.15, "Part 15: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", March 2012.
- [lpwa\_par] P802.15, "Part 15: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", March 2012.
- [lpwa\_schc] Joerg Robert, "Discussion on Suitable Parameters for SCHC", May 2018.
- [lpwa\_use\_cases] Joerg Robert, "LPWA Use-Cases", Mar 2017.
- [minaburo\_email] Ana Minaburo, "SCHC technology specific parameters", Feb 2018.

## Authors' Addresses

Joerg Robert  
Friedrich-Alexander Universitaet Erlangen-Nuernberg  
Am Wolfsmantel 33  
Erlangen 91058  
Germany

Phone: +49-9131-85-25373  
Email: joerg.robert@fau.de

Charles E. Perkins  
Futurewei Inc.  
2330 Central Expressway  
Santa Clara, CA 95050  
USA

Phone: +1-408-330-4586  
Email: charliep@computer.org

lpwan Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 3, 2019

A. Minaburo  
Acklio  
L. Toutain  
Institut MINES TELECOM; IMT Atlantique  
R. Andreasen  
Universidad de Buenos Aires  
July 02, 2018

LPWAN Static Context Header Compression (SCHC) for CoAP  
draft-ietf-lpwan-coap-static-context-hc-04

Abstract

This draft defines the way SCHC header compression can be applied to CoAP headers. CoAP header structure differs from IPv6 and UDP protocols since the CoAP use a flexible header with a variable number of options themselves of a variable length. Another important difference is the asymmetry in the header format used in request and response messages. Most of the compression mechanisms have been introduced in [I-D.ietf-lpwan-ipv6-static-context-hc], this document explains how to use the SCHC compression for CoAP.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents



(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. SCHC Compression Process . . . . .	3
3. CoAP Compression with SCHC . . . . .	4
4. Compression of CoAP header fields . . . . .	6
4.1. CoAP version field . . . . .	6
4.2. CoAP type field . . . . .	6
4.3. CoAP code field . . . . .	6
4.4. CoAP Message ID field . . . . .	6
4.5. CoAP Token fields . . . . .	7
5. CoAP options . . . . .	7
5.1. CoAP Content and Accept options. . . . .	7
5.2. CoAP option Max-Age field, CoAP option Uri-Host and Uri-Port fields . . . . .	7
5.3. CoAP option Uri-Path and Uri-Query fields . . . . .	8
5.3.1. Variable length Uri-Path and Uri-Query . . . . .	8
5.3.2. Variable number of path or query elements . . . . .	9
5.4. CoAP option Size1, Size2, Proxy-URI and Proxy-Scheme fields . . . . .	9
5.5. CoAP option ETag, If-Match, If-None-Match, Location-Path and Location-Query fields . . . . .	9
6. Other RFCs . . . . .	9
6.1. Block . . . . .	9
6.2. Observe . . . . .	10
6.3. No-Response . . . . .	10
6.4. Time Scale . . . . .	10
6.5. OSCORE . . . . .	10
7. Examples of CoAP header compression . . . . .	12
7.1. Mandatory header with CON message . . . . .	12
7.2. Complete exchange . . . . .	13
7.3. OSCORE Compression . . . . .	14
7.4. Example OSCORE Compression . . . . .	17
8. Normative References . . . . .	22
Authors' Addresses . . . . .	22

## 1. Introduction

CoAP [rfc7252] is an implementation of the REST architecture for constrained devices. Nevertheless, if limited, the size of a CoAP header may be too large for LPWAN constraints and some compression may be needed to reduce the header size.

[I-D.ietf-lpwan-ipv6-static-context-hc] defines a header compression mechanism for LPWAN network based on a static context. The context is said static since the field description composing the Rules and the context are not learned during the packet exchanges but are previously defined. The context(s) is(are) known by both ends before transmission.

A context is composed of a set of rules that are referenced by Rule IDs (identifiers). A rule contains an ordered list of the fields descriptions containing a field ID (FID), its length (FL) and its position (FP), a direction indicator (DI) (upstream, downstream and bidirectional) and some associated Target Values (TV). Target Value indicates the value that can be expected. TV can also be a list of values. A Matching Operator (MO) is associated to each header field description. The rule is selected if all the MOs fit the TVs for all fields. In that case, a Compression/Decompression Action (CDA) associated to each field defines the link between the compressed and decompressed value for each of the header fields. Compression results mainly in 4 actions: send the field value, send nothing, send less significant bits of a field, send an index. Values sent are called Compression Residues and follows the rule ID.

## 2. SCHC Compression Process

The SCHC Compression rules can be applied to CoAP flows. SCHC Compression of the CoAP header may be done in conjunction with the above layers (IPv6/UDP) or independently. The SCHC adaptation layers as described in [I-D.ietf-lpwan-ipv6-static-context-hc] may be used as shown in the Figure 1.

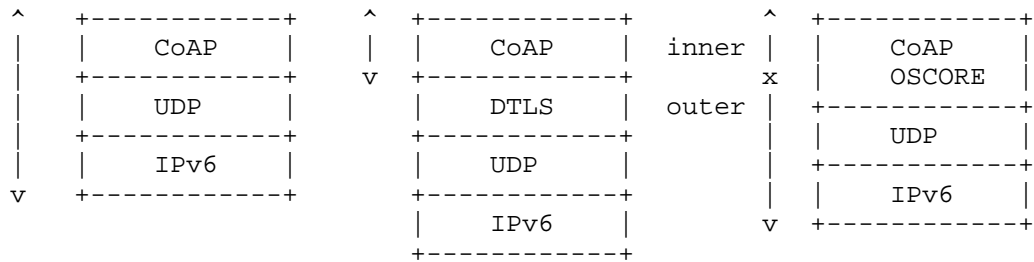


Figure 1: rule scope for CoAP

Figure 1 shows some examples for CoAP architecture and the SCHC rule's scope. A rule can covers all headers from IPv6 to CoAP, SCHC C/D is done in the device and at the LPWAN boundary. If an end-to-end encryption mechanisms is used between the device and the application. CoAP must be compressed independently of the other layers. The rule ID and the compression residue are encrypted using a mechanism such as DTLS. Only the other end can decipher the information.

Layers below may also be compressed using other SCHC rules (this is out of the scope of this document). OSCORE

[I-D.ietf-core-object-security] can also define 2 rules to compress the CoAP message. A first rule focuses on the inner header and is end to end, a second rule may compress the outer header and the layer above. SCHC C/D for inner header is done by both ends, SCHC C/D for outer header and other headers is done between the device and the LPWAN boundary.

### 3. CoAP Compression with SCHC

CoAP differs from IPv6 and UDP protocols on the following aspects:

- o IPv6 and UDP are symmetrical protocols. The same fields are found in the request and in the response, only the location in the header may vary (e.g. source and destination fields). A CoAP request is different from a response. For example, the URI-path option is mandatory in the request and is not found in the response, a request may contain an Accept option and the response a Content option.

[I-D.ietf-lpwan-ipv6-static-context-hc] defines the use of a message direction (DI) when processing the rule which allows the description of message header format in both directions.

- o Even when a field is "symmetric" (i.e. found in both directions) the values carried in each direction are different. Combined with a matching list in the TV, this will allow to reduce the range of expected values in a particular direction and therefore reduce the size of a compression residue. For instance, if a client sends only CON request, the type can be elided by compression and the answer may use one bit to carry either the ACK or RST type. Same behavior can be applied to the CoAP Code field (0.0X code are present in the request and Y.ZZ in the answer). The direction allows to split in two parts the possible values for each direction.
- o In IPv6 and UDP header fields have a fixed size. In CoAP, Token size may vary from 0 to 8 bytes, length is given by a field in the header. More systematically, the CoAP options are described using the Type-Length-Value.

[I-D.ietf-lpwan-ipv6-static-context-hc] offers the possibility to define a function for the Field Length in the Field Description.

- o In CoAP headers, a field can be duplicated several times, for instances, elements of an URI (path or queries). The position defined in a rule, associated to a Field ID, can be used to identify the proper element.

[I-D.ietf-lpwan-ipv6-static-context-hc] allows a Field id to appears several times in the rule, the Field Position (FP) removes ambiguities for the matching operation.

- o Field size defined in the CoAP protocol can be too large regarding LPWAN traffic constraints. This is particularly true for the message ID field or Token field. The use of MSB MO can be used to reduce the information carried on LPWANs.
- o CoAP also obeys to the client/server paradigm and the compression rate can be different if the request is issued from an LPWAN node or from a non LPWAN device. For instance a Device (Dev) aware of LPWAN constraints can generate a 1 byte token, but a regular CoAP client will certainly send a larger token to the Thing. SCHC compression will not modify the values to offer a better compression rate. Nevertheless a proxy placed before the compressor may change some field values to offer a better compression rate and maintain the necessary context for interoperability with existing CoAP implementations.

#### 4. Compression of CoAP header fields

This section discusses of the compression of the different CoAP header fields.

##### 4.1. CoAP version field

This field is bidirectional and must be elided during the SCHC compression, since it always contains the same value. In the future, if new version of CoAP are defined, new rules ID will be defined avoiding ambiguities between versions.

##### 4.2. CoAP type field

[rfc7252] defines 4 types of messages: CON, NON, ACK and RST. The latter two ones are a response of the two first ones. If the device plays a specific role, a rule can exploit these property with the mapping list: [CON, NON] for one direction and [ACK, RST] for the other direction. Compression residue is reduced to 1 bit.

The field must be elided if for instance a client is sending only NON or CON messages.

In any case, a rule must be defined to carry RST to a client.

##### 4.3. CoAP code field

The compression of the CoAP code field follows the same principle as for the CoAP type field. If the device plays a specific role, the set of code values can be split in two parts, the request codes with the 0 class and the response values.

If the device implement only a CoAP client, the request code can be reduced to the set of request the client is able to process.

All the response codes should be compressed with a SCHC rule.

##### 4.4. CoAP Message ID field

This field is bidirectional and is used to manage acknowledgments. Server memorizes the value for a EXCHANGE\_LIFETIME period (by default 247 seconds) for CON messages and a NON\_LIFETIME period (by default 145 seconds) for NON messages. During that period, a server receiving the same Message ID value will process the message has a retransmission. After this period, it will be processed as a new messages.

In case the Device is a client, the size of the message ID field may be too large regarding the number of messages sent. Client may use only small message ID values, for instance 4 bit long. Therefore a MSB can be used to limit the size of the compression residue.

In case the Device is a server, client may be located outside of the LPWAN area and view the device as a regular device connected to the internet. The client will generate Message ID using the 16 bits space offered by this field. A CoAP proxy can be set before the SCHC C/D to reduce the value of the Message ID, to allow its compression with the MSB matching operator and LSB CDA.

#### 4.5. CoAP Token fields

Token is defined through two CoAP fields, Token Length in the mandatory header and Token Value directly following the mandatory CoAP header.

Token Length is processed as a tradition protocol field. If the value remains the same during all the transaction, the size can be stored in the context and elided during the transmission. Otherwise it will have to be sent as a compression residue.

Token Value size should not be defined directly in the rule in the Field Length (FL). Instead a specific function designed as "TKL" must be used. This function informs the SCHC C/D that the length of this field has to be read from the Token Length field.

#### 5. CoAP options

##### 5.1. CoAP Content and Accept options.

These fields are both unidirectional and must not be set to bidirectional in a rule entry.

If single value is expected by the client, it can be stored in the TV and elided during the transmission. Otherwise, if several possible values are expected by the client, a matching-list should be used to limit the size of the residue. If not the possible, the value has to be sent as a residue (fixed or variable length).

##### 5.2. CoAP option Max-Age field, CoAP option Uri-Host and Uri-Port fields

This field is unidirectional and must not be set to bidirectional in a rule entry. It is used only by the server to inform of the caching duration and is never found in client requests.

If the duration is known by both ends, value can be elided on the LPWAN.

A matching list can be used if some well-known values are defined.

Otherwise these options should be sent as a residue (fixed or variable length).

### 5.3. CoAP option Uri-Path and Uri-Query fields

This fields are unidirectional and must not be set to bidirectional in a rule entry. They are used only by the client to access to a specific resource and are never found in server responses.

Uri-Path and Uri-Query elements are a repeatable options, the Field Position (FP) gives the position in the path.

A Mapping list can be used to reduce size of variable Paths or Queries. In that case, to optimize the compression, several elements can be regrouped into a single entry. Numbering of elements do not change, MO comparison is set with the first element of the matching.

FID	FL	FP	DI	TV	MO	CDA
URI-Path	1	up		["/a/b", "/c/d"]	equal	not-sent
URI-Path	3	up			ignore	value-sent

Figure 2: complex path example

In Figure 2 a single bit residue can be used to code one of the 2 paths. If regrouping was not allowed, a 2 bits residue whould have been needed.

#### 5.3.1. Variable length Uri-Path and Uri-Query

When the length is known at the rule creation, the Field Length must be set to variable, and the unit is set to bytes.

The MSB MO can be apply to a Uri-Path or Uri-Query element. Since MSB value is given in bit, the size must always be a multiple of 8 bits and the LSB CDA must not carry any value.

The length sent at the beginning of a variable length residue indicates the size of the LSB in bytes.

For instance for a CoMi path /c/X6?k="eth0" the rule can be set to:

FID	FL	FP	DI	TV	MO	CDA
URI-Path	1	up	"c"	equal	not-sent	
URI-Path	2	up		ignore	value-sent	
URI-Query	1	up	"k="	MSB (16)	LSB	

Figure 3: CoMi URI compression

Figure 3 shows the parsing and the compression of the URI. where c is not sent. The second element is sent with the length (i.e. 0x2 X 6) followed by the query option (i.e. 0x05 "eth0").

#### 5.3.2. Variable number of path or query elements

The number of Uri-path or Uri-Query element in a rule is fixed at the rule creation time. If the number varies, several rules should be created to cover all the possibilities. Another possibilities is to define the length of Uri-Path to variable and send a compression residue with a length of 0 to indicate that this Uri-Path is empty. This add 4 bits to the compression residue.

#### 5.4. CoAP option Size1, Size2, Proxy-URI and Proxy-Scheme fields

These fields are unidirectional and must not be set to bidirectional in a rule entry. They are used only by the client to access to a specific resource and are never found in server response.

If the field value must be sent, TV is not set, MO is set to "ignore" and CDF is set to "value-sent". A mapping can also be used.

Otherwise the TV is set to the value, MO is set to "equal" and CDF is set to "not-sent"

#### 5.5. CoAP option ETag, If-Match, If-None-Match, Location-Path and Location-Query fields

These fields are unidirectional.

These fields values cannot be stored in a rule entry. They must always be sent with the compression residues.

### 6. Other RFCs

#### 6.1. Block

Block [rfc7959] allows a fragmentation at the CoAP level. SCHC includes also a fragmentation protocol. They are compatible. If a block option is used, its content must be sent as a compression residue.



## 6.2. Observe

[rfc7641] defines the Observe option. The TV is not set, MO is set to "ignore" and the CDF is set to "value-sent". SCHC does not limit the maximum size for this option (3 bytes). To reduce the transmission size either the device implementation should limit the value increase or a proxy can modify the incrementation.

Since RST message may be sent to inform a server that the client do not require Observe response, a rule must allow the transmission of this message.

## 6.3. No-Response

[rfc7967] defines an No-Response option limiting the responses made by a server to a request. If the value is not known by both ends, then TV is set to this value, MO is set to "equal" and CDF is set to "not-sent".

Otherwise, if the value is changing over time, TV is not set, MO is set to "ignore" and CDA to "value-sent". A matching list can also be used to reduce the size.

## 6.4. Time Scale

Time scale [I-D.toutain-core-time-scale] option allows a client to inform the server that it is in a slow network and that message ID should be kept for a duration given by the option.

If the value is not known by both ends, then TV is set to this value, MO is set to "equal" and CDA is set to "not-sent".

Otherwise, if the value is changing over time, TV is not set, MO is set to "ignore" and CDA to "value-sent". A matching list can also be used to reduce the size.

## 6.5. OSCORE

OSCORE [I-D.ietf-core-object-security] defines end-to-end protection for CoAP messages. This section describes how SCHC rules can be applied to compress OSCORE-protected messages.

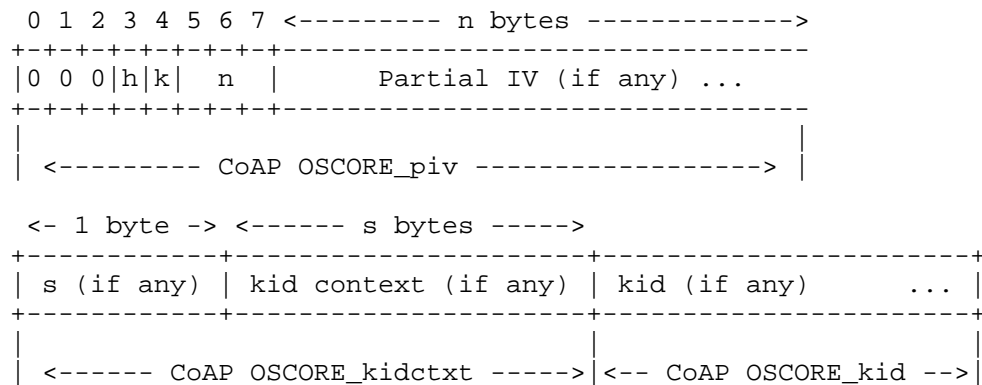


Figure 4: OSCORE Option

The encoding of the OSCORE Option Value defined in Section 6.1 of [I-D.ietf-core-object-security] is repeated in Figure 4.

The first byte is used for flags that specify the contents of the OSCORE option. The 3 most significant bits are reserved and always set to 0. Bit h, when set, indicates the presence of the kid context field in the option. Bit k, when set, indicates the presence of a kid field. The 3 least significant bits n indicate to length of the piv field in bytes, n = 0 taken to mean that no piv is present.

After the flag byte follow the piv field, kid context field and kid field in order and if present; the length of the kid context field is encoded in the first byte denoting by s the length of the kid context in bytes.

This draft recommends to implement a parser that is able to identify the OSCORE Option and the fields it contains - this makes it possible to do a preliminary processing of the message in preparation for regular SCHC compression.

Conceptually, the OSCORE option can transmit up to 3 distinct pieces of information at a time: the piv, the kid context, and the kid. This draft proposes that the SCHC Parser split the contents of this option into 3 SCHC fields:

- o CoAP OSCORE\_piv,
- o CoAP OSCORE\_ctxt,
- o CoAP OSCORE\_kid.

These fields are superposed on the OSCORE Option format in Figure 4, and include the corresponding flag and size bits for each part of the option. Both the flag and size bits can be omitted by use of the MSB matching operator on each field.

## 7. Examples of CoAP header compression

### 7.1. Mandatory header with CON message

In this first scenario, the LPWAN compressor receives from outside client a POST message, which is immediately acknowledged by the Device. For this simple scenario, the rules are described Figure 5.

Rule ID 1

Field	FL	FP	DI	Target Value	Match Opera.	CDA	Sent [bits]
CoAP version			bi	01	equal	not-sent	
CoAP version			bi	01	equal	not-sent	
CoAP Type			dw	CON	equal	not-sent	
CoAP Type			up	[ACK, RST]	match-map	matching-sent	T
CoAP TKL			bi	0	equal	not-sent	
CoAP Code			bi	ML1	match-map	matching-sent	CC CCC
CoAP MID			bi	0000	MSB(7 )	LSB(9)	M-ID
CoAP Uri-Path			dw	path	equal 1	not-sent	

Figure 5: CoAP Context to compress header without token

The version and Token Length fields are elided. Code has shrunk to 5 bits using a matching list. Uri-Path contains a single element indicated in the matching operator.

Figure 6 shows the time diagram of the exchange. A client in the Application Server sends a CON request. It can go through a proxy which reduces the message ID to a smallest value, with at least the 9 most significant bits equal to 0. SCHC Compression reduces the header sending only the Type, a mapped code and the least 9 significant bits of Message ID.

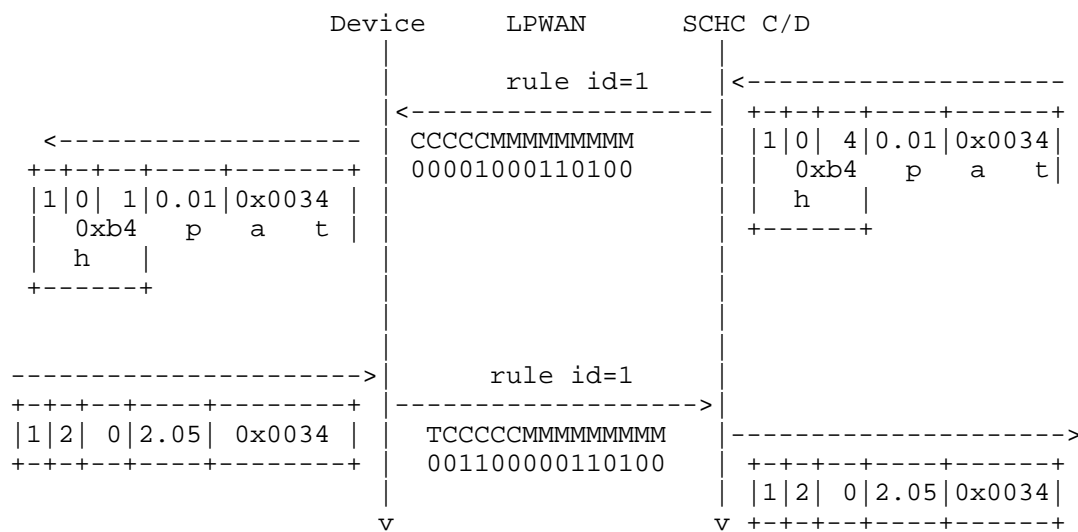
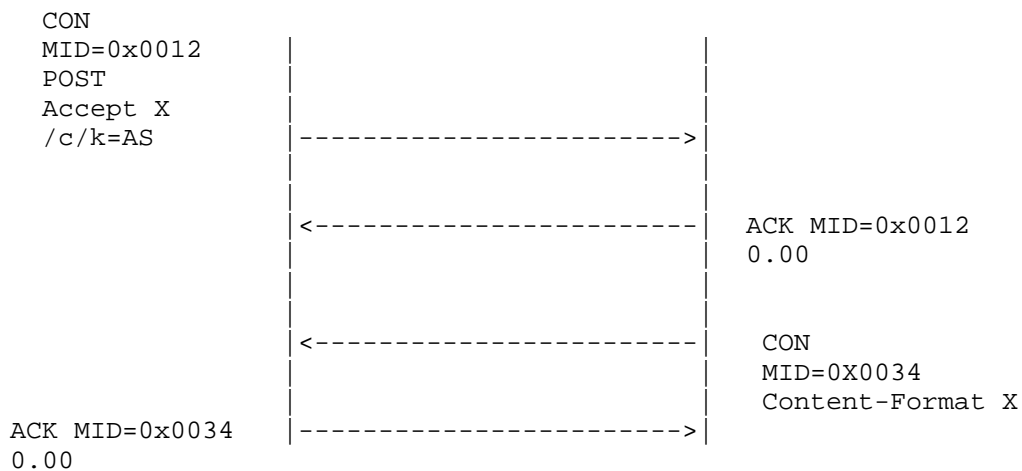


Figure 6: Compression with global addresses

## 7.2. Complete exchange

In that example, the Thing is using CoMi and sends queries for 2 SID.



### 7.3. OSCORE Compression

OSCORE aims to solve the problem of end-to-end encryption for CoAP messages, which are otherwise required to terminate their TLS or DTLS protection at the proxy, as discussed in Section 11.2 of [rfc7252]. CoAP proxies are men-in-the-middle, but not all of the information they have access to is necessary for their operation. The goal, therefore, is to hide as much of the message as possible while still enabling proxy operation.

Conceptually this is achieved by splitting the CoAP message into an Inner Plaintext and Outer OSCORE Message. The Inner Plaintext contains sensible information which is not necessary for proxy operation. This, in turn, is the part of the message which can be encrypted and need not be decrypted until it reaches its end destination. The Outer Message acts as a shell matching the format of a regular CoAP message, and includes all Options and information needed for proxy operation and caching. This decomposition is illustrated in Figure 7.

CoAP options are sorted into one of 3 classes, each granted a specific type of protection by the protocol:

- o Class E: Encrypted options moved to the Inner Plaintext,
- o Class I: Integrity-protected options included in the AAD for the encryption of the Plaintext but otherwise left untouched in the Outer Message,
- o Class U: Unprotected options left untouched in the Outer Message.

Additionally, the OSCORE Option is added as an Outer option, signaling that the message is OSCORE protected. This option carries the information necessary to retrieve the Security Context with which the message was encrypted so that it may be correctly decrypted at the other end-point.



Figure 7: OSCORE inner and outer header form a CoAP message

Figure 7 shows the message format for the OSCORE Message and Plaintext. In the Outer Header, the original message code is hidden and replaced by a default value (POST or FETCH) depending on whether the original message was a Request or a Response. The original message code is put into the first byte of the Plaintext. Following the message code come the class E options and if present the original message Payload preceded by its payload marker.

The Plaintext is now encrypted by an AEAD algorithm which integrity protects Security Context parameters and eventually any class I options from the Outer Header. Currently no CoAP options are marked

class I. The resulting Ciphertext becomes the new Payload of the OSCORE message, as illustrated in Figure 8.

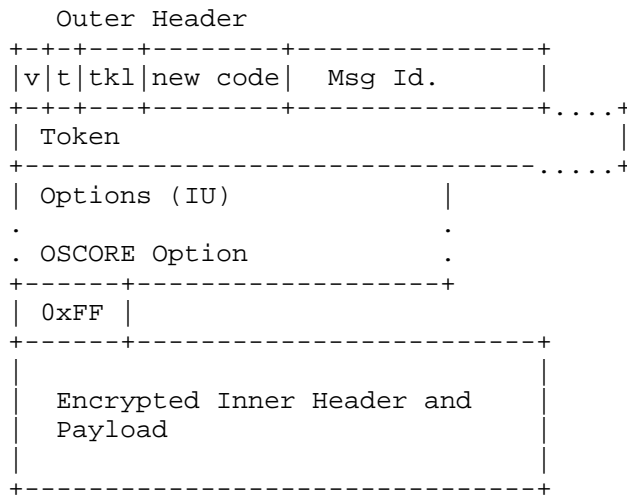


Figure 8: OSCORE message

The SCHC Compression scheme consists of compressing both the Plaintext before encryption and the resulting OSCORE message after encryption, see Figure 9. This way compression reaches all fields of the original CoAP message.

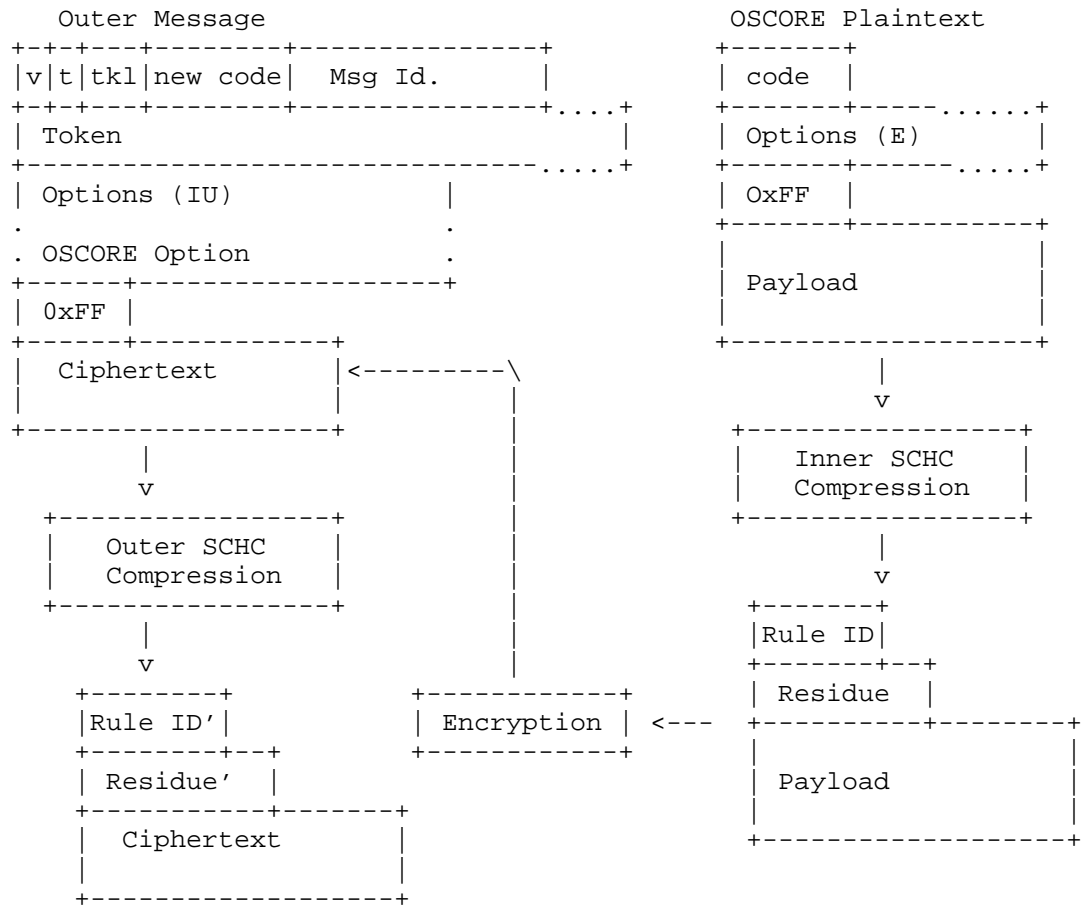


Figure 9: OSCORE Compression Diagram

#### 7.4. Example OSCORE Compression

In what follows we present an example GET Request and consequent CONTENT Response and show a possible set of rules for the Inner and Outer SCHC Compression. We then show a dump of the results and contrast SCHC + OSCORE performance with SCHC + COAP performance. This gives an approximation to the cost of security with SCHC-OSCORE.

Our first example CoAP message is the GET Request in Figure 10



```
Original message:
=====
0x4101000182bb74656d7065726174757265

Header:
0x4101
01   Ver
    00   CON
        0001   tk1
            00000001   Request Code 1 "GET"

0x0001 = mid
0x82 = token

Options:
0xbb74656d7065726174757265
Option 11: URI_PATH
Value = temperature

Original msg length:  17 bytes.
```

Figure 10: CoAP GET Request

Its corresponding response is the CONTENT Response in Figure 11.

```
Original message:
=====
0x6145000182ff32332043

Header:
0x6145
01   Ver
    10   ACK
        0001   tk1
            01000101   Successful Response Code 69 "2.05 Content"

0x0001 = mid
0x82 = token

0xFF Payload marker
Payload:
0x32332043

Original msg length:  10
```

Figure 11: CoAP CONTENT Response

The SCHC Rules for the Inner Compression include all fields that are already present in a regular CoAP message, what matters is the order of appearance and inclusion of only those CoAP fields that go into the Plaintext, Figure 12.

Rule ID 0

Field	FP	DI	Target Value	MO	CDA	Sent [bits]
CoAP Code		up	1	equal	not-sent	
CoAP Code		dw	[69,132]	match-map	match-sent	c
CoAP Uri-Path		up	temperature	equal	not-sent	
CoAP Option-End		dw	0xFF	equal	not-sent	

Figure 12: Inner SCHC Rules

The Outer SCHC Rules (Figure 13) must process the OSCORE Options fields. Here we mask off the repeated bits (most importantly the flag and size bits) with the MSB Mathing Operator.

Rule ID 0

Field	FP	DI	Target Value	MO	CDA	Sent [bits]
CoAP version		bi	01	equal	not-sent	
CoAP Type		up	0	equal	not-sent	
CoAP Type		dw	2	equal	not-sent	
CoAP TKL		bi	1	equal	not-sent	
CoAP Code		up	2	equal	not-sent	
CoAP Code		dw	68	equal	not-sent	
CoAP MID		bi	0000	MSB(12)	LSB	MMMM
CoAP Token		bi	0x80	MSB(5)	LSB	TTT
CoAP OSCORE_piv		up	0x0900	MSB(12)	LSB	PPPP
CoAP OSCORE_kid		up	b'\x06client'	MSB(52)	LSB	KKKK
CoAP OSCORE_piv		dw	b''	equal	not-sent	
CoAP Option-End		dw	0xFF	equal	not-sent	

Figure 13: Outer SCHC Rules

Next we show a dump of the compressed message:

```
Compressed message:
=====
0x00291287f0a5c4833760d170
0x00 = Rule ID

piv = 0x04

Compression residue:
0b0001 010 0100 0100 (15 bits -> 2 bytes with padding)
  mid  tkn  piv   kid

Payload
0xa1fc297120cdd8345c

Compressed message length: 12 bytes
```

Figure 14: SCHC-OSCORE Compressed GET Request

```
Compressed message:
=====
0x0015f4de9cb814c96aed9b1d981a3a58
0x00 = Rule ID

Compression residue:
0b0001 010  (7 bits -> 1 byte with padding)
  mid  tkn

Payload
0xfa6f4e5c0a64b576cd8ecc0d1d2c

Compressed msg length: 16 bytes
```

Figure 15: SCHC-OSCORE Compressed CONTENT Response

For contrast, we compare these results with what would be obtained by SCHC compressing the original CoAP messages without protecting them with OSCORE. To do this, we compress the CoAP messages according to the SCHC rules in Figure 16.

Rule ID 1

Field	FP	DI	Target Value	MO	CDA	Sent [bits]
CoAP version		bi	01	equal	not-sent	
CoAP Type		up	0	equal	not-sent	
CoAP Type		dw	2	equal	not-sent	
CoAP TKL		bi	1	equal	not-sent	
CoAP Code		up	2	equal	not-sent	
CoAP Code		dw	[69,132]	equal	not-sent	
CoAP MID		bi	0000	MSB(12)	LSB	MMMM
CoAP Token		bi	0x80	MSB(5)	LSB	TTT
CoAP Uri-Path		up	temperature	equal	not-sent	
COAP Option-End		dw	0xFF	equal	not-sent	

Figure 16: SCHC-CoAP Rules (No OSCORE)

This yields the results in Figure 17 for the Request, and Figure 18 for the Response.

Compressed message:

=====

0x0114

0x01 = Rule ID

Compression residue:

0b00010100 (1 byte)

Compressed msg length: 2

Figure 17: CoAP GET Compressed without OSCORE

Compressed message:

=====

0x010a32332043

0x01 = Rule ID

Compression residue:

0b00001010 (1 byte)

Payload

0x32332043

Compressed msg length: 6

Figure 18: CoAP CONTENT Compressed without OSCORE

As can be seen, the difference between applying SCHC + OSCORE as compared to regular SCHC + COAP is about 10 bytes of cost.

## 8. Normative References

- [I-D.ietf-core-object-security]  
Selander, G., Mattsson, J., Palombini, F., and L. Seitz,  
"Object Security for Constrained RESTful Environments  
(OSCORE)", draft-ietf-core-object-security-13 (work in  
progress), June 2018.
- [I-D.ietf-lpwan-ipv6-static-context-hc]  
Minaburo, A., Toutain, L., Gomez, C., and D. Barthel,  
"LPWAN Static Context Header Compression (SCHC) and  
fragmentation for IPv6 and UDP", draft-ietf-lpwan-ipv6-  
static-context-hc-16 (work in progress), June 2018.
- [I-D.toutain-core-time-scale]  
Minaburo, A. and L. Toutain, "CoAP Time Scale Option",  
draft-toutain-core-time-scale-00 (work in progress),  
October 2017.
- [rfc7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained  
Application Protocol (CoAP)", RFC 7252,  
DOI 10.17487/RFC7252, June 2014,  
<<https://www.rfc-editor.org/info/rfc7252>>.
- [rfc7641] Hartke, K., "Observing Resources in the Constrained  
Application Protocol (CoAP)", RFC 7641,  
DOI 10.17487/RFC7641, September 2015,  
<<https://www.rfc-editor.org/info/rfc7641>>.
- [rfc7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in  
the Constrained Application Protocol (CoAP)", RFC 7959,  
DOI 10.17487/RFC7959, August 2016,  
<<https://www.rfc-editor.org/info/rfc7959>>.
- [rfc7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T.  
Bose, "Constrained Application Protocol (CoAP) Option for  
No Server Response", RFC 7967, DOI 10.17487/RFC7967,  
August 2016, <<https://www.rfc-editor.org/info/rfc7967>>.

## Authors' Addresses

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

Email: ana@ackl.io

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

Email: Laurent.Toutain@imt-atlantique.fr

Ricardo Andreasen  
Universidad de Buenos Aires  
Av. Paseo Colon 850  
C1063ACV Ciudad Autonoma de Buenos Aires  
Argentina

Email: randreasen@fi.uba.ar

lpwan Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: December 31, 2018

A. Minaburo  
Acklio  
L. Toutain  
IMT-Atlantique  
C. Gomez  
Universitat Politecnica de Catalunya  
D. Barthel  
Orange Labs  
June 29, 2018

LPWAN Static Context Header Compression (SCHC) and fragmentation for  
IPv6 and UDP  
draft-ietf-lpwan-ipv6-static-context-hc-16

Abstract

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

SCHC compression is based on a common static context stored in both the LPWAN devices 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 two maximum payload size.

The SCHC header compression and fragmentation mechanisms are independent of the specific LPWAN technology over which they are used. Note that this document defines generic functionalities and advisedly offers flexibility with regard to parameter settings and mechanism choices. Such settings and choices are expected to be made in other technology-specific documents.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute

working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on December 31, 2018.

#### Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1. Introduction . . . . .	4
2. Requirements Notation . . . . .	5
3. LPWAN Architecture . . . . .	5
4. Terminology . . . . .	6
5. SCHC overview . . . . .	9
6. Rule ID . . . . .	13
7. Static Context Header Compression . . . . .	13
7.1. SCHC C/D Rules . . . . .	14
7.2. Rule ID for SCHC C/D . . . . .	16
7.3. Packet processing . . . . .	16
7.4. Matching operators . . . . .	18
7.5. Compression Decompression Actions (CDA) . . . . .	18
7.5.1. not-sent CDA . . . . .	20
7.5.2. value-sent CDA . . . . .	20
7.5.3. mapping-sent CDA . . . . .	20
7.5.4. LSB CDA . . . . .	20
7.5.5. DevIID, AppIID CDA . . . . .	21
7.5.6. Compute-* . . . . .	21
8. Fragmentation . . . . .	21
8.1. Overview . . . . .	21
8.2. Fragmentation Tools . . . . .	22



8.3.	Reliability modes . . . . .	25
8.4.	Fragmentation Formats . . . . .	27
8.4.1.	Fragments that are not the last one . . . . .	27
8.4.2.	All-1 fragment . . . . .	29
8.4.3.	SCHC ACK format . . . . .	31
8.4.4.	Abort formats . . . . .	33
8.5.	Baseline mechanism . . . . .	35
8.5.1.	No-ACK . . . . .	36
8.5.2.	ACK-Always . . . . .	36
8.5.3.	ACK-on-Error . . . . .	39
8.6.	Supporting multiple window sizes . . . . .	40
8.7.	Downlink SCHC Fragment transmission . . . . .	41
9.	Padding management . . . . .	42
10.	SCHC Compression for IPv6 and UDP headers . . . . .	43
10.1.	IPv6 version field . . . . .	43
10.2.	IPv6 Traffic class field . . . . .	43
10.3.	Flow label field . . . . .	44
10.4.	Payload Length field . . . . .	44
10.5.	Next Header field . . . . .	44
10.6.	Hop Limit field . . . . .	45
10.7.	IPv6 addresses fields . . . . .	45
10.7.1.	IPv6 source and destination prefixes . . . . .	45
10.7.2.	IPv6 source and destination IID . . . . .	46
10.8.	IPv6 extensions . . . . .	46
10.9.	UDP source and destination port . . . . .	46
10.10.	UDP length field . . . . .	47
10.11.	UDP Checksum field . . . . .	47
11.	IANA Considerations . . . . .	48
12.	Security considerations . . . . .	48
12.1.	Security considerations for SCHC Compression/Decompression . . . . .	48
12.2.	Security considerations for SCHC Fragmentation/Reassembly . . . . .	48
13.	Acknowledgements . . . . .	49
14.	References . . . . .	50
14.1.	Normative References . . . . .	50
14.2.	Informative References . . . . .	50
Appendix A.	SCHC Compression Examples . . . . .	51
Appendix B.	Fragmentation Examples . . . . .	54
Appendix C.	Fragmentation State Machines . . . . .	60
Appendix D.	SCHC Parameters - Ticket #15 . . . . .	67
Appendix E.	Note . . . . .	68
Authors' Addresses	. . . . .	69

## 1. Introduction

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

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

- o The network topology is star-oriented, which means that all packets 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 Network Gateways (NGW).
- o Because devices embed built-in applications, the traffic flows to be compressed are known in advance. Indeed, new applications cannot be easily installed in LPWAN devices, as they would in computers or smartphones.

The Static Context Header Compression (SCHC) is defined for this environment. SCHC 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, that would be incompatible with LPWAN characteristics. 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, among others, by a very reduced data unit and/or payload size (see [RFC8376]). However, some of these technologies do not provide fragmentation functionality, therefore the only option for them to support the IPv6 MTU requirement of 1280 bytes [RFC8200] is to use a fragmentation protocol at the adaptation layer, below IPv6. In response to this need, this document also defines a fragmentation/reassembly mechanism, which supports the IPv6 MTU requirement over LPWAN technologies. Such functionality has been designed under the assumption that there is no out-of-sequence delivery of data units between the entity performing fragmentation and the entity performing reassembly.

Note that this document defines generic functionality and purposefully offers flexibility with regard to parameter settings and mechanism choices. Such settings and choices are expected to be made in other, technology-specific 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)

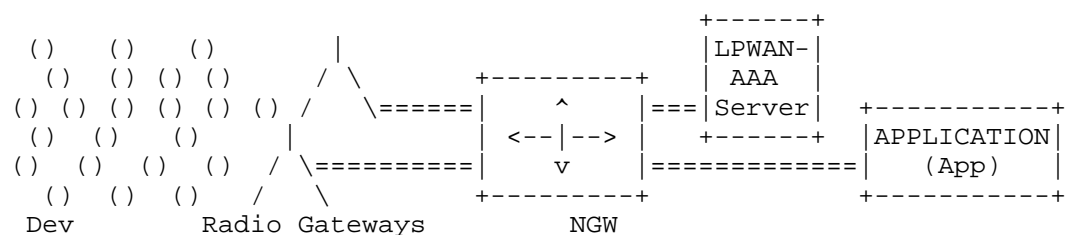


Figure 1: LPWAN Architecture

#### 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".

- o Abort. A SCHC Fragment format to signal the other end-point that the on-going fragment transmission is stopped and finished.
- o All-0. The SCHC Fragment format for the last fragment of a window that is not the last one of a SCHC Packet (see window in this glossary).
- o All-1. The SCHC Fragment format for the last fragment of the SCHC Packet.
- o All-0 empty. An All-0 SCHC Fragment without payload. It is used to request the SCHC ACK with the encoded Bitmap when the Retransmission Timer expires, in a window that is not the last one of a packet.
- o All-1 empty. An All-1 SCHC Fragment without payload. It is used to request the SCHC ACK with the encoded Bitmap when the Retransmission Timer expires in the last window of a packet.
- o App: LPWAN Application. An application sending/receiving IPv6 packets to/from the Device.
- o AppIID: Application Interface Identifier. The IID that identifies the application server interface.
- o Bi: Bidirectional. Characterises a Rule Entry that applies to headers of packets travelling in either direction (Up and Dw, see this glossary).
- o Bitmap: a bit field in the SCHC ACK message that tells the sender which SCHC Fragments in a window of fragments were correctly received.
- o C: Checked bit. Used in an acknowledgement (SCHC ACK) header to determine if the MIC locally computed by the receiver matches (1) the received MIC or not (0).
- o 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.

- o Compression Residue. The bits that need to be sent (beyond the Rule ID itself) after applying the SCHC compression over each header field.
- o Context: A set of Rules used to compress/decompress headers.
- o Dev: Device. A node connected to an LPWAN. A Dev SHOULD implement SCHC.
- o DevIID: Device Interface Identifier. The IID that identifies the Dev interface.
- o DI: Direction Indicator. This field tells which direction of packet travel (Up, Dw or Bi) a Rule applies to. This allows for asymmetric processing.
- o DTag: Datagram Tag. This SCHC F/R header field is set to the same value for all SCHC Fragments carrying the same SCHC Packet.
- o Dw: Downlink direction for compression/decompression in both sides, from SCHC C/D in the network to SCHC C/D in the Dev.
- o FCN: Fragment Compressed Number. This SCHC F/R header field carries an efficient representation of a larger-sized fragment number.
- o Field Description. A line in the Rule table.
- o FID: Field Identifier. This is an index to describe the header fields in a Rule.
- o 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.
- o FP: Field Position is a value that is used to identify the position where each instance of a field appears in the header.
- o IID: Interface Identifier. See the IPv6 addressing architecture [RFC7136]

- o Inactivity Timer. A timer used after receiving a SCHC Fragment to detect when, due to a communication error, there is no possibility to continue an on-going fragmented SCHC Packet transmission.
- o L2: Layer two. The immediate lower layer SCHC interfaces with. It is provided by an underlying LPWAN technology.
- o 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.
- o MIC: Message Integrity Check. A SCHC F/R header field computed over the fragmented SCHC Packet and potential fragment padding, used for error detection after SCHC Packet reassembly.
- o MO: Matching Operator. An operator used to match a value contained in a header field with a value contained in a Rule.
- o 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.
- o Retransmission Timer. A timer used by the SCHC Fragment sender during an on-going fragmented SCHC Packet transmission to detect possible link errors when waiting for a possible incoming SCHC ACK.
- o Rule: A set of header field values.
- o Rule entry: A column in a Rule that describes a parameter of the header field.
- o 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.
- o SCHC ACK: A SCHC acknowledgement for fragmentation. This message is used to report on the success of reception of a set of SCHC Fragments. See Section 8 for more details.
- o 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.

- o SCHC F/R: Static Context Header Compression Fragmentation/Reassembly. A protocol used on both sides, at the Dev and at the network, to achieve Fragmentation/Reassembly of SCHC Packets. SCHC F/R has three reliability modes.
- o SCHC Fragment: A data unit that carries a subset of a SCHC Packet. SCHC F/R is needed when the size of a SCHC packet exceeds the available payload size of the underlying L2 technology data unit. See Section 8.
- o 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.
- o TV: Target value. A value contained in a Rule that will be matched with the value of a header field.
- o Up: Uplink direction for compression/decompression in both sides, from the Dev SCHC C/D to the network SCHC C/D.
- o W: Window bit. A SCHC Fragment header field used in ACK-on-Error or ACK-Always mode Section 8, which carries the same value for all SCHC Fragments of a window.
- o Window: A subset of the SCHC Fragments needed to carry a SCHC Packet (see Section 8).

## 5. SCHC overview

SCHC can be abstracted 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.

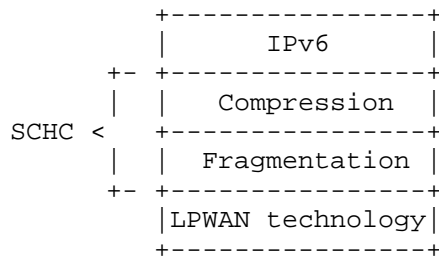
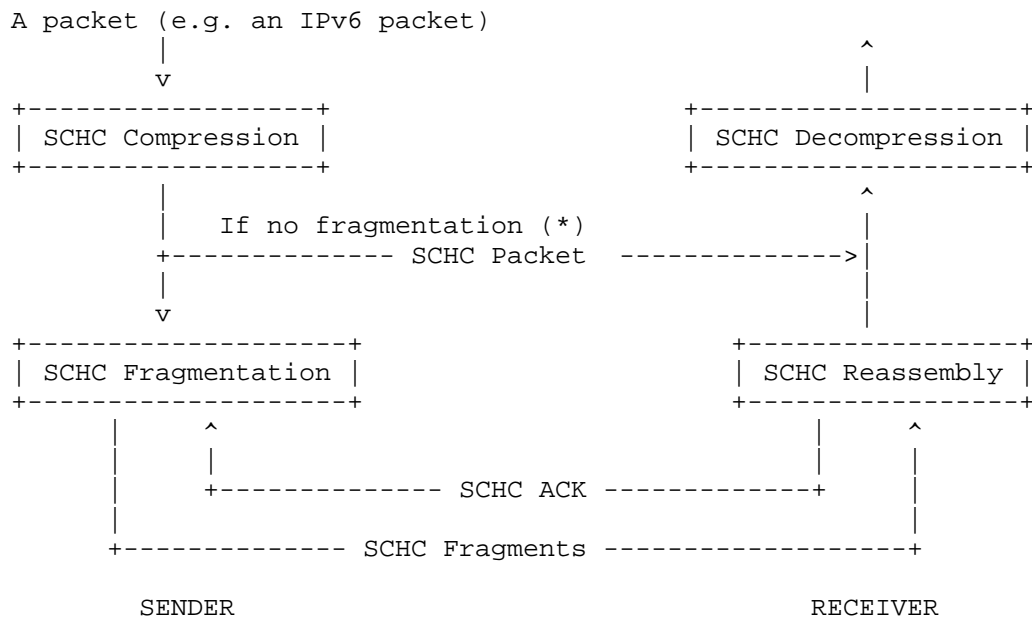


Figure 2: Protocol stack comprising IPv6, SCHC and an LPWAN technology

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 size exceeds the layer 2 (L2) MTU, 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 LPWAN technology over which SCHC is applied. See LPWAN technology-specific documents.

Figure 3: SCHC operations taking place at the sender and the receiver

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 a Rule ID and a Compression Residue. The Compression Residue may be absent, see Section 7. Both the Rule ID and the Compression Residue potentially have a variable size, and generally are not a multiple of bytes in size.

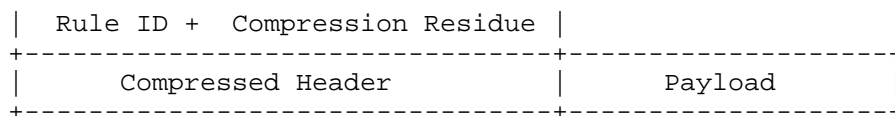


Figure 4: SCHC Packet

The Fragment Header size is variable and depends on the Fragmentation parameters. The Fragment payload contains a part of the SCHC Packet Compressed Header, a part of the SCHC Packet Payload or both. Its

size depends on the L2 data unit, see Section 8. The SCHC Fragment has the following format:

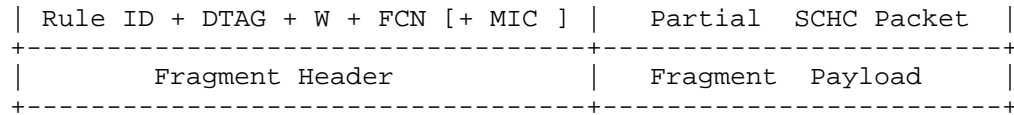


Figure 5: SCHC Fragment

The SCHC ACK is only used for Fragmentation. It has the following format:

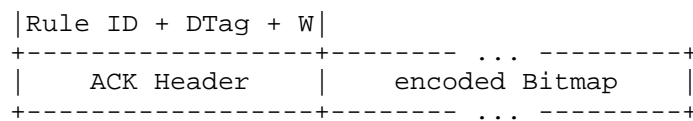


Figure 6: SCHC ACK

The SCHC ACK Header and the encoded Bitmap both have variable size.

Figure 7 below maps the functional elements of Figure 3 onto the LPWAN architecture elements of Figure 1.

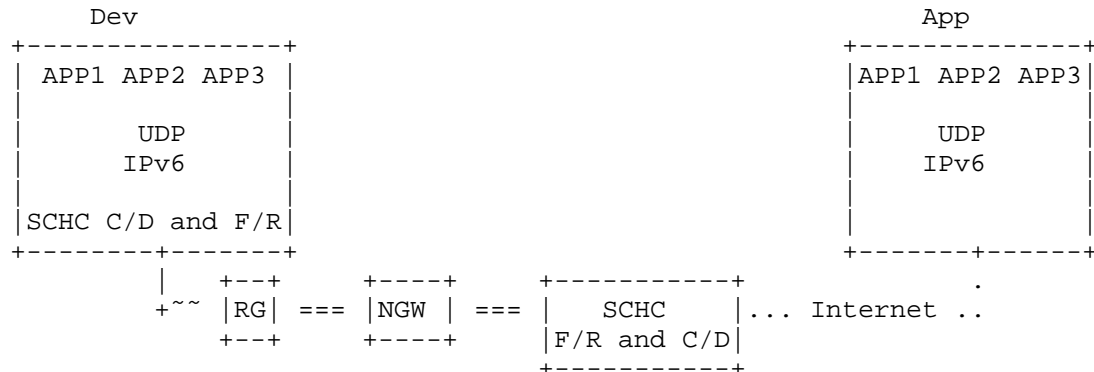


Figure 7: Architecture

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.

Let's describe the operation in the Uplink direction. The Device application packets use IPv6 or IPv6/UDP protocols. Before sending

these packets, the Dev compresses their headers using SCHC C/D and, if the SCHC Packet resulting from the compression exceeds the maximum payload size of the underlying LPWAN technology, SCHC F/R is performed (see Section 8). The resulting SCHC Fragments are sent as one or more L2 frames to an LPWAN Radio Gateway (RG) which forwards them to a Network Gateway (NGW). The NGW sends the data to a SCHC F/R and then to the SCHC C/D for decompression. 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. After decompression, the packet can be sent over the Internet to one or several LPWAN Application Servers (App).

The SCHC C/D and F/R process is symmetrical, therefore the description of the Downlink direction trivially derives from 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.

The Rule IDs are used:

- o 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.
- o At least one Rule ID MAY be allocated to tagging packets for which SCHC compression was not possible (no matching Rule was found).
- o In SCHC F/R, to identify the specific modes and settings of SCHC Fragments being transmitted, and to identify the SCK ACKs, including their modes and settings. Note that in the case of bidirectional communication, at least two Rule ID values are therefore needed for F/R.

## 7. Static Context Header Compression

In order to perform header compression, this document defines a mechanism called Static Context Header Compression (SCHC), which is based on using context, i.e. a set of Rules to compress or decompress

headers. SCHC avoids context synchronization, which is the most bandwidth-consuming operation in other header compression mechanisms such as RoHC [RFC5795]. Since the nature 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 on both ends 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. How Rules are generated is out of the scope of this document. The Rules MAY be changed at run-time but the way to do this will be specified in another document.

The context contains a list of Rules (cf. Figure 8). 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).

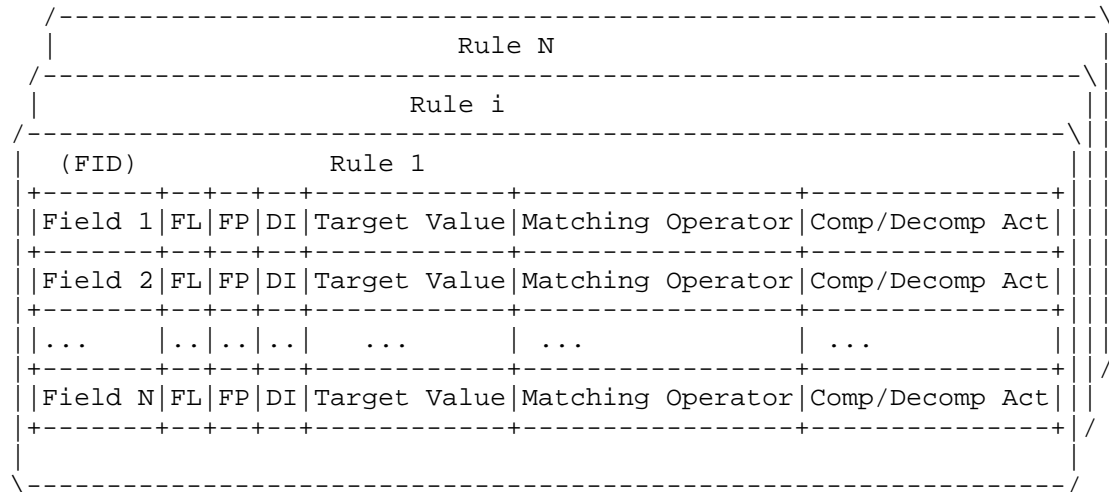


Figure 8: Compression/Decompression Context

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 Compression Residue is sent. 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:

- o Field ID (FID) is a unique value to define the header field.
- o 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.
- o 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 occurrence).
- o 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.
- o Target Value (TV) is the value used to make the match with the packet header field. The Target Value can be of any type (integer, strings, etc.). For instance, it can be a single value or a more complex structure (array, list, etc.), such as a JSON or a CBOR structure.
- o 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.

- o 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 correlate 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:

- o Compression Rule selection: The goal is to identify which Rule(s) will be used to compress the packet's headers. 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 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 explore 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 Rule is found, then the header MUST be sent without compression. This MAY require the use of the SCHC F/R process.
- o Sending: If an eligible Rule is found, the Rule ID is sent to the other end followed by the Compression Residue (which could be empty) 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 specific underlying LPWAN technology. For example, it can be either included in an L2 header or sent in the first byte of the L2 payload. (Cf. Figure 9). This process will be specified in the LPWAN technology-specific document 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.
- o 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 (e.g. MAC address, if exists) and selects the appropriate Rule from the Rule ID. If a source identifier is present in the L2 technology, it is used to select the Rule ID. This Rule describes the compressed header format and associates the values to the header fields. The receiver applies the CDA action to reconstruct the original header fields. The CDA application order can be different from the order given by the Rule. For instance, Compute-\* SHOULD be applied at the end, after all the other CDAs.

```

+--- ... ---+----- ... -----+-----+-----+
| Rule ID |Compression Residue| packet payload |
+--- ... ---+----- ... -----+-----+-----+

|----- compressed header -----|

```

Figure 9: SCHC C/D Packet Format

#### 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 indifferently 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:

- o equal: The match result is True if a field value in a packet and the value in the TV are equal.
- o ignore: No check is done between a field value in a packet and a TV in the Rule. The result of the matching is always true.
- o 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.
- o 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 10: Compression and Decompression Actions

Figure 10 summarizes the basic functions that can be used to compress and decompress a field. The first column lists the actions name. The second and third columns outline the reciprocal compression/decompression behavior for each action.

Compression is done in order that Fields Descriptions appear in a Rule. The result of each Compression/Decompression Action is appended to the working 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.

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

- o If the size is between 0 and 14, it is sent as a 4-bits integer.
- o For values between 15 and 254, the first 4 bits sent are set to 1 and the size is sent using 8 bits integer.
- o For higher values of size, the first 12 bits are set to 1 and the next two bytes contain the size value as a 16 bits 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.

#### 7.5.1. not-sent CDA

The not-sent function is generally used when the field value is specified in a Rule and therefore known by both the Compressor and the Decompressor. This action is generally 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.

#### 7.5.2. 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. This function is generally used with the "ignore" MO.

#### 7.5.3. mapping-sent CDA

The mapping-sent is used to send a smaller index (the index into the Target Value list of values) instead of the original value. This function 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.

#### 7.5.4. 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.

#### 7.5.5. DevIID, AppIID CDA

These functions 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 current LPWAN technologies frames contain a single 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 LPWAN technology and MAY depend on the Device ID size.

In the downlink direction (Dw), at the compressor, this DevIID CDA may be used to generate the L2 addresses on the LPWAN, based on the packet destination address.

#### 7.5.6. Compute-\*

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

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

#### 8.1. Overview

In LPWAN technologies, the L2 data unit size typically varies from tens to hundreds of bytes. The SCHC F/R (Fragmentation /Reassembly) MAY be used either because after applying SCHC C/D or when SCHC C/D is not possible the entire SCHC Packet still exceeds the L2 data unit.

The SCHC F/R functionality defined in this document has been designed under the assumption that data unit out-of-sequence delivery will not

happen between the entity performing fragmentation and the entity performing reassembly. This assumption allows reducing the complexity and overhead of the SCHC F/R mechanism.

This document also assumes that the L2 data unit size does not vary while a fragmented SCHC Packet is being transmitted.

To adapt the SCHC F/R to the capabilities of LPWAN technologies, it is required to enable optional SCHC Fragment retransmission and to allow for a range of reliability options for sending the SCHC Fragments. This document does not make any decision with regard to which SCHC Fragment delivery reliability mode will be used over a specific LPWAN technology. These details will be defined in other technology-specific documents.

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 generates SCHC Fragments and SCHC ACKs that are, for most of them, multiples of L2 Words. The padding overhead is kept to the absolute minimum. See Section 9.

## 8.2. Fragmentation Tools

This subsection describes the different tools that are used to enable the SCHC F/R functionality defined in this document, such as fields in the SCHC F/R header frames (see the related formats in Section 8.4), windows and timers.

- o Rule ID. The Rule ID is present in the SCHC Fragment header and in the SCHC ACK header formats. The Rule ID in a SCHC Fragment header is used to identify that a SCHC Fragment is being carried, which SCHC F/R reliability mode is used and which window size is used. The Rule ID in the SCHC Fragment header also allows interleaving non-fragmented SCHC Packets and SCHC Fragments that carry other SCHC Packets. The Rule ID in a SCHC ACK identifies the message as a SCHC ACK.
- o Fragment Compressed Number (FCN). The FCN is included in all SCHC Fragments. This field can be understood as a truncated, efficient representation of a larger-sized fragment number, and does not carry an absolute SCHC Fragment number. There are two FCN reserved values that are used for controlling the SCHC F/R process, as described next:
  - \* The FCN value with all the bits equal to 1 (All-1) denotes the last SCHC Fragment of a packet. The last window of a packet is called an All-1 window.

- \* The FCN value with all the bits equal to 0 (All-0) denotes the last SCHC Fragment of a window that is not the last one of the packet. Such a window is called an All-0 window.

The rest of the FCN values are assigned in a sequentially decreasing order, which has the purpose to avoid possible ambiguity for the receiver that might arise under certain conditions. In the SCHC Fragments, this field is an unsigned integer, with a size of N bits. In the No-ACK mode, the size is set to 1 bit (N=1), All-0 is used in all SCHC Fragments and All-1 for the last one. For the other reliability modes, it is recommended to use a number of bits (N) equal to or greater than 3. Nevertheless, the appropriate value of N MUST be defined in the corresponding technology-specific profile documents. For windows that are not the last one of a fragmented SCHC Packet, the FCN for the last SCHC Fragment in such windows is an All-0. This indicates that the window is finished and communication proceeds according to the reliability mode in use. The FCN for the last SCHC Fragment in the last window is an All-1, indicating the last SCHC Fragment of the SCHC Packet. It is also important to note that, in the No-ACK mode or when N=1, the last SCHC Fragment of the packet will carry a FCN equal to 1, while all previous SCHC Fragments will carry a FCN to 0. For further details see Section 8.5. The highest FCN in the window, denoted MAX\_WIND\_FCN, MUST be a value equal to or smaller than  $2^N - 2$ . (Example for N=5, MAX\_WIND\_FCN MAY be set to 23, then subsequent FCNs are set sequentially and in decreasing order, and the FCN will wrap from 0 back to 23).

- o Datagram Tag (DTag). The DTag field, if present, is set to the same value for all SCHC Fragments carrying the same SCHC packet, and to different values for different SCHC Packets. Using this field, the sender can interleave fragments from different SCHC Packets, while the receiver can still tell them apart. In the SCHC Fragment formats, the size of the DTag field is T bits, which MAY be set to a value greater than or equal to 0 bits. For each new SCHC Packet processed by the sender, DTag MUST be sequentially increased, from 0 to  $2^T - 1$  wrapping back from  $2^T - 1$  to 0. In the SCHC ACK format, DTag carries the same value as the DTag field in the SCHC Fragments for which this SCHC ACK is intended. When there is no Dtag, there can be only one SCHC Packet in transit. Only after all its fragments have been transmitted can another SCHC Packet be sent. The length of DTag, denoted T, is not specified in this document because it is technology dependant. It will be defined in the corresponding technology-specific documents, based on the number of simultaneous packets that are to be supported.

- o W (window): W is a 1-bit field. This field carries the same value for all SCHC Fragments of a window, and it is complemented for the next window. The initial value for this field is 0. In the SCHC ACK format, this field also has a size of 1 bit. In all SCHC ACKs, the W bit carries the same value as the W bit carried by the SCHC Fragments whose reception is being positively or negatively acknowledged by the SCHC ACK.
- o Message Integrity Check (MIC). This field is computed by the sender over the complete SCHC Packet and before SCHC fragmentation. The MIC allows the receiver to check errors in the reassembled packet, while it also enables compressing the UDP checksum by use of SCHC compression. The CRC32 as 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 algorithms MAY be required and are defined in the technology-specific documents as well as the length in bits of the MIC used.
- o C (MIC checked): C is a 1-bit field. This field is used in the SCHC ACK packets to report the outcome of the MIC check, i.e. whether the reassembled packet was correctly received or not. A value of 1 represents a positive MIC check at the receiver side (i.e. the MIC computed by the receiver matches the received MIC).
- o Retransmission Timer. A SCHC Fragment sender uses it after the transmission of a window to detect a transmission error of the SCHC ACK corresponding to this window. Depending on the reliability mode, it will lead to a request a SCHC ACK retransmission (in ACK-Always mode) or it will trigger the transmission of the next window (in ACK-on-Error mode). The duration of this timer is not defined in this document and MUST be defined in the corresponding technology-specific documents.
- o Inactivity Timer. A SCHC Fragment receiver uses it to take action when there is a problem in the transmission of SCHC fragments. Such a problem could be detected by the receiver not getting a single SCHC Fragment during a given period of time. When this happens, an Abort message will be sent (see related text later in this section). Initially, and each time a SCHC Fragment is received, the timer is reinitialized. The duration of this timer is not defined in this document and MUST be defined in the corresponding technology-specific document.
- o Attempts. This counter counts the requests for a missing SCHC ACK. When it reaches the value MAX\_ACK\_REQUESTS, the sender assumes there are recurrent SCHC Fragment transmission errors and determines that an Abort is needed. The default value

MAX\_ACK\_REQUESTS is not stated in this document, and it is expected to be defined in the corresponding technology-specific document. The Attempts counter is defined per window. It is initialized each time a new window is used.

- o Bitmap. The Bitmap is a sequence of bits carried in a SCHC ACK. Each bit in the Bitmap corresponds to a SCHC fragment of the current window, and provides feedback on whether the SCHC Fragment has been received or not. The right-most position on the Bitmap reports if the All-0 or All-1 fragment has been received or not. Feedback on the SCHC fragment with the highest FCN value is provided by the bit in the left-most position of the Bitmap. In the Bitmap, a bit set to 1 indicates that the SCHC Fragment of FCN corresponding to that bit position has been correctly sent and received. The text above describes the internal representation of the Bitmap. When inserted in the SCHC ACK for transmission from the receiver to the sender, the Bitmap is shortened for energy/bandwidth optimisation, see more details in Section 8.4.3.1.
- o Abort. On expiration of the Inactivity timer, or when Attempts reaches MAX\_ACK\_REQUESTS or upon occurrence of some other error, the sender or the receiver may use the Abort. When the receiver needs to abort the on-going fragmented SCHC Packet transmission, it sends the Receiver-Abort format. When the sender needs to abort the transmission, it sends the Sender-Abort format. None of the Aborts are acknowledged.

### 8.3. Reliability modes

This specification defines three reliability modes: No-ACK, ACK-Always, and ACK-on-Error. ACK-Always and ACK-on-Error operate on windows of SCHC Fragments. A window of SCHC Fragments is a subset of the full set of SCHC Fragments needed to carry a SCHC Packet.

- o No-ACK. No-ACK is the simplest SCHC Fragment reliability mode. The receiver does not generate overhead in the form of acknowledgements (ACKs). However, this mode does not enhance reliability beyond that offered by the underlying LPWAN technology. In the No-ACK mode, the receiver MUST NOT issue SCHC ACKs. See further details in Section 8.5.1.
- o ACK-Always. The ACK-Always mode provides flow control using a windowing scheme. This mode is also able to handle long bursts of lost SCHC Fragments since detection of such events can be done before the end of the SCHC Packet transmission as long as the window size is short enough. However, such benefit comes at the expense of SCHC ACK use. In ACK-Always, the receiver sends a SCHC ACK after a window of SCHC Fragments has been received. The SCHC

ACK is used to inform the sender which SCHC Fragments in the current window have been well received. Upon a SCHC ACK reception, the sender retransmits the lost SCHC Fragments. When a SCHC ACK is lost and the sender has not received it by the expiration of the Retransmission Timer, the sender uses a SCHC ACK request by sending the All-0 empty SCHC Fragment when it is not the last window and the All-1 empty Fragment when it is the last window. The maximum number of SCHC ACK requests is `MAX_ACK_REQUESTS`. If `MAX_ACK_REQUESTS` is reached, the transmission needs to be aborted. See further details in Section 8.5.2.

- o ACK-on-Error. The ACK-on-Error mode is suitable for links offering relatively low L2 data unit loss probability. In this mode, the SCHC Fragment receiver reduces the number of SCHC ACKs transmitted, which MAY be especially beneficial in asymmetric scenarios. The receiver transmits a SCHC ACK only after the complete window transmission and if at least one SCHC Fragment of this window has been lost. An exception to this behavior is in the last window, where the receiver MUST transmit a SCHC ACK, including the C bit set based on the MIC checked result, even if all the SCHC Fragments of the last window have been correctly received. The SCHC ACK gives the state of all the SCHC Fragments of the current window (received or lost). Upon a SCHC ACK reception, the sender retransmits any lost SCHC Fragments based on the SCHC ACK. If a SCHC ACK is not transmitted back by the receiver at the end of a window, the sender assumes that all SCHC Fragments have been correctly received. When a SCHC ACK is lost, the sender assumes that all SCHC Fragments covered by the lost SCHC ACK have been successfully delivered, so the sender continues transmitting the next window of SCHC Fragments. If the next SCHC Fragments received belong to the next window and it is still expecting fragments from the previous window, the receiver will abort the on-going fragmented packet transmission. See further details in Section 8.5.3.

The same reliability mode MUST be used for all SCHC Fragments of a SCHC Packet. The decision on which reliability mode will be used and whether the same reliability mode applies to all SCHC Packets is an implementation problem and is out of the scope of this document.

Note that the reliability mode choice is not necessarily tied to a particular characteristic of the underlying L2 LPWAN technology, e.g. the No-ACK mode MAY be used on top of an L2 LPWAN technology with symmetric characteristics for uplink and downlink. This document does not make any decision as to which SCHC Fragment reliability modes are relevant for a specific LPWAN technology.



Examples of the different reliability modes described are provided in Appendix B.

#### 8.4. Fragmentation Formats

This section defines the SCHC Fragment format, including the All-0 and All-1 formats and their "empty" variations, the SCHC ACK format and the Abort formats.

A SCHC Fragment conforms to the general format shown in Figure 11. It comprises a SCHC Fragment Header and a SCHC Fragment Payload. In addition, the last SCHC Fragment carries as many padding bits as needed to fill up an L2 Word. The SCHC Fragment Payload carries a subset of the SCHC Packet. The SCHC Fragment is the data unit passed on to the L2 for transmission.

```
+-----+-----+~~~~~
| Fragment Header |   Fragment payload   | padding (as needed)
+-----+-----+~~~~~
```

Figure 11: SCHC Fragment general format. Presence of a padding field is optional

##### 8.4.1. Fragments that are not the last one

In ACK-Always or ACK-on-Error, SCHC Fragments except the last one SHALL conform to the detailed format defined in Figure 12.

```
|----- Fragment Header -----|
|  -- T -- | 1 |  -- N -- |
+-+ ... +-+ ... +-+ ... +-----+
| Rule ID | DTag | W | FCN | Fragment payload |
+-+ ... +-+ ... +-+ ... +-----+
```

Figure 12: Fragment Detailed Format for Fragments except the Last One, ACK-Always and ACK-on-Error

In the No-ACK mode, SCHC Fragments except the last one SHALL conform to the detailed format defined in Figure 13.

```

|---- Fragment Header ----|
|  -- T --  |  -- N --  |
+-- ... --+  ... --+  ... --+-----+
| Rule ID | DTag | FCN | Fragment payload |
+-- ... --+  ... --+  ... --+-----+

```

Figure 13: Fragment Detailed Format for Fragments except the Last One, No-ACK mode

The total size of the fragment header is not necessarily a multiple of the L2 Word size. To build the fragment payload, SCHC F/R MUST take from the SCHC Packet a number of bits that makes the SCHC Fragment an exact multiple of L2 Words. As a consequence, no padding bit is used for these fragments.

#### 8.4.1.1. All-0 fragment

The All-0 format is used for sending the last SCHC Fragment of a window that is not the last window of the SCHC Packet.

```

|----- Fragment Header -----|
|  -- T --  | 1 |  -- N --  |
+-- ... --+  ... --+  ... --+-----+
| Rule ID | DTag | W | 0..0 | Fragment payload |
+-- ... --+  ... --+  ... --+-----+

```

Figure 14: All-0 fragment detailed format

This is simply an instance of the format described in Figure 12. An All-0 fragment payload MUST be at least the size of an L2 Word. The rationale is that the All-0 empty fragment (see Section 8.4.1.2) needs to be distinguishable from the All-0 regular fragment, even in the presence of padding.

#### 8.4.1.2. All-0 empty fragment

The All-0 empty fragment is an exception to the All-0 fragment described above. It is used by a sender to request the retransmission of a SCHC ACK by the receiver. It is only used in ACK-Always mode.

```

|----- Fragment Header -----|
|  -- T -- | 1 |  -- N -- |
+-- ... ---+ ... ---+ ... ---+ ~~~~~
| Rule ID | DTag | W | 0..0 | padding (as needed)      (no payload)
+-- ... ---+ ... ---+ ... ---+ ~~~~~

```

Figure 15: All-0 empty fragment detailed format

The size of the All-0 fragment header is generally not a multiple of the L2 Word size. Therefore, an All-0 empty fragment generally needs padding bits. The padding bits are always less than an L2 Word.

Since an All-0 payload MUST be at least the size of an L2 Word, a receiver can distinguish an All-0 empty fragment from a regular All-0 fragment, even in the presence of padding.

#### 8.4.2. All-1 fragment

In the No-ACK mode, the last SCHC Fragment of a SCHC Packet SHALL contain a SCHC Fragment header that conforms to the detailed format shown in Figure 16.

```

|----- Fragment Header -----|
|  -- T -- | -N=1- |
+----- ... ---+ ... ---+-----+ ~~~~~
| Rule ID | DTag | 1 | MIC | payload | padding (as needed)
+----- ... ---+ ... ---+-----+ ~~~~~

```

Figure 16: All-1 Fragment Detailed Format for the Last Fragment, No-ACK mode

In ACK-Always or ACK-on-Error mode, the last fragment of a SCHC Packet SHALL contain a SCHC Fragment header that conforms to the detailed format shown in Figure 17.

```

|----- Fragment Header -----|
|  -- T -- | 1 |  -- N -- |
+-- ... ---+ ... ---+ ... ---+ ~~~~~
| Rule ID | DTag | W | 11..1 | MIC | payload | padding (as needed)
+-- ... ---+ ... ---+ ... ---+ ~~~~~
                                (FCN)

```

Figure 17: All-1 Fragment Detailed Format for the Last Fragment, ACK-Always or ACK-on-Error

The total size of the All-1 SCHC Fragment header is generally not a multiple of the L2 Word size. The All-1 fragment being the last one of the SCHC Packet, SCHC F/R cannot freely choose the payload size to align the fragment to an L2 Word. Therefore, padding bits are generally appended to the All-1 fragment to make it a multiple of L2 Words in size.

The MIC MUST be computed on the payload and the padding bits. The rationale is that the SCHC Reassembler needs to check the correctness of the reassembled SCHC packet but has no way of knowing where the payload ends. Indeed, the latter requires decompressing the SCHC Packet.

An All-1 fragment payload MUST be at least the size of an L2 Word. The rationale is that the All-1 empty fragment (see Section 8.4.2.1) needs to be distinguishable from the All-1 fragment, even in the presence of padding. This may entail saving an L2 Word from the previous fragment payload to make the payload of this All-1 fragment big enough.

The values for N, T and the length of MIC are not specified in this document, and SHOULD be determined in other documents (e.g. technology-specific profile documents).

The length of the MIC MUST be at least an L2 Word size. The rationale is to be able to distinguish a Sender-Abort (see Section 8.4.4) from an All-1 Fragment, even in the presence of padding.

#### 8.4.2.1. All-1 empty fragment

The All-1 empty fragment format is an All-1 fragment format without a payload (see Figure 18). It is used by a fragment sender, in either ACK-Always or ACK-on-Error, to request a retransmission of the SCHC ACK for the All-1 window.

The size of the All-1 empty fragment header is generally not a multiple of the L2 Word size. Therefore, an All-1 empty fragment generally needs padding bits. The padding bits are always less than an L2 Word.

Since an All-1 payload MUST be at least the size of an L2 Word, a receiver can distinguish an All-1 empty fragment from a regular All-1 fragment, even in the presence of padding.

```

|----- Fragment Header -----|
|-- T --|1|-- N --|
+--- ... ---+ ... ---+ ... ---+ ... ---+ ~~~~~
| Rule ID | DTag | W | 1..1 | MIC | padding as needed (no payload)
+--- ... ---+ ... ---+ ... ---+ ... ---+ ~~~~~

```

Figure 18: All-1 for Retries format, also called All-1 empty

#### 8.4.3. SCHC ACK format

The format of a SCHC ACK that acknowledges a window that is not the last one (denoted as All-0 window) is shown in Figure 19.

```

|-- T --|1|
+----- ... ---+ ... ---+----- ... -----+
| Rule ID | DTag | W | encoded Bitmap | (no payload)
+----- ... ---+ ... ---+----- ... -----+

```

Figure 19: ACK format for All-0 windows

To acknowledge the last window of a packet (denoted as All-1 window), a C bit (i.e. MIC checked) following the W bit is set to 1 to indicate that the MIC check computed by the receiver matches the MIC present in the All-1 fragment. If the MIC check fails, the C bit is set to 0 and the Bitmap for the All-1 window follows.

```

|-- T --|1|1|
+----- ... ---+ ... ---+----- ... -----+
| Rule ID | DTag | W | 1 | (MIC correct)
+----- ... ---+ ... ---+----- ... -----+

+----- ... ---+ ... ---+----- ... -----+
| Rule ID | DTag | W | 0 | encoded Bitmap | (MIC Incorrect)
+----- ... ---+ ... ---+----- ... -----+
                        C

```

Figure 20: Format of a SCHC ACK for All-1 windows

The Rule ID and Dtag values in the SCHC ACK messages MUST be identical to the ones used in the SCHC Fragments that are being acknowledged. This allows matching the SCHC ACK and the corresponding SCHC Fragments.

The Bitmap carries information on the reception of each fragment of the window as described in Section 8.2.

See Appendix D for a discussion on the size of the Bitmaps.

In order to reduce the SCK ACK size, the Bitmap that is actually transmitted is shortened ("encoded") as explained in Section 8.4.3.1.

#### 8.4.3.1. Bitmap Encoding

The SCHC ACK that is transmitted is truncated by applying the following algorithm: the longest contiguous sequence of bits that starts at an L2 Word boundary of the SCHC ACK, where the bits of that sequence are all set to 1, are all part of the Bitmap and finish exactly at the end of the Bitmap, if one such sequence exists, MUST NOT be transmitted. Because the SCHC Fragment sender knows the actual Bitmap size, it can reconstruct the original Bitmap from the shortened bitmap.

When shortening effectively takes place, the SCHC ACK is a multiple of L2 Words, and padding MUST NOT be appended. When shortening does not happen, padding bits MUST be appended as needed to fill up the last L2 Word.

Figure 21 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.

```

|-- SCHC ACK Header --|-----|          Bitmap          |-----| | | | | | | | | | | | | | | | | | | | |
| Rule ID  | DTag  |W|1|0|1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|1|
| next L2 Word boundary ->| next L2 Word | next L2 Word |

```

Figure 21: A non-encoded Bitmap

Figure 22 shows that the last 14 bits are not sent.

```

          |-- T --|1|
+----- ... --+- ... +--+--+--+
| Rule ID | DTag |W|1|0|1|
+----- ... --+- ... +--+--+--+
      next L2 Word boundary ->|

```

Figure 22: Optimized Bitmap format

Figure 23 shows an example of a SCHC ACK with FCN ranging from 6 down to 0, where the Bitmap indicates that the second and the fifth SCHC Fragments have not been correctly received.

```

                                6 5 4 3 2 1 0 (*)
                                |-- T --|1|
+-----+-----+-----+-----+-----+
| Rule ID | DTag |W|1|0|1|1|0|1|1|          Bitmap before tx
+-----+-----+-----+-----+-----+
next L2 Word boundary ->|<-- L2 Word -->|
      (*)=(FCN values)

+-----+-----+-----+-----+-----+~~~+
| Rule ID | DTag |W|1|0|1|1|0|1|1|Pad|      Encoded Bitmap
+-----+-----+-----+-----+-----+~~~+
next L2 Word boundary ->|<-- L2 Word -->|

```

Figure 23: Example of a Bitmap before transmission, and the transmitted one, for a window that is not the last one

Figure 24 shows an example of a SCHC ACK with FCN ranging from 6 down to 0, where MIC check has failed but the Bitmap indicates that there is no missing SCHC Fragment.

```

|- Fragmentation Header-|6 5 4 3 2 1 7 (*)
                        |-- T --|1|
| Rule ID | DTag |W|0|1|1|1|1|1|1|          Bitmap before tx
next L2 Word boundary ->|<-- L2 Word -->|
                        C
+----- ... ---+ ... -+---+---+
| Rule ID | DTag |W|0|1|          Encoded Bitmap
+----- ... ---+ ... -+---+---+
next L2 Word boundary ->|
      (*) = (FCN values indicating the order)

```

Figure 24: Example of the Bitmap in ACK-Always or ACK-on-Error for the last window

#### 8.4.4. Abort formats

When a SCHC Fragment sender needs to abort the on-going fragmented SCHC Packet transmission, it sends a Sender-Abort. The Sender-Abort format (see Figure 25) is a variation of the All-1 fragment, with neither a MIC nor a payload. All-1 fragments contain at least a MIC. The absence of the MIC indicates a Sender-Abort.

```
|--- Sender-Abort Header ---|
+--- ... ---+ ... -+-+-...-+ ~~~~~
| Rule ID | DTag | W | FCN | padding (as needed)
+--- ... ---+ ... -+-+-...-+ ~~~~~
```

Figure 25: Sender-Abort format. All FCN field bits in this format are set to 1

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

Since an All-1 fragment MIC MUST be at least the size of an L2 Word, a receiver can distinguish a Sender-Abort from an All-1 fragment, even in the presence of padding.

When a SCHC Fragment receiver needs to abort the on-going fragmented SCHC Packet transmission, it transmits a Receiver-Abort. The Receiver-Abort format is a variation on the SCHC ACK format, creating an exception in the encoded Bitmap algorithm. As shown in Figure 26, a Receiver-Abort is coded as a SCHC ACK message with a shortened Bitmap set to 1 up to the first L2 Word boundary, followed by an extra L2 Word full of 1's. Such a message never occurs in a regular acknowledgement and is detected as a Receiver-Abort.

The Rule ID and Dtag values in the Receive-Abort message MUST be identical to the ones used in the fragments of the SCHC Packet the transmission of which is being aborted.

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

```

|- Receiver-Abort Header -|
+---+ ... +---+ ... +---+ +---+ +---+ +---+ +---+ +---+ +---+
| Rule ID | DTag |W| 1..1| 1..1 |
+---+ ... +---+ ... +---+ +---+ +---+ +---+ +---+ +---+ +---+
      next L2 Word boundary -> |<-- L2 Word -->|

```

Figure 26: Receiver-Abort format

Neither the Sender-Abort nor the Receiver-Abort messages are ever acknowledged or retransmitted.

Use cases for the Sender-Abort and Receiver-Abort messages are explained in Section 8.5 or Appendix C.



### 8.5. Baseline mechanism

If after applying SCHC header compression (or when SCHC header compression is not possible) the SCHC Packet does not fit within the payload of a single L2 data unit, the SCHC Packet SHALL be broken into SCHC Fragments and the fragments SHALL be sent to the fragment receiver. The fragment receiver needs to identify all the SCHC Fragments that belong to a given SCHC Packet. To this end, the receiver SHALL use:

- o The sender's L2 source address (if present),
- o The destination's L2 address (if present),
- o Rule ID,
- o DTag (if present).

Then, the fragment receiver MAY determine the SCHC Fragment reliability mode that is used for this SCHC Fragment based on the Rule ID in that fragment.

After a SCHC Fragment reception, the receiver starts constructing the SCHC Packet. It uses the FCN and the arrival order of each SCHC Fragment to determine the location of the individual fragments within the SCHC Packet. For example, the receiver MAY place the fragment payload within a payload reassembly buffer at the location determined from the FCN, the arrival order of the SCHC Fragments, and the fragment payload sizes. In ACK-on-Error or ACK-Always, the fragment receiver also uses the W bit in the received SCHC Fragments. Note that the size of the original, unfragmented packet cannot be determined from fragmentation headers.

Fragmentation functionality uses the FCN value to transmit the SCHC Fragments. It has a length of N bits where the All-1 and All-0 FCN values are used to control the fragmentation transmission. The rest of the FCN numbers MUST be assigned sequentially in a decreasing order, the first FCN of a window is RECOMMENDED to be MAX\_WIND\_FCN, i.e. the highest possible FCN value depending on the FCN number of bits.

In all modes, the last SCHC Fragment of a packet MUST contain a MIC which is used to check if there are errors or missing SCHC Fragments and MUST use the corresponding All-1 fragment format. Note that a SCHC Fragment with an All-0 format is considered the last SCHC Fragment of the current window.

If the receiver receives the last fragment of a SCHC Packet (All-1), it checks for the integrity of the reassembled SCHC Packet, based on the MIC received. In No-ACK, if the integrity check indicates that the reassembled SCHC Packet does not match the original SCHC Packet (prior to fragmentation), the reassembled SCHC Packet MUST be discarded. In ACK-on-Error or ACK-Always, a MIC check is also performed by the fragment receiver after reception of each subsequent SCHC Fragment retransmitted after the first MIC check.

Notice that the SCHC ACK for the All-1 window carries one more bit (the C bit) compared to the SCHC ACKs for the previous windows. See Appendix D for a discussion on various options to deal with this "bump" in the SCHC ACK.

There are three reliability modes: No-ACK, ACK-Always and ACK-on-Error. In ACK-Always and ACK-on-Error, a jumping window protocol uses two windows alternatively, identified as 0 and 1. A SCHC Fragment with all FCN bits set to 0 (i.e. an All-0 fragment) indicates that the window is over (i.e. the SCHC Fragment is the last one of the window) and allows to switch from one window to the next one. The All-1 FCN in a SCHC Fragment indicates that it is the last fragment of the packet being transmitted and therefore there will not be another window for this packet.

#### 8.5.1. No-ACK

In the No-ACK mode, there is no feedback communication from the fragment receiver. The sender will send all the SCHC fragments of a packet without any possibility of knowing if errors or losses have occurred. As, in this mode, there is no need to identify specific SCHC Fragments, a one-bit FCN MAY be used. Consequently, the FCN All-0 value is used in all SCHC fragments except the last one, which carries an All-1 FCN and the MIC. The receiver will wait for SCHC Fragments and will set the Inactivity timer. The receiver will use the MIC contained in the last SCHC Fragment to check for errors. When the Inactivity Timer expires or if the MIC check indicates that the reassembled packet does not match the original one, the receiver will release all resources allocated to reassembling this packet. The initial value of the Inactivity Timer will be determined based on the characteristics of the underlying LPWAN technology and will be defined in other documents (e.g. technology-specific profile documents).

#### 8.5.2. ACK-Always

In ACK-Always, the sender transmits SCHC Fragments by using the two-jumping-windows procedure. A delay between each SCHC fragment can be added to respect local regulations or other constraints imposed by

the applications. Each time a SCHC fragment is sent, the FCN is decreased by one. When the FCN reaches value 0, if there are more SCHC Fragments remaining to be sent, the sender transmits the last SCHC Fragment of this window using the All-0 fragment format. It then starts the Retransmission Timer and waits for a SCHC ACK. Otherwise, if FCN reaches 0 and the sender transmits the last SCHC Fragment of the SCHC Packet, the sender uses the All-1 fragment format, which includes a MIC. The sender sets the Retransmission Timer and waits for the SCHC ACK to know if transmission errors have occurred.

The Retransmission Timer is dimensioned based on the LPWAN technology in use. When the Retransmission Timer expires, the sender sends an All-0 empty (resp. All-1 empty) fragment to request again the SCHC ACK for the window that ended with the All-0 (resp. All-1) fragment just sent. The window number is not changed.

After receiving an All-0 or All-1 fragment, the receiver sends a SCHC ACK with an encoded Bitmap reporting whether any SCHC fragments have been lost or not. When the sender receives a SCHC ACK, it checks the W bit carried by the SCHC ACK. Any SCHC ACK carrying an unexpected W bit value is discarded. If the W bit value of the received SCHC ACK is correct, the sender analyzes the rest of the SCHC ACK message, such as the encoded Bitmap and the MIC. If all the SCHC Fragments sent for this window have been well received, and if at least one more SCHC Fragment needs to be sent, the sender advances its sending window to the next window value and sends the next SCHC Fragments. If no more SCHC Fragments have to be sent, then the fragmented SCHC Packet transmission is finished.

However, if one or more SCHC Fragments have not been received as per the SCHC ACK (i.e. the corresponding bits are not set in the encoded Bitmap) then the sender resends the missing SCHC Fragments. When all missing SCHC Fragments have been retransmitted, the sender starts the Retransmission Timer, even if an All-0 or an All-1 has not been sent as part of this retransmission and waits for a SCHC ACK. Upon receipt of the SCHC ACK, if one or more SCHC Fragments have not yet been received, the counter Attempts is increased and the sender resends the missing SCHC Fragments again. When Attempts reaches MAX\_ACK\_REQUESTS, the sender aborts the on-going fragmented SCHC Packet transmission by sending a Sender-Abort message and releases any resources for transmission of the packet. The sender also aborts an on-going fragmented SCHC Packet transmission when a failed MIC check is reported by the receiver or when a SCHC Fragment that has not been sent is reported in the encoded Bitmap.

On the other hand, at the beginning, the receiver side expects to receive window 0. Any SCHC Fragment received but not belonging to

the current window is discarded. All SCHC Fragments belonging to the correct window are accepted, and the actual SCHC Fragment number managed by the receiver is computed based on the FCN value. The receiver prepares the encoded Bitmap to report the correctly received and the missing SCHC Fragments for the current window. After each SCHC Fragment is received, the receiver initializes the Inactivity Timer. When the Inactivity Timer expires, the transmission is aborted by the receiver sending a Receiver-Abort message.

When an All-0 fragment is received, it indicates that all the SCHC Fragments have been sent in the current window. Since the sender is not obliged to always send a full window, some SCHC Fragment number not set in the receiver memory may not correspond to losses. The receiver sends the corresponding SCHC ACK, the Inactivity Timer is set and the transmission of the next window by the sender can start.

If an All-0 fragment has been received and all SCHC Fragments of the current window have also been received, the receiver then expects a new Window and waits for the next SCHC Fragment. Upon receipt of a SCHC Fragment, if the window value has not changed, the received SCHC Fragments are part of a retransmission. A receiver that has already received a SCHC Fragment SHOULD discard it, otherwise, it updates the Bitmap. If all the bits of the Bitmap are set to one, the receiver MUST send a SCHC ACK without waiting for an All-0 fragment and the Inactivity Timer is initialized.

On the other hand, if the window value of the next received SCHC Fragment is set to the next expected window value, this means that the sender has received a correct encoded Bitmap reporting that all SCHC Fragments have been received. The receiver then updates the value of the next expected window.

When an All-1 fragment is received, it indicates that the last SCHC Fragment of the packet has been sent. Since the last window is not always full, the MIC will be used by the receiver to detect if all SCHC Fragments of the packet have been received. A correct MIC indicates the end of the transmission but the receiver MUST stay alive for an Inactivity Timer period to answer to any empty All-1 fragments the sender MAY send if SCHC ACKs sent by the receiver are lost. If the MIC is incorrect, some SCHC Fragments have been lost. The receiver sends the SCHC ACK regardless of successful fragmented SCHC Packet reception or not, the Inactivity Timer is set. In case of an incorrect MIC, the receiver waits for SCHC Fragments belonging to the same window. After MAX\_ACK\_REQUESTS, the receiver will abort the on-going fragmented SCHC Packet transmission by transmitting a the Receiver-Abort format. The receiver also aborts upon Inactivity Timer expiration by sending a Receiver-Abort message.

If the sender receives a SCK ACK with a Bitmap containing a bit set for a SCHC Fragment that it has not sent during the transmission phase of this window, it MUST abort the whole fragmentation and transmission of this SCHC Packet.

#### 8.5.3. ACK-on-Error

The senders behavior for ACK-on-Error and ACK-Always are similar. The main difference is that in ACK-on-Error the SCHC ACK with the encoded Bitmap is not sent at the end of each window but only when at least one SCHC Fragment of the current window has been lost. Except for the last window where a SCHC ACK MUST be sent to finish the transmission.

In ACK-on-Error, the Retransmission Timer expiration is considered as a positive acknowledgement for all windows but the last one. This timer is set after sending an All-0 or an All-1 fragment. For an All-0 fragment, on timer expiration, the sender resumes operation and sends the SCHC Fragments of the next window.

If the sender receives a SCHC ACK, it checks the window value. SCHC ACKs with an unexpected window number are discarded. If the window number in the received SCHC ACK is correct, the sender verifies if the receiver has received all SCHC fragments of the current window. When at least one SCHC Fragment has been lost, the counter Attempts is increased by one and the sender resends the missing SCHC Fragments again. When Attempts reaches MAX\_ACK\_REQUESTS, the sender sends a Sender-Abort message and releases all resources for the on-going fragmented SCHC Packet transmission. When the retransmission of the missing SCHC Fragments is finished, the sender starts listening for a SCHC ACK (even if an All-0 or an All-1 has not been sent during the retransmission) and initializes the Retransmission Timer.

After sending an All-1 fragment, the sender listens for a SCHC ACK, initializes Attempts, and starts the Retransmission Timer. If the Retransmission Timer expires, Attempts is increased by one and an empty All-1 fragment is sent to request the SCHC ACK for the last window. If Attempts reaches MAX\_ACK\_REQUESTS, the sender aborts the on-going fragmented SCHC Packet transmission by transmitting the Sender-Abort fragment.

At the end of any window, if the sender receives a SCK ACK with a Bitmap containing a bit set for a SCHC Fragment that it has not sent during the transmission phase of that window, it MUST abort the whole fragmentation and transmission of this SCHC Packet.

Unlike the sender, the receiver for ACK-on-Error has a larger amount of differences compared with ACK-Always. First, a SCHC ACK is not

sent unless there is a lost SCHC Fragment or an unexpected behavior. With the exception of the last window, where a SCHC ACK is always sent regardless of SCHC Fragment losses or not. The receiver starts by expecting SCHC Fragments from window 0 and maintains the information regarding which SCHC Fragments it receives. After receiving a SCHC Fragment, the Inactivity Timer is set. If no further SCHC Fragment are received and the Inactivity Timer expires, the SCHC Fragment receiver aborts the on-going fragmented SCHC Packet transmission by transmitting the Receiver-Abort data unit.

Any SCHC Fragment not belonging to the current window is discarded. The actual SCHC Fragment number is computed based on the FCN value. When an All-0 fragment is received and all SCHC Fragments have been received, the receiver updates the expected window value and expects a new window and waits for the next SCHC Fragment. If the window value of the next SCHC Fragment has not changed, the received SCHC Fragment is a retransmission. A receiver that has already received a Fragment discard it. If all SCHC Fragments of a window (that is not the last one) have been received, the receiver does not send a SCHC ACK. While the receiver waits for the next window and if the window value is set to the next value, and if an All-1 fragment with the next value window arrived the receiver knows that the last SCHC Fragment of the packet has been sent. Since the last window is not always full, the MIC will be used to detect if all SCHC Fragments of the window have been received. A correct MIC check indicates the end of the fragmented SCHC Packet transmission. An ACK is sent by the SCHC Fragment receiver. In case of an incorrect MIC, the receiver waits for SCHC Fragments belonging to the same window or the expiration of the Inactivity Timer. The latter will lead the receiver to abort the on-going SCHC fragmented packet transmission by transmitting the Receiver-Abort message.

If, after receiving an All-0 fragment the receiver missed some SCHC Fragments, the receiver uses a SCHC ACK with the encoded Bitmap to ask the retransmission of the missing fragments and expect to receive SCHC Fragments with the actual window. While waiting the retransmission an All-0 empty fragment is received, the receiver sends again the SCHC ACK with the encoded Bitmap, if the SCHC Fragments received belongs to another window or an All-1 fragment is received, the transmission is aborted by sending a Receiver-Abort fragment. Once it has received all the missing fragments it waits for the next window fragments.

#### 8.6. Supporting multiple window sizes

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.

#### 8.7. Downlink SCHC Fragment transmission

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 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 frame that requires an L2 ACK) or it MAY be triggered from an upper layer.

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 other documents (e.g. technology-specific profiles). 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).

#### 9. Padding management

SCHC C/D and SCHC F/R operate on bits, not bytes. SCHC itself does not have any alignment prerequisite. If the Layer 2 below SCHC constrains the L2 Data Unit to align to some boundary, called L2 Words (for example, bytes), SCHC will meet that constraint and produce messages with the correct alignment. This may entail adding extra bits (called padding bits).

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

Padding happens at most once for each Packet going through the full SCHC chain, i.e. Compression and (optionally) SCHC Fragmentation (see Figure 2). If a SCHC Packet is sent unfragmented (see Figure 27), it is padded as needed. If a SCHC Packet is fragmented, only the last fragment is padded as needed.



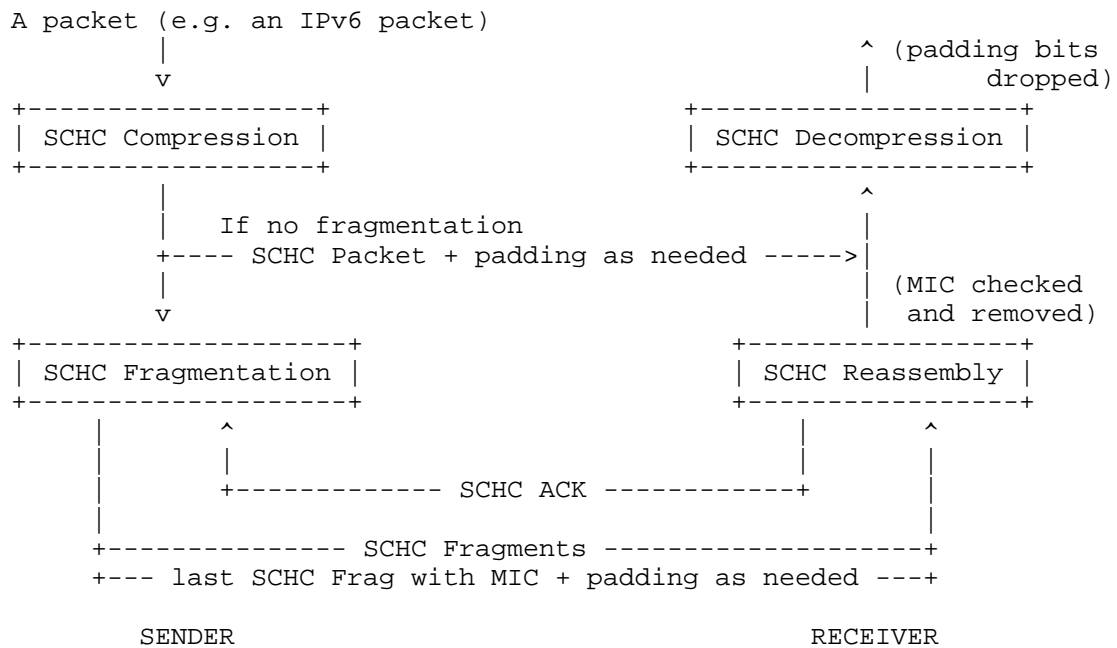


Figure 27: SCHC operations, including padding as needed

Each technology-specific document 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.

## 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, two possibilities can be considered depending on the variability of the value:

- o 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".
- o 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(y).

#### 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:

- o 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".
- o 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(y).

#### 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(s).

#### 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:

- o 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".
- o 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 frame (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 29

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 CDF 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 frame (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 technology-specific documents.

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, Eduardo Ingles Sanchez, Arunprabhu Kandasamy, Rahul Jadhav, Sergio Lopez Bernal, Antony Markovski, Alexander Pelov, Pascal Thubert, Juan Carlos Zuniga, Diego Dujovne, Edgar Ramos, and Shoichi Sakane 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  $\alpha::\text{IID}/64$  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 28 presents the protocol stack for this Device. IPv6 and UDP are represented with dotted lines since these protocols are compressed on the radio link.

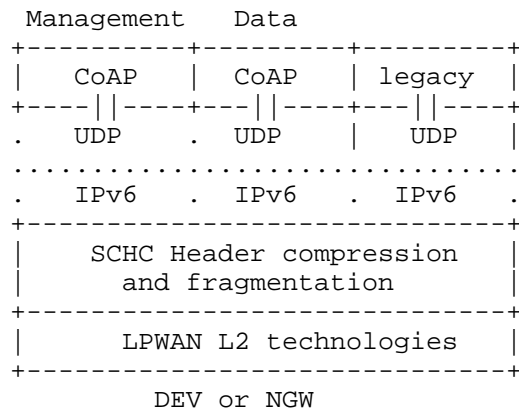


Figure 28: Simplified Protocol Stack for LP-WAN

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	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 29: Context Rules

All the fields described in the three Rules depicted on Figure 29 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 30 illustrates the transmission in No-ACK mode of an IPv6 packet that needs 11 fragments. FCN is 1 bit wide.

Sender	Receiver
-----FCN=0----->	
-----FCN=0----->	
-----FCN=0----->	
-----FCN=0----->	
-----FCN=0----->	
-----FCN=0----->	
-----FCN=0----->	
-----FCN=0----->	
-----FCN=0----->	
-----FCN=0----->	
-----FCN=0----->	
-----FCN=1 + MIC ---->	MIC checked: success =>

Figure 30: Transmission in No-ACK mode of an IPv6 packet carried by 11 fragments

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

Figure 31 illustrates the transmission in ACK-on-Error of an IPv6 packet that needs 11 fragments, with MAX\_WIND\_FCN=6 and no fragment loss.

Sender	Receiver
-----W=0, FCN=6----->	
-----W=0, FCN=5----->	
-----W=0, FCN=4----->	
-----W=0, FCN=3----->	
-----W=0, FCN=2----->	
-----W=0, FCN=1----->	
-----W=0, FCN=0----->	
(no ACK)	
-----W=1, FCN=6----->	
-----W=1, FCN=5----->	
-----W=1, FCN=4----->	
--W=1, FCN=7 + MIC-->	MIC checked: success =>
<----- ACK, W=1 ----->	

Figure 31: Transmission in ACK-on-Error mode of an IPv6 packet carried by 11 fragments, with MAX\_WIND\_FCN=6 and no loss.

Figure 32 illustrates the transmission in ACK-on-Error mode of an IPv6 packet that needs 11 fragments, with MAX\_WIND\_FCN=6 and three lost fragments.

Sender	Receiver
-----W=0, FCN=6----->	
-----W=0, FCN=5----->	
-----W=0, FCN=4--X-->	
-----W=0, FCN=3----->	
-----W=0, FCN=2--X-->	
-----W=0, FCN=1----->	
-----W=0, FCN=0----->	
<-----ACK, W=0----->	7
-----W=0, FCN=4----->	/
-----W=0, FCN=2----->	6543210
	Bitmap:1101011
(no ACK)	
-----W=1, FCN=6----->	
-----W=1, FCN=5----->	
-----W=1, FCN=4--X-->	
- W=1, FCN=7 + MIC ->	MIC checked: failed
<-----ACK, W=1----->	C=0 Bitmap:1100001
-----W=1, FCN=4----->	MIC checked: success =>
<----- ACK, W=1 ----->	C=1, no Bitmap

Figure 32: Transmission in ACK-on-Error mode of an IPv6 packet carried by 11 fragments, with MAX\_WIND\_FCN=6 and three lost fragments.

Figure 33 illustrates the transmission in ACK-Always mode of an IPv6 packet that needs 11 fragments, with MAX\_WIND\_FCN=6 and no loss.

Sender	Receiver
-----W=0, FCN=6----->	
-----W=0, FCN=5----->	
-----W=0, FCN=4----->	
-----W=0, FCN=3----->	
-----W=0, FCN=2----->	
-----W=0, FCN=1----->	
-----W=0, FCN=0----->	
<-----ACK, W=0-----	Bitmap:1111111
-----W=1, FCN=6----->	
-----W=1, FCN=5----->	
-----W=1, FCN=4----->	
--W=1, FCN=7 + MIC-->	MIC checked: success =>
<-----ACK, W=1-----	C=1 no Bitmap
(End)	

Figure 33: Transmission in ACK-Always mode of an IPv6 packet carried by 11 fragments, with MAX\_WIND\_FCN=6 and no lost fragment.

Figure 34 illustrates the transmission in ACK-Always mode of an IPv6 packet that needs 11 fragments, with MAX\_WIND\_FCN=6 and three lost fragments.

Sender	Receiver
-----W=1, FCN=6----->	
-----W=1, FCN=5----->	
-----W=1, FCN=4--X-->	
-----W=1, FCN=3----->	
-----W=1, FCN=2--X-->	7
-----W=1, FCN=1----->	/
-----W=1, FCN=0----->	6543210
<-----ACK, W=1-----	Bitmap:1101011
-----W=1, FCN=4----->	
-----W=1, FCN=2----->	
<-----ACK, W=1-----	Bitmap:
-----W=0, FCN=6----->	
-----W=0, FCN=5----->	
-----W=0, FCN=4--X-->	
--W=0, FCN=7 + MIC-->	MIC checked: failed
<-----ACK, W=0-----	C= 0 Bitmap:11000001
-----W=0, FCN=4----->	MIC checked: success =>
<-----ACK, W=0-----	C= 1 no Bitmap
(End)	

Figure 34: Transmission in ACK-Always mode of an IPv6 packet carried by 11 fragments, with MAX\_WIND\_FCN=6 and three lost fragments.

Figure 35 illustrates the transmission in ACK-Always mode of an IPv6 packet that needs 6 fragments, with MAX\_WIND\_FCN=6, three lost fragments and only one retry needed to recover each lost fragment.

Sender	Receiver
-----W=0, FCN=6----->	
-----W=0, FCN=5----->	
-----W=0, FCN=4--X-->	
-----W=0, FCN=3--X-->	
-----W=0, FCN=2--X-->	
--W=0, FCN=7 + MIC-->	MIC checked: failed
<-----ACK, W=0-----	C= 0 Bitmap:11000001
-----W=0, FCN=4----->	MIC checked: failed
-----W=0, FCN=3----->	MIC checked: failed
-----W=0, FCN=2----->	MIC checked: success
<-----ACK, W=0-----	C=1 no Bitmap
(End)	

Figure 35: Transmission in ACK-Always mode of an IPv6 packet carried by 11 fragments, with MAX\_WIND\_FCN=6, three lost fragments and only one retry needed for each lost fragment.

Figure 36 illustrates the transmission in ACK-Always mode of an IPv6 packet that needs 6 fragments, with MAX\_WIND\_FCN=6, three lost fragments, and the second ACK lost.

	Sender	Receiver
	-----W=0, FCN=6----->	
	-----W=0, FCN=5----->	
	-----W=0, FCN=4--X-->	
	-----W=0, FCN=3--X-->	
	-----W=0, FCN=2--X-->	
	--W=0, FCN=7 + MIC-->	MIC checked: failed
	<-----ACK, W=0-----	C=0 Bitmap:1100001
	-----W=0, FCN=4----->	MIC checked: failed
	-----W=0, FCN=3----->	MIC checked: failed
	-----W=0, FCN=2----->	MIC checked: success
	X---ACK, W=0-----	C= 1 no Bitmap
timeout		
	--W=0, FCN=7 + MIC-->	
	<-----ACK, W=0-----	C= 1 no Bitmap
	(End)	

Figure 36: Transmission in ACK-Always mode of an IPv6 packet carried by 11 fragments, with MAX\_WIND\_FCN=6, three lost fragments, and the second ACK lost.

Figure 37 illustrates the transmission in ACK-Always mode of an IPv6 packet that needs 6 fragments, with MAX\_WIND\_FCN=6, with three lost fragments, and one retransmitted fragment lost again.



Sender	Receiver
-----W=0, FCN=6----->	
-----W=0, FCN=5----->	
-----W=0, FCN=4--X-->	
-----W=0, FCN=3--X-->	
-----W=0, FCN=2--X-->	
--W=0, FCN=7 + MIC-->	MIC checked: failed
<-----ACK, W=0-----	C=0 Bitmap:1100001
-----W=0, FCN=4----->	MIC checked: failed
-----W=0, FCN=3----->	MIC checked: failed
-----W=0, FCN=2--X-->	
timeout	
--W=0, FCN=7 + MIC-->	All-0 empty
<-----ACK, W=0-----	C=0 Bitmap: 1111101
-----W=0, FCN=2----->	MIC checked: success
<-----ACK, W=0-----	C=1 no Bitmap
(End)	

Figure 37: Transmission in ACK-Always mode of an IPv6 packet carried by 11 fragments, with MAX\_WIND\_FCN=6, with three lost fragments, and one retransmitted fragment lost again.

Figure 38 illustrates the transmission in ACK-Always mode of an IPv6 packet that needs 28 fragments, with N=5, MAX\_WIND\_FCN=23 and two lost fragments. Note that MAX\_WIND\_FCN=23 may be useful when the maximum possible Bitmap size, considering the maximum lower layer technology payload size and the value of R, is 3 bytes. Note also that the FCN of the last fragment of the packet is the one with FCN=31 (i.e.  $FCN=2^N-1$  for N=5, or equivalently, all FCN bits set to 1).

Sender	Receiver
-----W=0, FCN=23----->	
-----W=0, FCN=22----->	
-----W=0, FCN=21--X-->	
-----W=0, FCN=20----->	
-----W=0, FCN=19----->	
-----W=0, FCN=18----->	
-----W=0, FCN=17----->	
-----W=0, FCN=16----->	
-----W=0, FCN=15----->	
-----W=0, FCN=14----->	
-----W=0, FCN=13----->	
-----W=0, FCN=12----->	
-----W=0, FCN=11----->	
-----W=0, FCN=10--X-->	
-----W=0, FCN=9 ----->	
-----W=0, FCN=8 ----->	
-----W=0, FCN=7 ----->	
-----W=0, FCN=6 ----->	
-----W=0, FCN=5 ----->	
-----W=0, FCN=4 ----->	
-----W=0, FCN=3 ----->	
-----W=0, FCN=2 ----->	
-----W=0, FCN=1 ----->	
-----W=0, FCN=0 ----->	
<-----ACK, W=0----->	lcl-Bitmap:11011111111111011111111111
-----W=0, FCN=21----->	encoded Bitmap:11011111111111011
-----W=0, FCN=10----->	
<-----ACK, W=0----->	no Bitmap
-----W=1, FCN=23----->	
-----W=1, FCN=22----->	
-----W=1, FCN=21----->	
--W=1, FCN=31 + MIC-->	MIC checked: sucess =>
<-----ACK, W=1----->	no Bitmap

(End)

Figure 38: Transmission in ACK-Always mode of an IPv6 packet carried by 28 fragments, with N=5, MAX\_WIND\_FCN=23 and two lost 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:

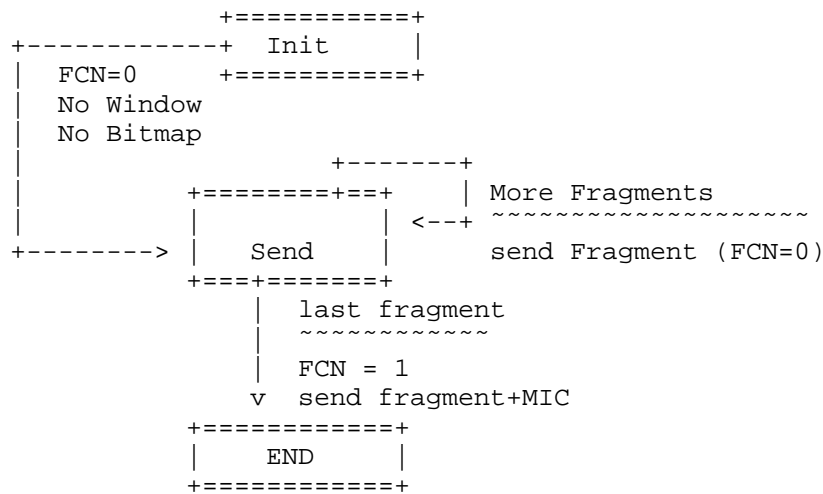


Figure 39: Sender State Machine for the No-ACK Mode

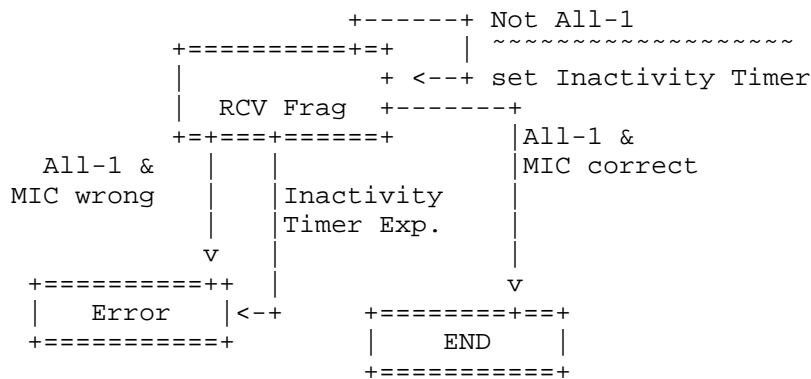


Figure 40: Receiver State Machine for the No-ACK Mode

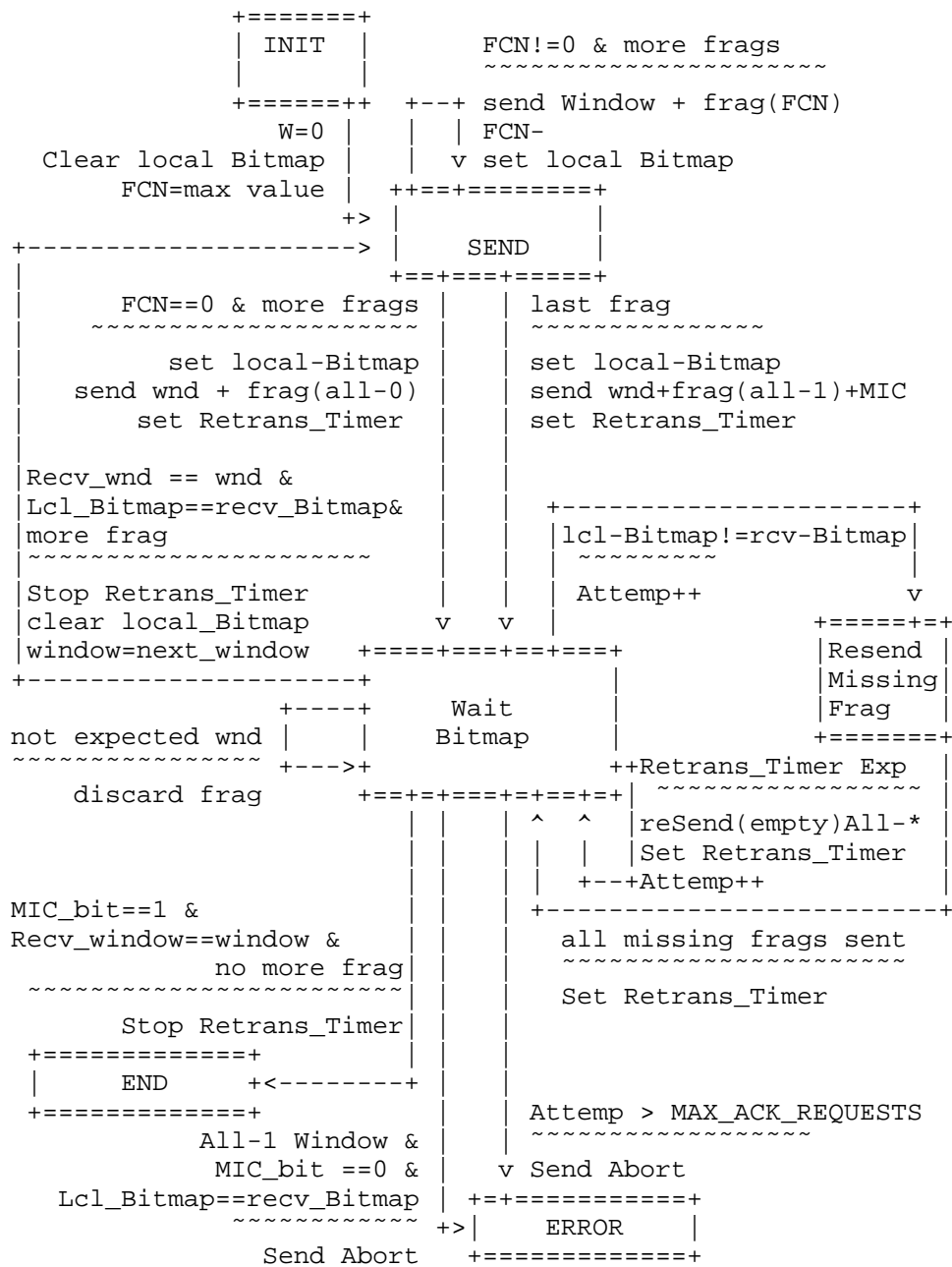
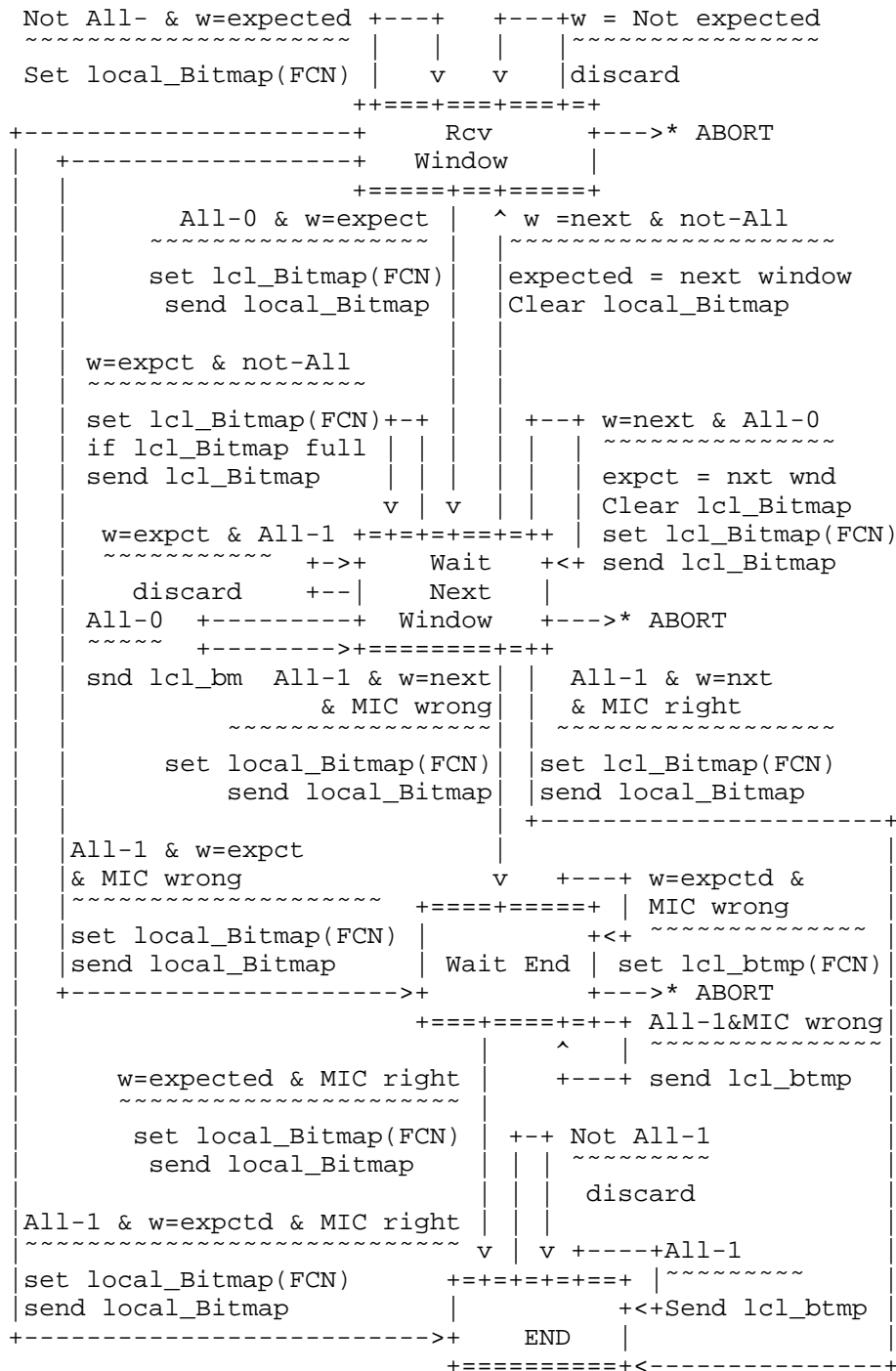


Figure 41: Sender State Machine for the ACK-Always Mode



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

Figure 42: Receiver State Machine for the ACK-Always Mode

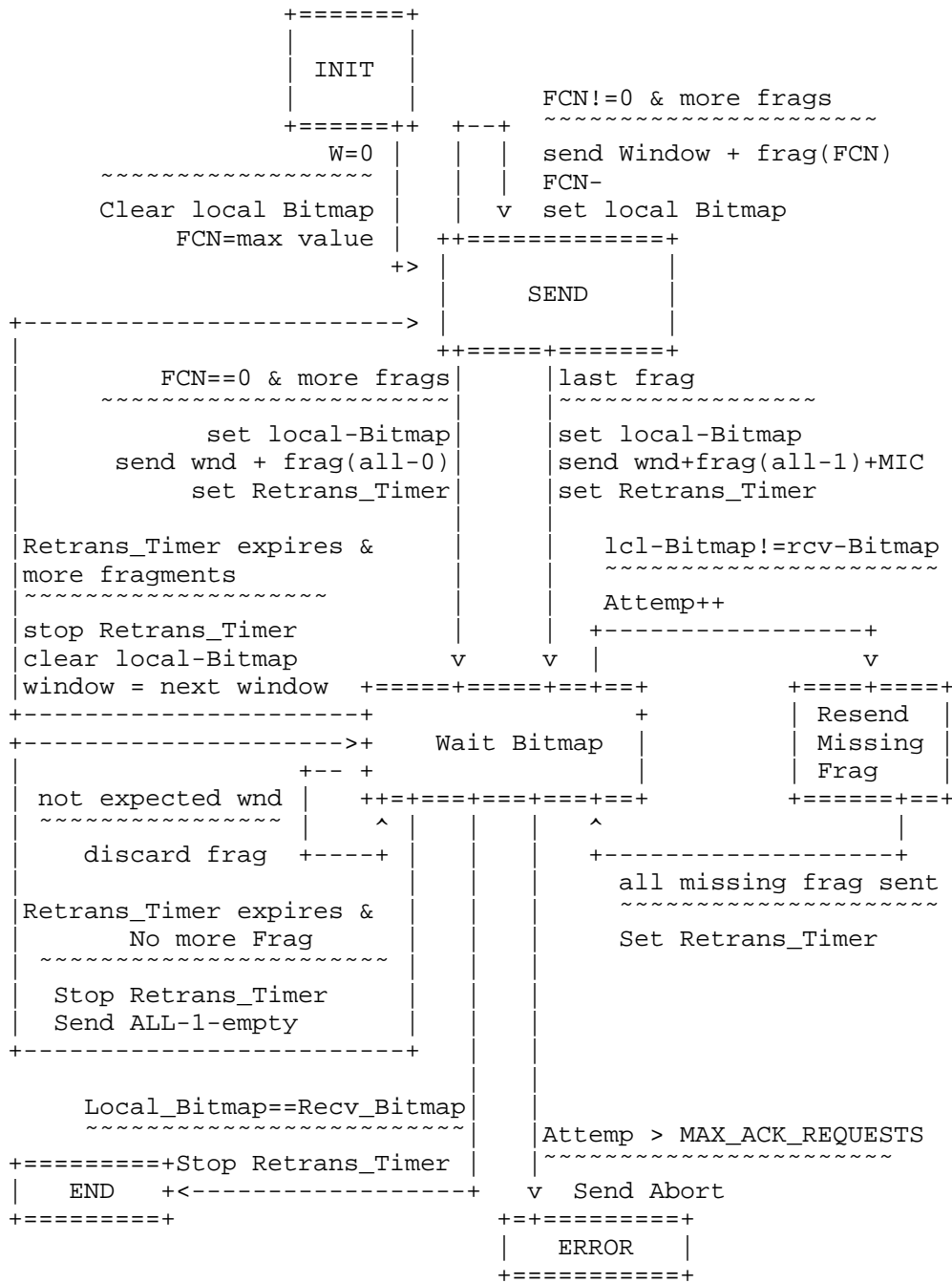


Figure 43: Sender State Machine for the ACK-on-Error Mode

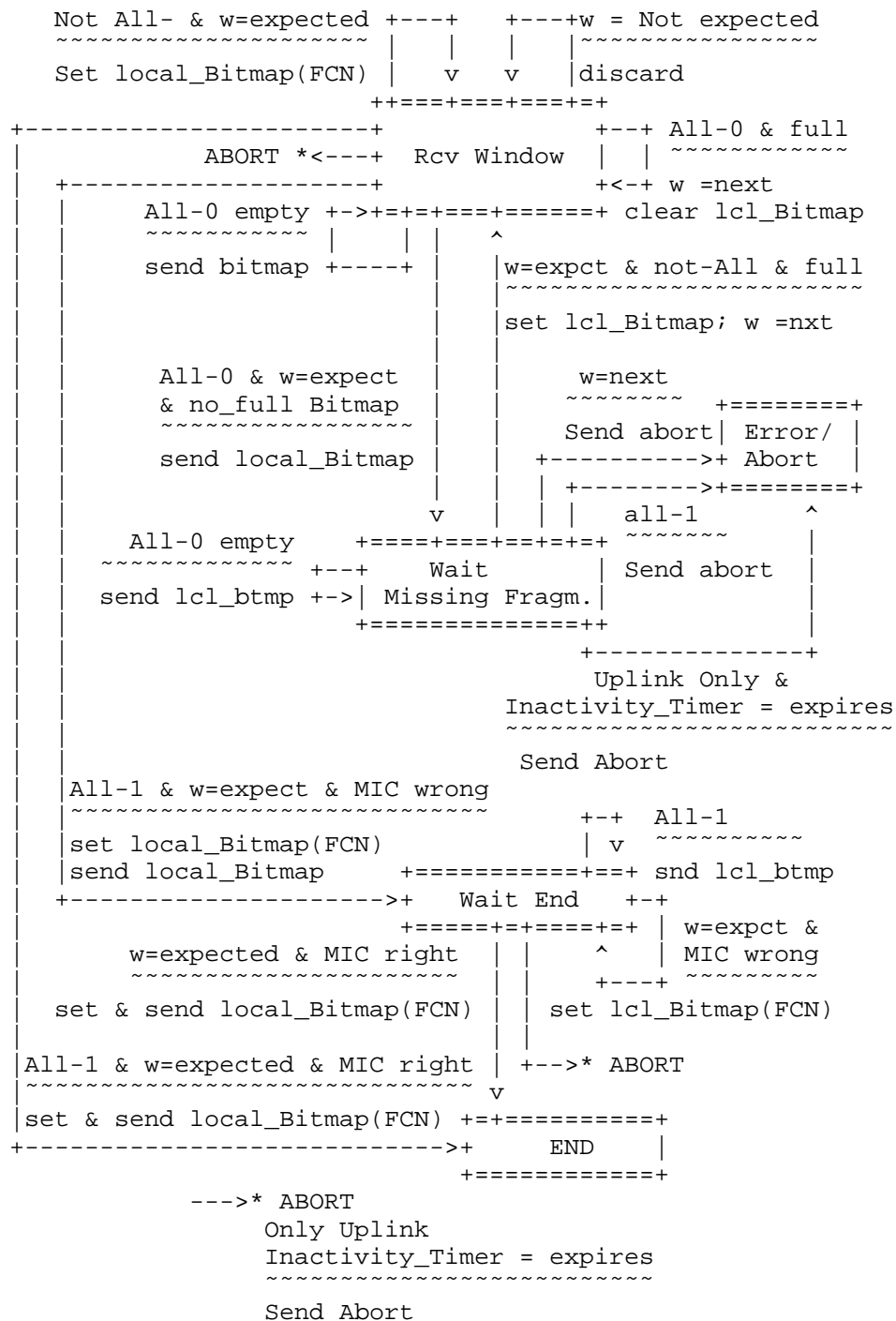




Figure 44: Receiver State Machine for the ACK-on-Error Mode

## Appendix D. SCHC Parameters - Ticket #15

This section gives the list of parameters that need to be defined in the technology-specific documents.

- o Define the most common uses case and how SCHC may be deployed.
- o LPWAN Architecture. Explain the SCHC entities (Compression and Fragmentation), how/where they are represented in the corresponding technology architecture. If applicable, explain the various potential channel conditions for the technology and the corresponding recommended use of C/D and F/R.
- o L2 fragmentation decision
- o Technology developers must evaluate that L2 has strong enough integrity checking to match SCHC's assumption.
- o Rule ID numbering system, number of Rules
- o Size of the Rule IDs
- o The way the Rule ID is sent (L2 or L3) and how (describe)
- o Fragmentation delivery reliability mode used in which cases (e.g. based on link channel condition)
- o Define the number of bits for FCN (N) and DTag (T)
- o in particular, is interleaved packet transmission supported and to what extent
- o The MIC algorithm to be used and the size, if different from the default CRC32
- o Retransmission Timer duration
- o Inactivity Timer duration
- o Define MAX\_ACK\_REQUEST (number of attempts)
- o Padding: size of the L2 Word (for most technologies, a byte; for some technologies, a bit). Value of the padding bits (1 or 0). The value of the padding bits needs to be specified because the padding bits are included in the MIC calculation.

- o Take into account that the length of Rule ID + N + T + W when possible is good to have a multiple of 8 bits to complete a byte and avoid padding
- o In the ACK format to have a length for Rule ID + T + W bit into a complete number of byte to do optimization more easily
- o The technology documents will describe if Rule ID is constrained by any alignment
- o When fragmenting in ACK-on-Error or ACK-Always mode, it is expected that the last window (called All-1 window) will not be fully utilised, i.e. there won't be fragments with all FCN values from MAX\_WIND\_FCN down to 1 and finally All-1. It is worth noting that this document does not mandate that other windows (called All-0 windows) are fully utilised either. This document purposely does not specify that All-1 windows use Bitmaps with the same number of bits as All-0 windows do. By default, Bitmaps for All-0 and All-1 windows are of the same size MAX\_WIND\_FCN + 1. But a technology-specific document MAY revert that decision. The rationale for reverting the decision could be the following: Note that the SCHC ACK sent as a response to an All-1 fragment includes a C bit that SCHC ACK for other windows don't have. Therefore, the SCHC ACK for the All-1 window is one bit bigger. An L2 technology with a severely constrained payload size might decide that this "bump" in the SCHC ACK for the last fragment is a bad resource usage. It could thus mandate that the All-1 window is not allowed to use the FCN value 1 and that the All-1 SCHC ACK Bitmap size is reduced by 1 bit. This provides room for the C bit without creating a bump in the SCHC ACK.

And the following parameters need to be addressed in another document but not forcibly in the technology-specific one:

- o The way the contexts are provisioning
- o The way the Rules are generated

#### Appendix E. 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

lpwan Working Group  
Internet-Draft  
Intended status: Informational  
Expires: March 8, 2019

A. Minaburo  
Acklio  
E. Ramos  
Ericsson  
S. Shanmugalingam  
Acklio  
September 04, 2018

LPWAN Static Context Header Compression (SCHC) over NB-IoT  
draft-minaburo-lpwan-nbiot-hc-01

Abstract

The Static Context Header Compression (SCHC) specification describes a header compression and fragmentation functionalities for LPWAN (Low Power Wide Area Networks) technologies. SCHC was designed to be adapted over any of the LPWAN technologies.

This document describes the use of SCHC over the NB-IoT wireless access, and provides elements for an efficient parameterization.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 8, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Architecture . . . . .	4
3.1. Data Transmission . . . . .	5
3.2. Data Transmission over User Plane . . . . .	6
3.2.1. Packet Data Convergence Protocol (PDCP) . . . . .	7
3.2.2. Radio Link Protocol (RLC) . . . . .	7
3.2.3. Medium Access Control (MAC) . . . . .	8
3.3. Data Over Control Plane . . . . .	9
3.4. SCHC entities . . . . .	13
4. Static Context Header Compression . . . . .	13
4.1. SCHC Rules . . . . .	13
4.2. Packet processing . . . . .	13
4.3. SCHC Context . . . . .	13
5. Fragmentation . . . . .	13
5.1. Fragmentation modes . . . . .	14
5.2. Fragmentation Parameters . . . . .	14
6. Padding . . . . .	14
7. Security considerations . . . . .	14
8. Appendix . . . . .	14
8.1. NB-IoT example with mobility . . . . .	15
9. Informative References . . . . .	15
Authors' Addresses . . . . .	16

## 1. Introduction

The Static Context Header Compression (SCHC) [I-D.ietf-lpwan-ipv6-static-context-hc] defines a header compression scheme and fragmentation functionality, both specially tailored for Low Power Wide Area Networks (LPWAN) networks defined in [I-D.ietf-lpwan-overview].

Header compression is needed to efficiently bring Internet connectivity to the node within an NB-IoT network. SCHC uses an static context to performs header compression with specific parameters that need to be adapted into the NB-IoT wireless access. This document assumes functionality for NB-IoT of 3GPP release 15 otherwise other versions functionality is explicitly mentioned in the text.

This document describes the use of SCHC and its parameterizing over the NB-IoT wireless access.

## 2. Terminology

This document will follow the terms defined in [I-D.ietf-lpwan-ipv6-static-context-hc], in [I-D.ietf-lpwan-overview], and the TGPP23720.

- o CIoT. Cellular IoT
- o C-SGN. CIoT Serving Gateway Node
- o UE. User Equipment
- o eNB. Node B. Base Station that controls the UE
- o EPC. Evolved Packet Connectivity. Core network of 3GPP LTE systems.
- o EUTRAN. Evolved Universal Terrestrial Radio Access Network. Radio network from LTE based systems.
- o MME. Mobility Management Entity. Handle mobility of the UE
- o NB-IoT. Narrow Band IoT. Referring to 3GPP LPWAN technology based in LTE architecture but with additional optimization for IoT and using a Narrow Band spectrum frequency.
- o SGW. Serving Gateway. Routes and forwards the user data packets through the access network
- o HSS. Home Subscriber Server. It is a database that performs mobility management
- o PGW. Packet Data Node Gateway. Interface between the internal with the external network
- o PDU. Protocol Data Unit. Data packets including headers that are transmitted between entities through a protocol.
- o SDU. Service Data Unit. Data packets (PDUs) from higher layers protocols used by lower layer protocols as payload of their own PDUs that has not yet been encapsulated.
- o IWK-SCEF. InterWorking Service Capabilities Exposure Function. Used in roaming scenarios and serves for interconnection with the SCEF of the Home PLMN and is located in the Visited PLMN

- o SCEF. Service Capability Exposure Function. EPC node for exposure of 3GPP network service capabilities to 3rd party applications.

### 3. Architecture

#### ## NB-IoT entities

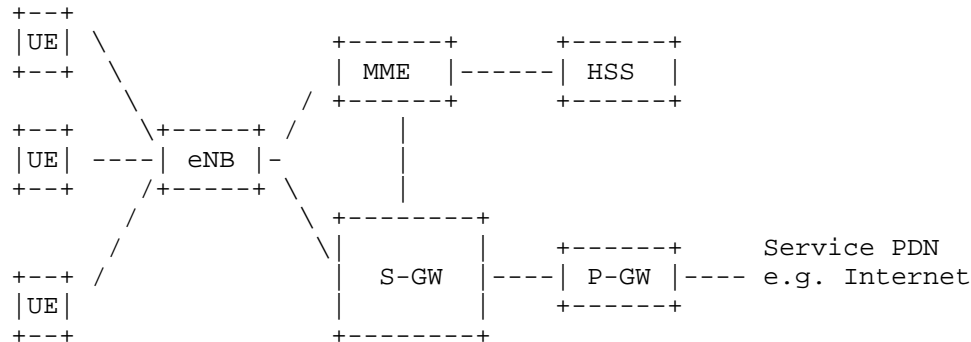


Figure 1: 3GPP network architecture

The architecture for 3GPP LTE network has been reused for NB-IoT with some optimizations and simplifications known as Cellular IoT (CIoT). Considering the typical use cases for CIoT devices here are described some of the additions to the LTE architecture specific for CIoT. C-SGN(CIoT Serving Gateway Node) is a deployment option co-locating EPS entities in the control plane and user plane paths (for example, MME + SGW + P-GW) and the external interfaces of the entities supported. The C-SGN also supports at least some of the following CIoT EPS Optimizations: \* Control Plane CIoT EPS Optimization for small data transmission. \* User Plane CIoT EPS Optimization for small data transmission. \* Necessary security procedures for efficient small data transmission. \* SMS without combined attach for NB-IoT only UEs. \* Paging optimizations for coverage enhancements. \* Support for non-IP data transmission via SGi tunneling and/or SCEF. \* Support for Attach without PDN (Packet Data Network) connectivity.

Another node introduced in the CIoT architecture is the SCEF (Service Capability Exposure Function) that provide means to securely expose service and network capabilities to entities external to the network operator. The northbound APIs are defined by OMA and OneM2M. The main functions of a SCEF are: \* Non-IP Data Delivery (NIDD) established through the SCEF. \* Monitoring and exposure of event related to UE reachability, loss of connectivity, location reporting,

roaming status, communication failure and change of IMEI-IMSI association.

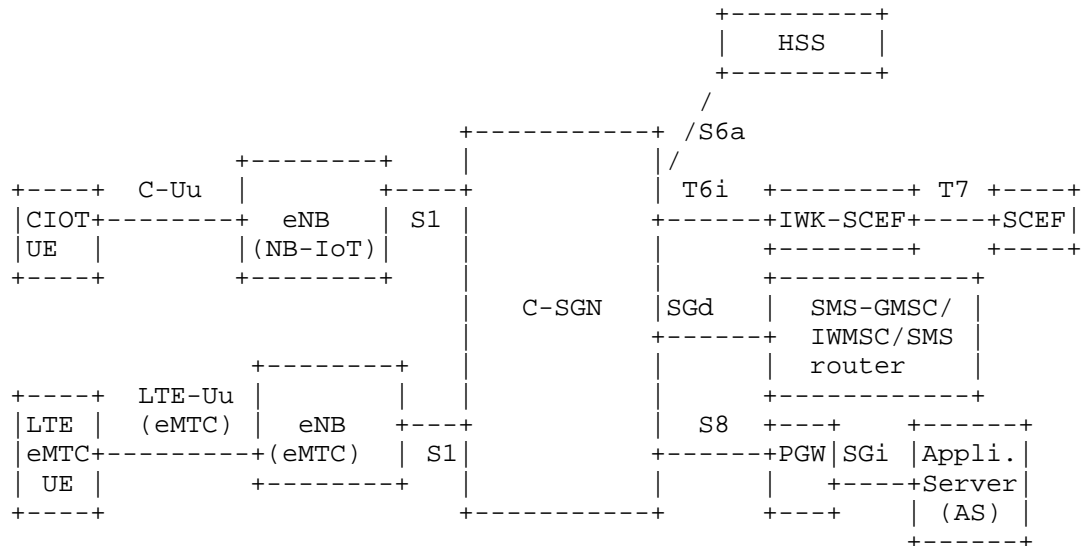


Figure 2: 3GPP optimized CIOT network architecture

### 3.1. Data Transmission

3GPP networks deals not only with data transmitted end-to-end but also with in-band signaling that is used between the nodes and functions to configure, control and monitor the system functions and behaviors. The control data is handled using a Control Plane which has an specific set of protocols, handling processes and entities. In contrast the end-to-end or user data utilize a User Plane with characteristics of its own separated from the Control Plane. The handling and setup of the Control Plane and User Plane spans over the whole 3GPP network and it has particular implications in the radio network (i.e., EUTRAN) and in the packet core (ex., EPC).

For the CIOT cases, additionally to transmissions of data over User Plane, 3GPP has specified optimizations for small data transmissions that allows to transport user data (IP, Non-IP) within signaling on the access network (Data transmission over Control Plane or Data Over NAS).

The maximum recommended MTU size is 1358 Bytes. The radio network protocols limits the packet sizes to be transmitted over the air



including radio protocol overhead to 1600 Octets. But the value is reduced further to avoid fragmentation in the backbone of the network due to the payload encryption size (multiple of 16) and handling of the additional core transport overhead.

### 3.2. Data Transmission over User Plane

The User Plane utilizes the protocol stack of the Access Stratum (AS) for data transfer. AS (Access Stratum) is the functional layer responsible for transporting data over wireless connection and managing radio resources. The user plane AS has support for features such as reliability, segmentation and concatenation. The transmissions of the AS utilize link adaptation, meaning that the transport format utilized for the transmissions are optimized according to the radio conditions, the number of bits to transmit and the power and interference constrains. That means that the number of bits transmitted over the air depends of the Modulation and Coding Schemes (MCS) selected. The transmissions in the physical layer happens at network synchronized intervals of times called TTI (Transmission Time Interval). The transmission of a Transport Block (TB) is completed during, at least, one TTI. Each Transport Block has a different MCS and number of bits available to transmit. The Transport Blocks characteristics are defined by the MAC technical specification {TGPP36321}. The Access Stratum for User Plane is comprised by Packet Data Convergence Protocol (PDCP) {TGPP36323}, Radio Link Protocol (RLC){TGPP36322}, Medium Access Control protocol (MAC){TGPP36321} and the Physical Layer {TGPP36201}.

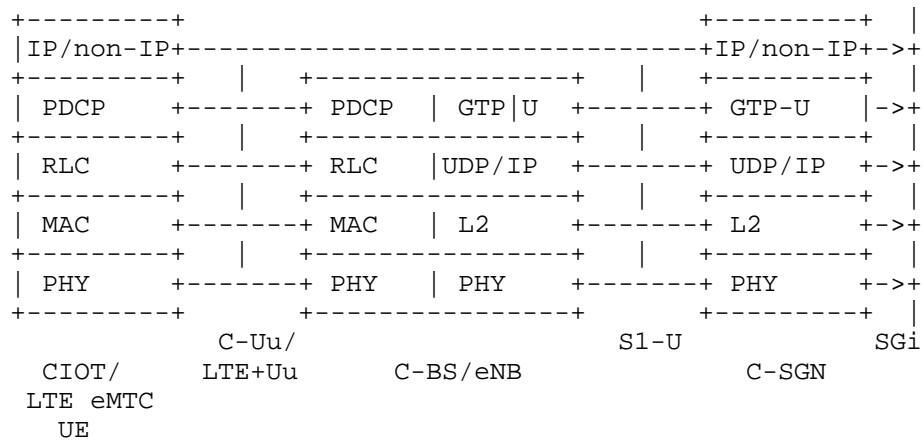


Figure 3: 3GPP CIOT radio protocol architecture for data over user plane

### 3.2.1. Packet Data Convergence Protocol (PDCP)

Each of the Radio Bearers (RB) are associated with one PDCP entity. And a PDCP entity is associated with one or two RLC entities depending of the unidirectional or bi-directional characteristics of the RB and RLC mode used. A PDCP entity is associated either control plane or user plane which independent configuration and functions. The maximum supported size for NB-IoT of a PDCP SDU is 1600 octets. The main services and functions of the PDCP sublayer for NB-IoT for the user plane include: \* Header compression and decompression by means of ROHC (Robust Header Compression) \* Transfer of user and control data to higher and lower layers \* Duplicate detection of lower layer SDUs when re-establishing connection (when RLC with Acknowledge Mode in use for User Plane only) \* Ciphering and deciphering \* Timer-based SDU discard in uplink

### 3.2.2. Radio Link Protocol (RLC)

RLC is a layer-2 protocol that operates between the UE and the base station (eNB). It supports the packet delivery from higher layers to MAC creating packets that are transmitted over the air optimizing the Transport Block utilization. RLC flow of data packets is unidirectional and it is composed of a transmitter located in the transmission device and a receiver located in the destination device. Therefore to configure bi-directional flows, two set of entities, one in each direction (downlink and uplink) must be configured and they are effectively peered to each other. The peering allows the transmission of control packets (ex., status reports) between entities. RLC can be configured for data transfer in one of the following modes: \* Transparent Mode (TM). In this mode RLC do not segment or concatenate SDUs from higher layers and do not include any header to the payload. When acting as a transmitter, RLC receives SDUs from upper layers and transmit directly to its flow RLC receiver via lower layers. Similarly, an TM RLC receiver would only deliver without additional processing the packets to higher layers upon reception. \* Unacknowledged Mode (UM). This mode provides support for segmentation and concatenation of payload. The size of the RLC packet depends of the indication given at a particular transmission opportunity by the lower layer (MAC) and are octets aligned. The packet delivery to the receiver do not include support for reliability and the lost of a segment from a packet means a whole packet loss. Also in case of lower layer retransmissions there is no support for re-segmentation in case of change of the radio conditions triggering the selection of a smaller transport block. Additionally it provides PDU duplication detection and discard, reordering of out of sequence and loss detection. \* Acknowledged Mode (AM). Additional to the same functions supported from UM, this mode also adds a moving windows based reliability service on top of the lower

layer services. It also provides support for re-segmentation and it requires bidirectional communication to exchange acknowledgment reports called RLC Status Report and trigger retransmissions is needed. Protocol error detection is also supported by this mode. The mode uses depends of the operator configuration for the type of data to be transmitted. For example, data transmissions supporting mobility or requiring high reliability would be most likely configured using AM, meanwhile streaming and real time data would be map to a UM configuration.

### 3.2.3. Medium Access Control (MAC)

MAC provides a mapping between the higher layers abstraction called Logical Channels comprised by the previously described protocols to the Physical layer channels (transport channels). Additionally, MAC may multiplex packets from different Logical Channels and prioritize what to fit into one Transport Block if there is data and space available to maximize the efficiency of data transmission. MAC also provides error correction and reliability support by means of HARQ, transport format selection and scheduling information reporting from the terminal to the network. MAC also adds the necessary padding and piggyback control elements when possible additional to the higher layers data.

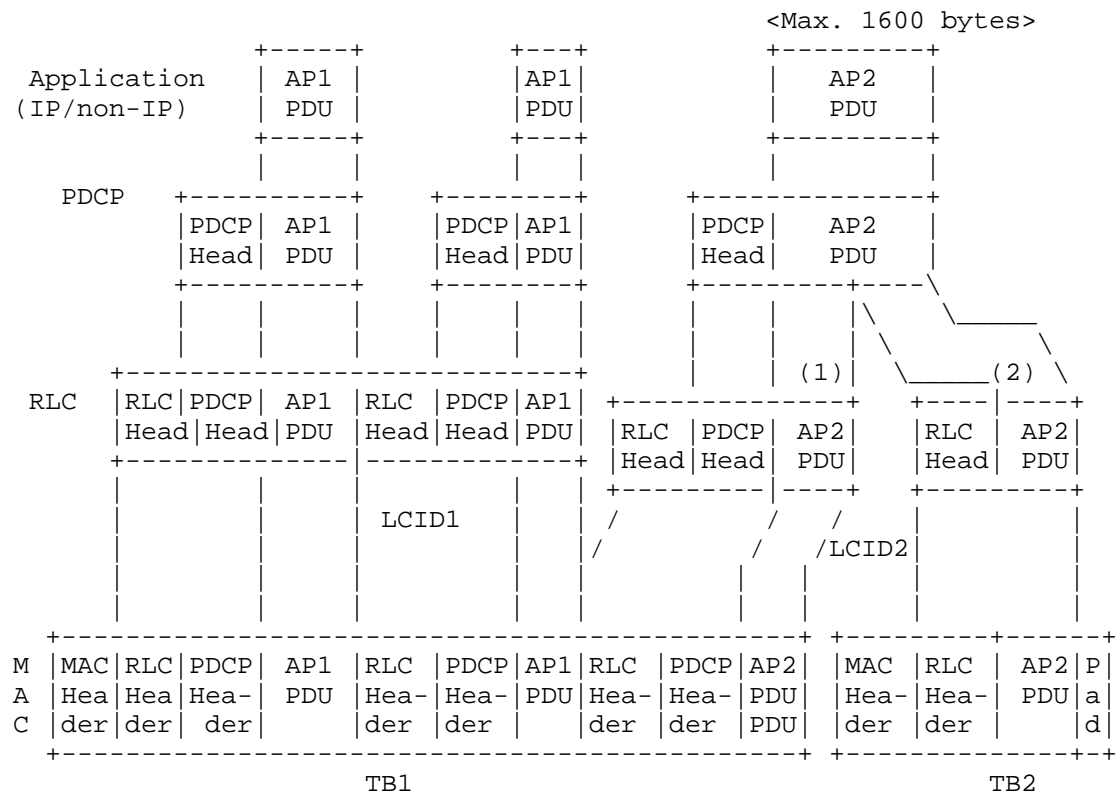
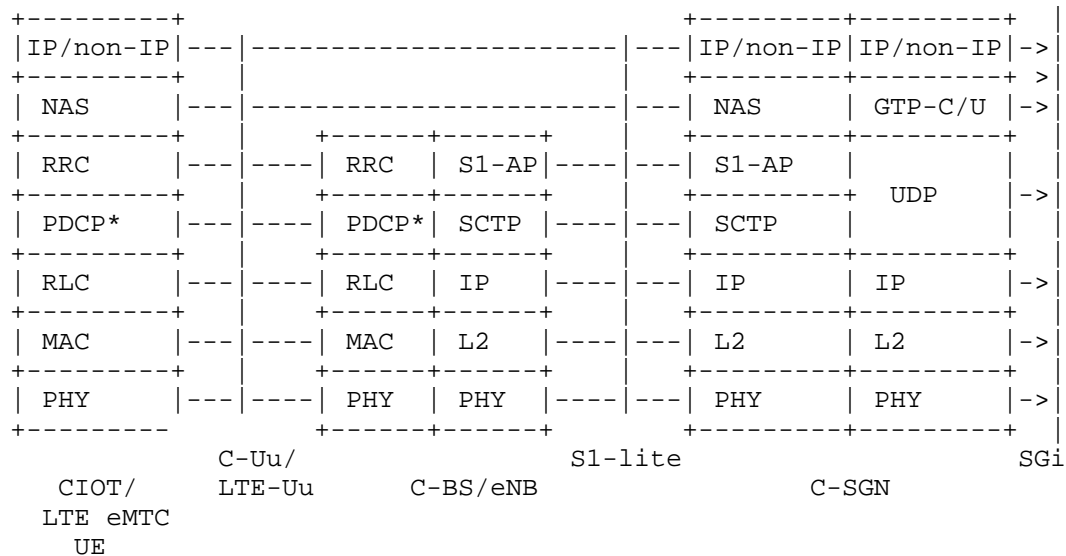


Figure 4: Example of User Plane packet encapsulation for two transport blocks

### 3.3. Data Over Control Plane

The Non-Access Stratum (NAS), conveys mainly control signaling between the UE and the cellular network [TGPP24301]. NAS is transported on top of the Access Stratum (AS) already presented in the previous sections.



\*PDCP is bypassed until AS security is activated TGPP36300.

Figure 5: 3GPP CIOT radio protocol architecture for DoNAS transmissions

NAS has been adapted to provide support for user plane data transmissions to reduce the overhead when transmitting infrequent small quantities of data. This is known as Data over NAS (DoNAS) or Control Plane CIoT EPS optimization. In DoNAS the UE makes use of the pre-established NAS security and piggyback uplink small data into the initial NAS uplink message, and uses an additional NAS message to receive downlink small data response. The data encryption from the network side is performed by the C-SGN in a NAS PDU. The AS protocol stack used by DoNAS is somehow special. Since the security associations are not established yet in the radio network, to reduce the protocol overhead, PDCP (Packet Data Convergence Protocol) is bypassed until AS security is activated. RLC (Radio Link Control protocol) is configured by default in AM mode, but depending of the features supported by the network and the terminal it may be configured in other modes by the network operator. For example, the transparent mode does not add any header or does not process the payload in any way reducing the overhead, but the MTU would be limited by the transport block used to transmit the data which is couple of thousand of bits maximum. If UM (only Release 15 compatible terminals) is used, the RLC mechanisms of reliability is disabled and only the reliability provided by the MAC layer by Hybrid Automatic Repeat request (HARQ) is available. In this case, the protocol overhead might be smaller than for the AM case because the

lack of status reporting but with the same support for segmentation up to 16000 Bytes. NAS packet are encapsulated within a RRC (Radio Resource Control){TGPP36331} message.

Depending of the data type indication signaled (IP or non-IP data), the network allocates an IP address or just establish a direct forwarding path. DoNAS is regulated under rate control upon previous agreement, meaning that a maximum number of bits per unit of time is agreed per device subscription beforehand and configured in the device. The use of DoNAS is typically expected when a terminal in a power saving state requires to do a short transmission and receive an acknowledgment or short feedback from the network. Depending of the size of buffered data to transmit, the UE might be instructed to deploy the connected mode transmissions instead, limiting and controlling the DoNAS transmissions to predefined thresholds and a good resource optimization balance for the terminal and the network. The support for mobility of DoNAS is present but produces additional overhead.

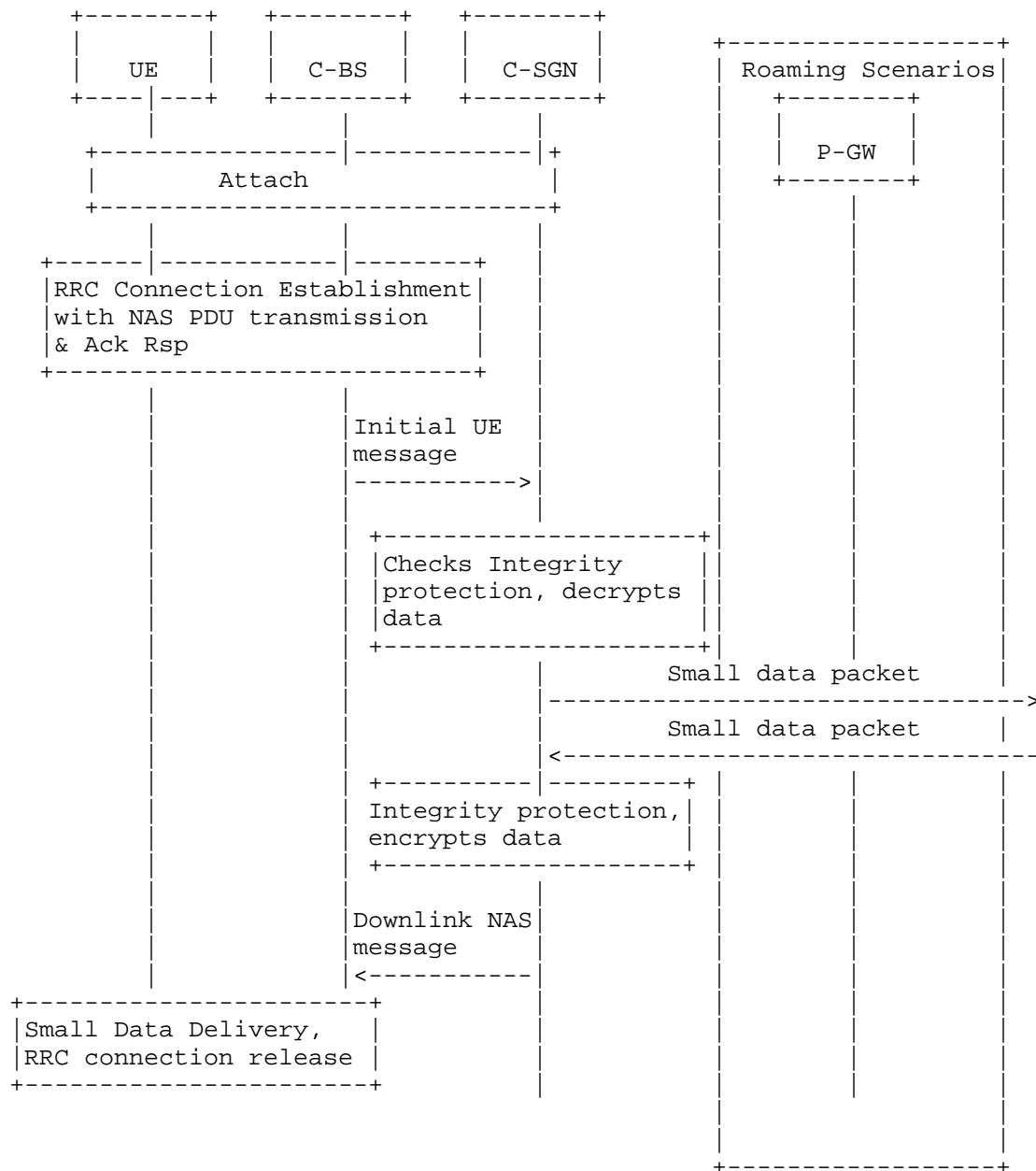


Figure 6: DoNAS transmission sequence from an Uplink initiated access

### 3.4. SCHC entities

SCHC could be deployed differently depending of the placing of the entities in the architecture. NB-IoT and in general the cellular technologies interfaces are standardized by 3GPP. Therefore the introduction of SCHC entities in the RAN (Radio Access Network) would require support from both the network and terminal entities. If SCHC entities are to be placed in RAN it would require to be added to be specified as an option for the UE - Base Station/C-SGN interfaces. Another option is to place the SCHC entities in the applications layer, and the SCHC packets would be transmitted as non-IP packets. The first option allows the deployment of IP for routing and addressing as well.

## 4. Static Context Header Compression

TBD (Edgar)

### 4.1. SCHC Rules

TBD (Ana) \* Depending of SCHC deployment case \* End-2-end \* Global rules to fetch customized rules \* Minimum rule set for applying functions \* Fragmentation, compression, NATing

- o Size of rule id
- o 1 fragment rule And at least one Rule ID may be reserved to the case where no SCHC C/D nor SCHC fragmentation were possible.

### 4.2. Packet processing

(Ana) \*Operation over top vs 3gpp entities how to recognize a schc packet

### 4.3. SCHC Context

- o NATing
- o What protocols can be identified for compression depending of the deployment

## 5. Fragmentation

The RLC layer of NB-IoT can segment packets in suitable units that fits the selected transport blocks for transmissions of the physical layer. The selection of the blocks is done according to the input of the link adaptation function in the MAC layer and the quantity of data in the buffer. The link adaptation layer may produce different



results at each Time Transmission Interval (TTI) for what is very difficult to set a fragmentation value according to the transport block that is selected for each transmission. Instead for NB-IoT SCHC must take care of keeping the application packets with a suitable size that do not exceed the MTU (1600 bytes).

#### 5.1. Fragmentation modes

(Sothy) Look at the different options of reliability and see the implications for NB-IoT for the different deployment modes

#### 5.2. Fragmentation Parameters

(Edgar) Headers sizes Example for the end2end case and check what is operator defined \* Rule ID

- o DTag
- o FCN
- o Retransmission Timer
- o Inactivity Timer
- o MAX\_ACK\_Retries
- o MAX\_ATTEMPS
- o MIC (Ana)

TBD

#### 6. Padding

NB-IoT and 3GPP wireless access in general assumes byte aligned payload. Therefore the L2 word for NB-IoT MUST be considered 8 bits and the treatment of padding should use this value accordingly.

#### 7. Security considerations

3GPP access security is specified in [TGPP33203].

#### 8. Appendix

## NB-IoT with data over NAS

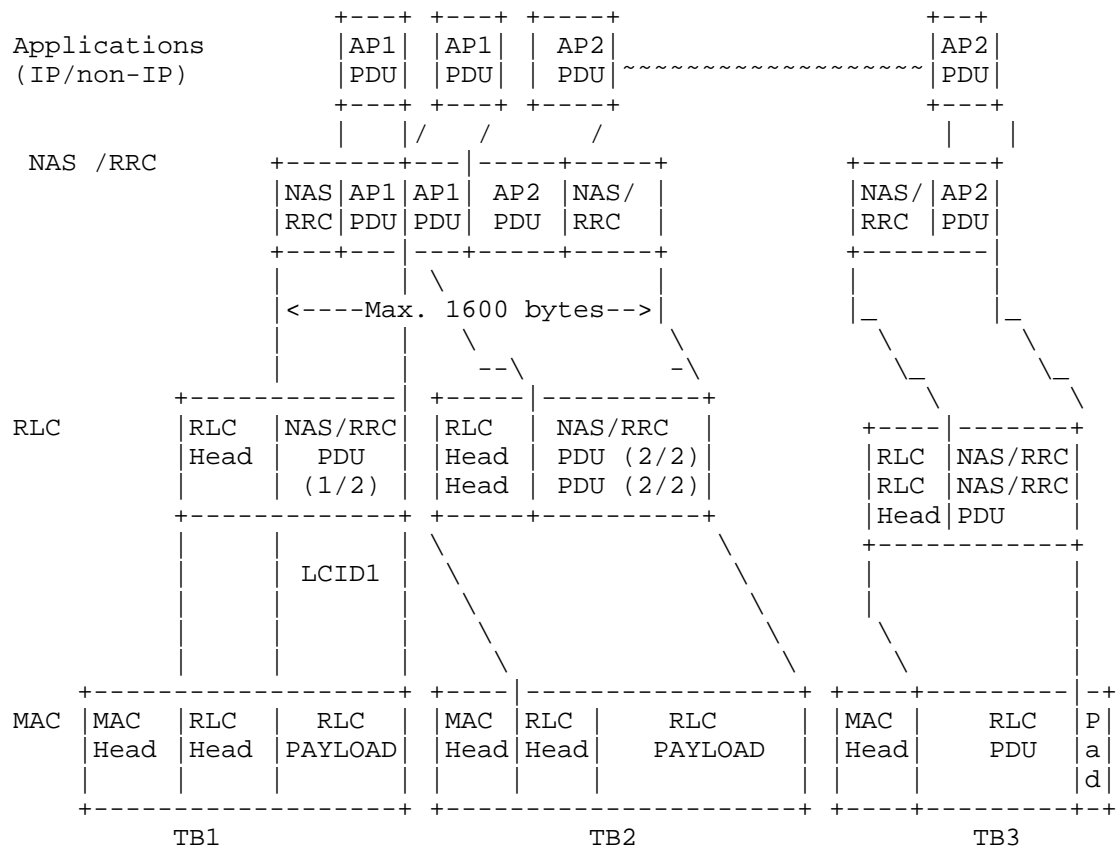


Figure 7: Example of User Plane packet encapsulation for Data over NAS

### 8.1. NB-IoT example with mobility

## LTE-M considerations

## 9. Informative References

- [I-D.ietf-lpwan-ipv6-static-context-hc]  
 Minaburo, A., Toutain, L., Gomez, C., and D. Barthel,  
 "LPWAN Static Context Header Compression (SCHC) and  
 fragmentation for IPv6 and UDP", draft-ietf-lpwan-ipv6-  
 static-context-hc-16 (work in progress), June 2018.

[I-D.ietf-lpwan-overview]

Farrell, S., "LPWAN Overview", draft-ietf-lpwan-overview-10 (work in progress), February 2018.

[TGPP24301]

"TS 24.301 v15.2.0 - Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3", n.d..

[TGPP33203]

"TS 33.203 v13.1.0 - 3G security; Access security for IP-based services", n.d..

[TGPP36300]

"TS 36.300 v15.1.0 - Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2", n.d..

[TGPP36321]

"TS 36.321 v13.2.0 - Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification", n.d..

[TGPP36323]

"TS 36.323 v13.2.0 - Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) specification", n.d..

[TGPP36331]

"TS 36.331 v13.2.0 - Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification", n.d..

#### Authors' Addresses

Ana Minaburo  
Acklio  
2bis rue de la Chataigneraie  
35510 Cesson-Sevigne Cedex  
France

Email: ana@ackl.io

Edgar Ramos  
Ericsson  
Hirsalantie 11  
02420 Jorvas  
Finland

Email: edgar.ramos@ericsson.com

Sivasothy Shanmugalingam  
Acklio  
2bis rue de la Chataigneraie  
35510 Cesson-Sevigne Cedex  
France

Email: sothy@ackl.io

lpwan Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 3, 2019

N. Sornin, Ed.  
M. Coracin  
Semtech  
I. Petrov  
Acklio  
A. Yegin  
Actility  
J. Catalano  
Kerlink  
V. Audebert  
EDF R&D  
July 02, 2018

Static Context Header Compression (SCHC) over LoRaWAN  
draft-petrov-lpwan-ipv6-schc-over-lorawan-02

Abstract

The Static Context Header Compression (SCHC) specification describes generic header compression and fragmentation techniques for LPWAN (Low Power Wide Area Networks) technologies. SCHC is a generic mechanism designed for great flexibility, so that it can be adapted for any of the LPWAN technologies.

This document provides the adaptation of SCHC for use in LoRaWAN networks, and provides elements such as efficient parameterization and modes of operation.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Static Context Header Compression Overview . . . . .	3
4. LoRaWAN Architecture . . . . .	4
4.1. Device classes (A, B, C) and interactions . . . . .	5
4.2. Device addressing . . . . .	6
4.3. General Message Types . . . . .	6
4.4. LoRaWAN MAC Frames . . . . .	6
5. SCHC over LoRaWAN . . . . .	6
5.1. Rule ID management . . . . .	6
5.2. IID computation . . . . .	6
5.3. Fragmentation . . . . .	6
5.3.1. Reliability options . . . . .	6
5.3.2. Supporting multiple window sizes . . . . .	11
5.3.3. Downlink fragment transmission . . . . .	11
5.3.4. SCHC behavior for devices in class A, B and C . . . . .	11
6. Security considerations . . . . .	11
7. Acknowledgements . . . . .	11
8. References . . . . .	11
8.1. Normative References . . . . .	11
8.2. Informative References . . . . .	12
Appendix A. Examples . . . . .	12
Appendix B. Note . . . . .	12
Authors' Addresses . . . . .	12

## 1. Introduction

The Static Context Header Compression (SCHC) specification [I-D.ietf-lpwan-ipv6-static-context-hc] describes generic header compression and fragmentation techniques that can be used on all LPWAN (Low Power Wide Area Networks) technologies defined in

[I-D.ietf-lpwan-overview]. Even though those technologies share a great number of common features like star-oriented topologies, network architecture, devices with mostly quite predictable communications, etc; they do have some slight differences in respect of payload sizes, reactivity, etc.

SCHC gives a generic framework that enables those devices to communicate with other Internet networks. However, for efficient performance, some parameters and modes of operation need to be set appropriately for each of the LPWAN technologies.

This document describes the efficient parameters and modes of operation when SCHC is used over LoRaWAN networks.

## 2. Terminology

This section defines the terminology and acronyms used in this document. For all other definitions, please look up the SCHC specification [I-D.ietf-lpwan-ipv6-static-context-hc].

- o DevEUI: an IEEE EUI-64 identifier used to identify the device during the procedure while joining the network (Join Procedure)
- o DevAddr: a 32-bit non-unique identifier assigned to a device statically or dynamically after a Join Procedure (depending on the activation mode)
- o TBD: all significant LoRaWAN-related terms.

## 3. Static Context Header Compression Overview

This section contains a short overview of Static Context Header Compression (SCHC). For a detailed description, refer to the full specification [I-D.ietf-lpwan-ipv6-static-context-hc].

Static Context Header Compression (SCHC) avoids context synchronization, which is the most bandwidth-consuming operation in other header compression mechanisms such as RoHC [RFC5795]. Based on the fact that the nature of data flows is highly predictable in LPWAN networks, some static contexts may be stored on the Device (Dev). The contexts must be stored in both ends, and it can either be learned by a provisioning protocol or by out of band means or it can be pre-provisioned, etc. The way the context is learned on both sides is out of the scope of this document.

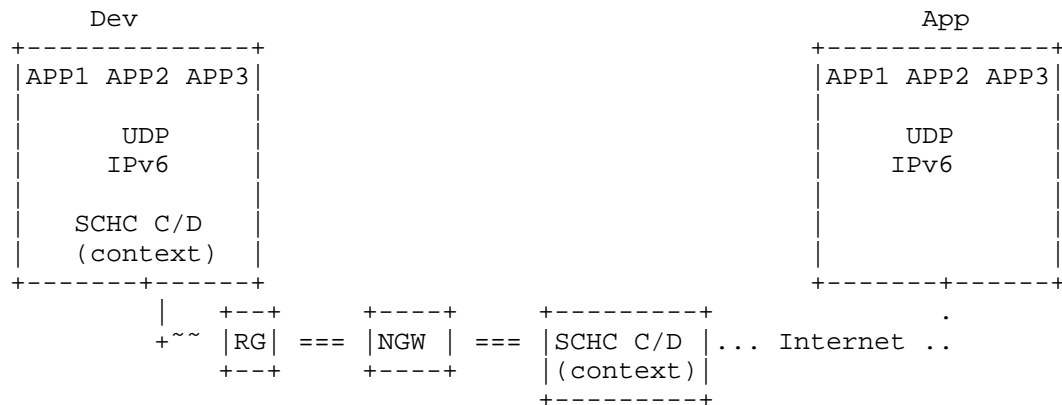


Figure 1: Architecture

Figure 1 represents the architecture for compression/decompression, it is based on [I-D.ietf-lpwan-overview] terminology. The Device is sending applications flows using IPv6 or IPv6/UDP protocols. These flows are compressed by an Static Context Header Compression Compressor/Decompressor (SCHC C/D) to reduce headers size. Resulting information is sent on a layer two (L2) frame to a LPWAN Radio Network (RG) which forwards the frame to a Network Gateway (NGW). The NGW sends the data to a SCHC C/D for decompression which shares the same rules with the Dev. The SCHC C/D can be located on the Network Gateway (NGW) or in another place as long as a tunnel is established between the NGW and the SCHC C/D. The SCHC C/D in both sides must share the same set of Rules. After decompression, the packet can be sent on the Internet to one or several LPWAN Application Servers (App).

The SCHC C/D process is bidirectional, so the same principles can be applied in the other direction.

In a LoRaWAN network, the RG is called a Gateway, the NGW is Network Server, and the SCHC C/D can be embedded in different places, for example in the Network Server and/or the Application Server.

Next steps for this section: detailed overview of the LoRaWAN architecture and its mapping to the SCHC architecture.

#### 4. LoRaWAN Architecture

An overview of LoRaWAN [lora-alliance-spec] protocol and architecture is described in [I-D.ietf-lpwan-overview]. Mapping between the LPWAN architecture entities as described in



[I-D.ietf-lpwan-ipv6-static-context-hc] and the ones in [lora-alliance-spec] is as follows:

- 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. This entity maps to the LoRaWAN End-device.
- o The Radio Gateway (RGW), which is the end point of the constrained link. This entity maps to the LoRaWAN Gateway.
- o The Network Gateway (NGW) is the interconnection node between the Radio Gateway and the Internet. This entity maps to the LoRaWAN Network Server.
- o LPWAN-AAA Server, which controls the user authentication and the applications. This entity maps to the LoRaWAN Join Server.
- o Application Server (App). The same terminology is used in LoRaWAN.

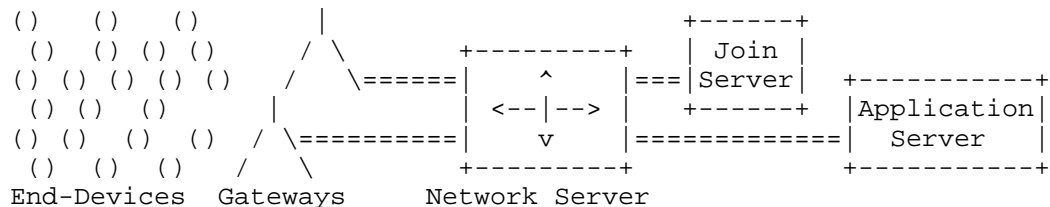


Figure 1: LPWAN/LoRaWAN Architecture

SCHC C/D (Compressor/Decompressor) and SCHC Fragmentation are performed on the LoRaWAN End-device and the Application Server. While the point-to-point link between the End-device and the Application Server constitutes single IP hop, the ultimate end-point of the IP communication may be an Internet node beyond the Application Server. In other words, the LoRaWAN Application Server acts as the first hop IP router for the End-device. Note that the Application Server and Network Server may be co-located, which effectively turns the Network/Application Server into the first hop IP router.

#### 4.1. Device classes (A, B, C) and interactions

TBD

#### 4.2. Device addressing

TBD

#### 4.3. General Message Types

TBD

#### 4.4. LoRaWAN MAC Frames

TBD

### 5. SCHC over LoRaWAN

#### 5.1. Rule ID management

Rule ID can be stored and transported in the FPort field of the LoRaWAN MAC frame or as the first bytes of the payload.

TBD

#### 5.2. IID computation

TBD

#### 5.3. Fragmentation

TBD

##### 5.3.1. Reliability options

###### 5.3.1.1. Uplinks: From device to gateway

In that case the device is the fragmentation transmitter, and the SCHC gateway the fragmentation receiver.

- o SCHC fragmentation reliability mode : "ACK\_ALWAYS"
- o Window size: 8, the FCN field is encoded on 3 bits
- o DTag : 1bit. this field is used to clearly separate two consecutive fragmentation sessions. A LoRaWAN device cannot interleave several fragmented SCHC datagrams.
- o MIC calculation algorithm: CRC32 using 0xEDB88320 (i.e. the reverse representation of the polynomial used e.g. in the Ethernet standard [RFC3385])

- o Retransmission Timer and inactivity Timer: LoRaWAN devices do not implement a "retransmission timer". At the end of a window the ACK corresponding to this window is transmitted by the network gateway in the RX1 or RX2 receive slot of the device. If this ACK is not received the device sends an all-0 (or an all-1) fragment with no payload to request an ACK retransmission. The periodicity between retransmission of the all-0/all-1 fragments is device/application specific and may be different for each device (not specified). The gateway implements an "inactivity timer". The default recommended duration of this timer is 12h. This value is mainly driven by application requirements and may be changed.

RuleID	DTag	W	FCN	Payload
-----	-----	-----	-----	-----
3 bits	1 bit	1 bit	3 bits	

Figure 2: All fragment except the last one. Header size is 8 bits.

RuleID	DTag	W	FCN	MIC	Payload
-----	-----	-----	-----	-----	-----
3 bits	1 bit	1 bit	3 bits	32 bits	

Figure 3: All-1 fragment detailed format for the last fragment.  
Header size is 8 bits.

The format of an all-0 or all-1 acknowledge is:

RuleID	DTag	W	Encoded bitmap	Padding (0s)
-----	-----	-----	-----	-----
3 bits	1 bit	1 bit	up to 8 bits	0 to 3 bits

Figure 4: ACK format for All-0 windows. Header size is 1 or 2 bytes.

RuleID	DTag	W	C	Encoded bitmap (if C = 0)	Padding (0s)
-----	-----	-----	-----	-----	-----
3 bits	1 bit	1 bit	1 bit	up to 8 bits	0 to 2 bits

Figure 5: ACK format for All-1 windows. Header size is 1 or 2 bytes.

## 5.3.1.2. Downlinks: From gateway to device

In that case the device is the fragmentation receiver, and the SCHC gateway the fragmentation transmitter. The following fields are common to all devices.

- o SCHC fragmentation reliability mode : ACK\_ALWAYS
- o Window size : 1 , The FCN field is encoded on 1 bits
- o DTag : 1bit. This field is used to clearly separate two consecutive fragmentation sessions. A LoRaWAN device cannot interleave several fragmented SCHC datagrams.
- o MIC calculation algorithm: CRC32 using 0xEDB88320 (i.e. the reverse representation of the polynomial used e.g. in the Ethernet standard [RFC3385])
- o MAX\_ACK\_REQUESTS : 8

RuleID	DTag	W	FCN	Payload	Padding	
+ -----	+ -----	+ -----	+ -----	+ -----	+ -----	+
3 bits	1 bit	1 bit	1 bits	X bytes	2 bits	

Figure 6: All fragments but the last one. Header size is 6 bits.

RuleID	DTag	W	FCN	MIC	Payload	Padding	
+ -----	+ -----	+ -----	+ -----	+ -----	+ -----	+ -----	+
3 bits	1 bit	1 bit	1 bits	32 bits	X bytes	2 bits	

Figure 7: All-1 Fragment Detailed Format for the Last Fragment.  
Header size is 6 bits.

The format of an all-0 or all-1 acknowledge is:

RuleID	DTag	W	Encoded bitmap	Padding (0s)	
+ -----	+ -----	+ -----	+ -----	+ -----	+
3 bits	1 bit	1 bit	1 bit	2 bits	

Figure 8: ACK format for All-0 windows. Header size is 8 bits.

RuleID	DTag	W	C = 1	Padding (0s)	
+ -----	+ -----	+ -----	+ -----	+ -----	+
3 bits	1 bit	1 bit	1 bit	2 bits	

Figure 9: ACK format for All-1 windows, MIC is correct. Header size is 8 bits.

RuleID	DTag	W	b'111	0xFF (all 1's)	
+ -----	+ -----	+ -----	+ -----	+ -----	+
3 bits	1 bit	1 bit	3 bits	8 bits	

Figure 10: Receiver ABORT packet (following an all-1 packet with incorrect MIC). Header size is 16 bits.

Class A and classB&C device do not manage retransmissions and timers in the same way.

#### 5.3.1.2.1. Class A devices

Class A devices can only receive in an RX slot following the transmission of an uplink. Therefore there cannot be a concept of "retransmission timer" for a gateway talking to classA devices for downlink fragmentation.

The device replies with an ACK fragment to every single fragment received from the gateway (because the window size is 1). Following the reception of a FCN=0 fragment (fragment that is not the last fragment of the packet or ACK-request), the device MUST transmit the ACK fragment until it receives the fragment of the next window. The device shall transmit up to MAX\_ACK\_REQUESTS ACK fragments before aborting. The device should transmit those ACK as soon as possible while taking into consideration eventual local radio regulation on duty-cycle, to progress the fragmentation session as quickly as possible. The ACK bitmap is 1 bit long and is always 1.

Following the reception of a FCN=1 fragment (the last fragment of a datagram) and if the MIC is correct, the device shall transmit the ACK with the "MIC is correct" indicator bit set. This message might be lost therefore the gateway may request a retransmission of this ACK in the next downlink. The device SHALL keep this ACK message in memory until it receives a downlink from the gateway different from an ACK-request indicating that the gateway has received the ACK message.

Following the reception of a FCN=1 fragment (the last fragment of a datagram) and if the MIC is NOT correct, the device shall transmit a receiver-ABORT fragment. The device SHALL keep this ABORT message in memory until it receives a downlink from the gateway different from an ACK-request indicating that the gateway has received the ABORT message. The fragmentation receiver (device) does not implement retransmission timer and inactivity timer.

The fragmentation sender (the gateway) implements an inactivity timer with default duration 12 hours. Once a fragmentation session is started, if the gateway has not received any ACK or receiver-ABORT message 12 hours later the last message from the device was received, the gateway may flush the fragmentation context. For devices with very low transmission rates (example 1 packet a day in normal operation) , that duration may be extended, but this is application specific.

#### 5.3.1.3. Class B or C devices

Class B&C devices can receive in scheduled RX slots or in RX slots following the transmission of an uplink. The device replies with an ACK fragment to every single fragment received from the gateway (because the window size is 1). Following the reception of a FCN=0 fragment (fragment that is not the last fragment of the packet or ACK-request), the device MUST always transmit the corresponding ACK fragment even if that fragment has already been received. The ACK bitmap is 1 bit long and is always 1. If the gateway receives this ACK, it proceeds to send the next window fragment. If the retransmission timer elapses and the gateway has not received the ACK of the current window it retransmits the last fragment. The gateway tries retransmitting up to MAX\_ACK\_REQUESTS times before aborting.

Following the reception of a FCN=1 fragment (the last fragment of a datagram) and if the MIC is correct, the device shall transmit the ACK with the "MIC is correct" indicator bit set. If the gateway receives this ACK, the current fragmentation session has succeeded and its context can be cleared.

If the retransmission timer elapses and the gateway has not received the all-1 ACK it retransmits the last fragment with the payload (not an ACK-request without payload). The gateway tries retransmitting up to MAX\_ACK\_REQUESTS times before aborting.

The device SHALL keep the all-1 ACK message in memory until it receives a downlink from the gateway different from the last (FCN=1) fragment indicating that the gateway has received the ACK message. Following the reception of a FCN=1 fragment (the last fragment of a datagram) and if the MIC is NOT correct, the device shall transmit a

receiver-ABORT fragment. The retransmission timer is used by the gateway (the sender), the optimal value is very much application specific but here are some recommended default values. For classB devices, this timer trigger is a function of the periodicity of the classB ping slots. The recommended value is equal to 3 times the classB ping slot periodicity. (modify 128sec) For classC devices which are nearly constantly receiving, the recommended value is 30 seconds. This means that the device shall try to transmit the ACK within 30 seconds of the reception of each fragment. The inactivity timer is implemented by the device to flush the context in-case it receives nothing from the gateway over an extended period of time. The recommended value is 12 hours for both classB&C devices.

#### 5.3.2. Supporting multiple window sizes

TBD

#### 5.3.3. Downlink fragment transmission

TBD

#### 5.3.4. SCHC behavior for devices in class A, B and C

TBD

### 6. Security considerations

TBD

### 7. Acknowledgements

TBD

### 8. References

#### 8.1. Normative 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>>.
- [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>>.

## 8.2. Informative References

- [I-D.ietf-lpwan-ipv6-static-context-hc]  
Minaburo, A., Toutain, L., Gomez, C., and D. Barthel,  
"LPWAN Static Context Header Compression (SCHC) and  
fragmentation for IPv6 and UDP", draft-ietf-lpwan-ipv6-  
static-context-hc-16 (work in progress), June 2018.
- [I-D.ietf-lpwan-overview]  
Farrell, S., "LPWAN Overview", draft-ietf-lpwan-  
overview-10 (work in progress), February 2018.
- [lora-alliance-spec]  
Alliance, L., "LoRaWAN Specification Version V1.0.2",  
<[http://portal.lora-  
alliance.org/DesktopModules/Inventures\\_Document/  
FileDownload.aspx?ContentID=1398](http://portal.lora-alliance.org/DesktopModules/Inventures_Document/FileDownload.aspx?ContentID=1398)>.

## Appendix A. Examples

## Appendix B. Note

## Authors' Addresses

Nicolas Sornin (editor)  
Semtech  
14 Chemin des Clos  
Meylan  
France  
  
Email: [nsornin@semtech.com](mailto:nsornin@semtech.com)



Michael Coracin  
Semtech  
14 Chemin des Clos  
Meylan  
France

Email: mcoracin@semtech.com

Ivaylo Petrov  
Acklio  
2bis rue de la Chataigneraie  
35510 Cesson-Sevigne Cedex  
France

Email: ivaylo@ackl.io

Alper Yegin  
Actility  
.  
Paris, Paris  
France

Email: alper.yegin@actility.com

Julien Catalano  
Kerlink  
1 rue Jacqueline Auriol  
35235 Thorigne-Fouillard  
France

Email: j.catalano@kerlink.fr

Vincent AUDEBERT  
EDF R&D  
7 bd Gaspard Monge  
91120 PALAISEAU  
FRANCE

Email: vincent.audebert@edf.fr

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: April 29, 2018

A. Minaburo  
Acklio  
L. Toutain  
Institut Mines Telecom Atlantique  
October 26, 2017

CoAP Time Scale Option  
draft-toutain-core-time-scale-00

Abstract

SCHC compression mechanism for LPWAN network enables IPv6 on devices connected to a constrained network (LPWAN). They can communicate with a CoAP server located anywhere in the Internet. LPWAN network characteristics limits the number of exchanges and may impose a long RTT. The CoAP server must be aware of these properties to manage correctly requests. The Time Scale option allows a device to inform a CoAP server of the duration the message ID value should be kept in memory to manage correctly message duplication.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 29, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## 1. CoAP Message ID

Constraint Application Protocol (CoAP) [RFC7252] implements a simple reliable transport mechanism based on ARQ. Each CoAP message contains a 16 bit Message ID (noted afterward MID). A client selects a MID in a CON message and expects an ACK message containing the same MID value. A timer makes the client resend the request if no ACK is received during a pre-defined period.

To avoid a second process of duplicated requests by the server, a list of messages ID already acknowledged must be maintained for a period of time. If the message ID is already in the list, the message is just acknowledged and not processed by upper layer. Therefore, the client cannot use this MID value in another request during the same period of time.

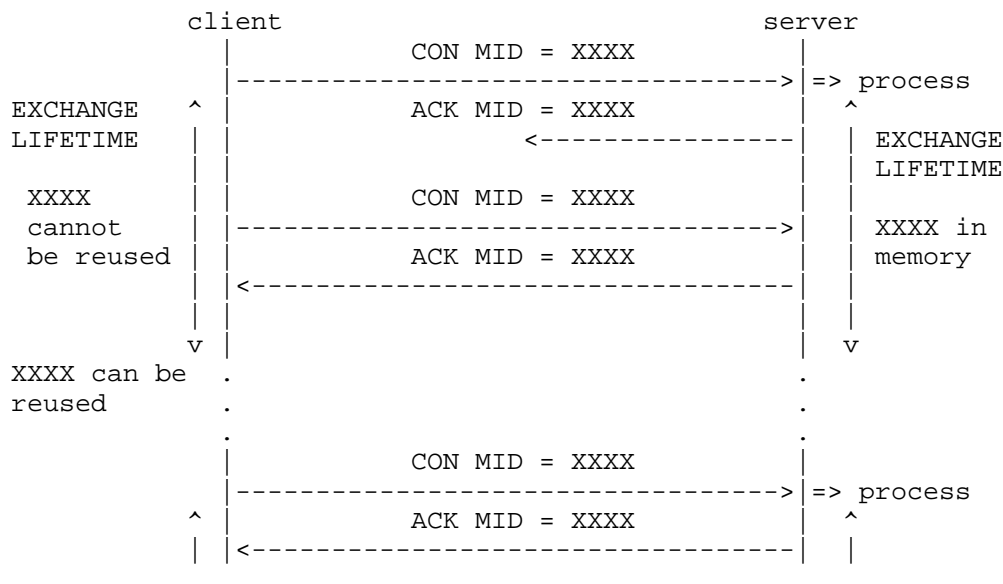


Figure 1: Delayed transmission.

[RFC7252] calls the period a MID is assigned to a request the `EXCHANGE_LIFETIME`. The value is based on the worst case scenario

taking into account the propagation time, the number of retransmissions and the processing time. The default value for EXCHANGE\_LIFETIME is set to 247 seconds for MAX\_RTT of 202 seconds.

## 2. LPWAN networks

Low Power Wide Area Network (LPWAN) family regroups networks dedicated to the Internet of Things. They provide a large coverage with a limited energy consumption. They mostly use the license-free ISM band. The [I-D.ietf-lpwan-overview] gives an overview of the technology and the star oriented topology architecture. A Network Gateway (NGW) is at the interconnection between the LPWAN and the Internet network.

To ensure fairness among nodes, regulation imposes a duty cycle. In practice, with a 1% duty cycle, a node sending a message of  $s$  seconds must wait  $99 \times s$  seconds before sending another message. For instance, in some technologies sending a 50 bytes message takes 2 seconds, forcing a silence of 198 seconds.

The device sleeps most of the time to preserve energy. If a device can use the uplink channel at any time, downlink channel is generally available during a short receiving window following the message emission. Therefore a message sent to a device out of this receiving window will be lost. Network Gateways are aware of this restriction and buffers downlink messages until an uplink message is received which opens the receiving window.

Figure 2 illustrates this. A CoAP client sends a request every hour. Even if the server replies immediately, the answer may be buffered by the Network GW until an new uplink message is sent. In that case, the client will only receive the answer after one hour when the next request is sent. The RTT is influenced by the message periodicity and the EXCHANGE\_LIFETIME value can be computed locally by client to dimension its timers.

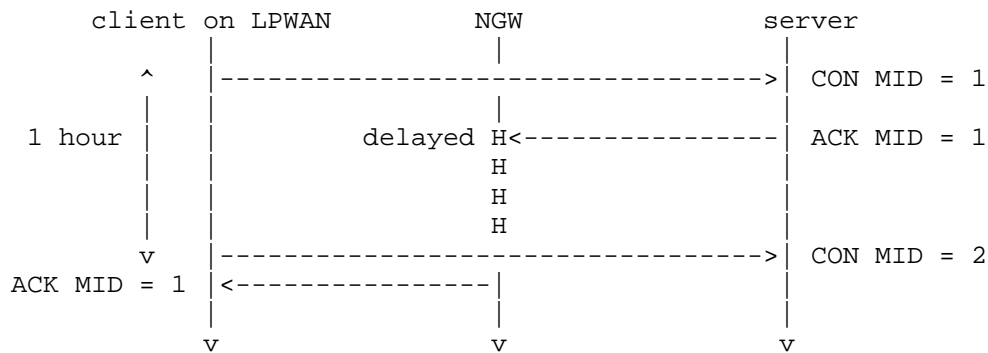


Figure 2: Delayed transmission.

The server should remain as generic as possible and `EXCHANGE_LIFETIME` parameter has to be adapted to the client behavior. If the period is too large, the server will have to memorize a longer list of MID for fast responding client. On the other hand, if the `EXCHANGE_LIFETIME` is too short, this leads to misbehaviors as shown in Figure 3, a retransmission will be viewed as a new request.

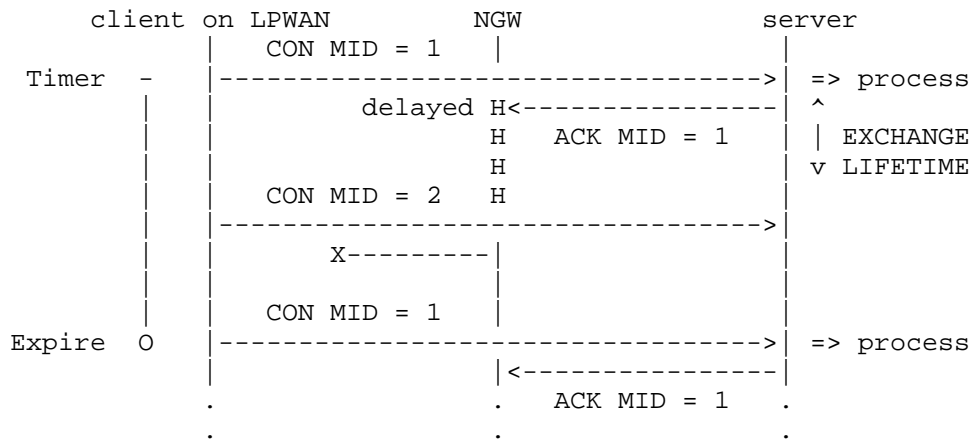


Figure 3: Retransmission.

The Time Scale option, added into all the CoAP requests, informs the server of the duration a message ID should be memorized into the server and therefore the duration during which a client should not reuse the same message ID for a new request. This way, the server can adapt its behavior to different environments.

It is important to notice that this option will not contribute to an DoS attack. This option does not increase the number of message ID memorized by the server. In fact, the Time Scale option can be viewed as a contract between the client and the server, which means that the client will send a reasonable number of request during that period. The number of memorized message ID is independent of the duration of the exchange but linked to the number a simultaneous request a client can send. If a client is sending a number of request larger than expected, they can be easily discarded by the server.

### 3. Timescale Option

Timescale is a new CoAP option that tells the server how many seconds the MID should be memorized by the server. This option must be included in all the exchanges coming from a high latency device.

Number	C	U	N	R	Name	Format	Length	Default
259	X				Time Scale	uint	1-4	3600

Figure 4: Time Scale Option.

This option is critical, if a server does not recognize it, it must inform the client that EXCHANGE\_LIFETIME cannot be modified. The option is Safe-to-forward so a proxy does not have to understand this option, since only the server is concerned with the MID management. The value (in seconds) contains the new EXCHANGE\_LIFETIME set by the server for this request. If the value is smaller than the default value, this option is discarded and the client receives an error message.

### 4. Normative References

[I-D.ietf-lpwan-overview]

Farrell, S., "LPWAN Overview", draft-ietf-lpwan-overview-07 (work in progress), October 2017.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.

Authors' Addresses

Ana Minaburo  
Acklio  
2bis rue de la Chataigneraie  
35510 Cesson-Sevigne Cedex  
France

Email: [ana@ackl.io](mailto:ana@ackl.io)

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

Email: [Laurent.Toutain@imt-atlantique.fr](mailto:Laurent.Toutain@imt-atlantique.fr)

lpwan Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 3, 2019

JC. Zuniga  
SIGFOX  
C. Gomez  
Universitat Politecnica de Catalunya  
L. Toutain  
IMT-Atlantique  
July 02, 2018

SCHC over Sigfox LPWAN  
draft-zuniga-lpwan-schc-over-sigfox-03

Abstract

The Static Context Header Compression (SCHC) specification describes a header compression scheme and fragmentation functionality for Low Power Wide Area Network (LPWAN) technologies. SCHC offers a great level of flexibility that can be tailored for different LPWAN technologies.

The present document provides the optimal parameters and modes of operation when SCHC is implemented over a Sigfox LPWAN.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of



publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Static Context Header Compression . . . . .	3
4. SCHC over Sigfox . . . . .	4
4.1. SCHC Rules . . . . .	4
4.2. Packet processing . . . . .	4
5. Fragmentation . . . . .	4
5.1. Fragmentation headers . . . . .	5
5.2. Uplink fragment transmissions . . . . .	5
5.2.1. Uplink No-ACK mode . . . . .	5
5.2.2. Uplink ACK-Always mode . . . . .	6
5.2.3. Uplink ACK-on-Error mode . . . . .	6
5.3. Downlink fragment transmissions . . . . .	6
6. Padding . . . . .	7
7. Security considerations . . . . .	8
8. Acknowledgements . . . . .	8
9. Informative References . . . . .	8
Authors' Addresses . . . . .	9

## 1. Introduction

The Static Context Header Compression (SCHC) specification [I-D.ietf-lpwan-ipv6-static-context-hc] defines a header compression scheme and fragmentation functionality that can be used on top of all the LPWAN systems defined in [I-D.ietf-lpwan-overview]. These LPWAN systems have similar characteristics such as star-oriented topologies, network architecture, connected devices with built-in applications, etc.

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

This document describes the optimal parameters and modes of operation when SCHC is implemented over a Sigfox LPWAN.

## 2. Terminology

The reader is assumed to be familiar with the terms and mechanisms defined in [I-D.ietf-lpwan-overview] and in [I-D.ietf-lpwan-ipv6-static-context-hc].

## 3. Static Context Header Compression

Static Context Header Compression (SCHC) avoids context synchronization because data flows are highly predictable in LPWAN networks. Contexts must be stored and configured on both ends. This can be done either by using a provisioning protocol, by out of band means, or by pre-provisioning them e.g. at manufacturing time. The way the contexts are configured and stored on both ends is out of the scope of this document.

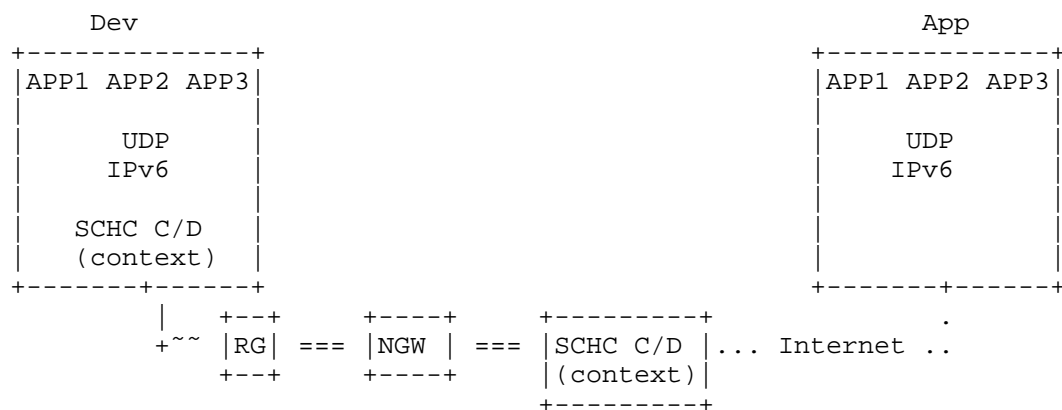


Figure 1: Architecture

Figure 1 represents the architecture for compression/decompression and fragmentation, which is based on [I-D.ietf-lpwan-overview] terminology.

The Device is sending applications flows that are compressed (and/or fragmented) by a Static Context Header Compression Compressor/Decompressor (SCHC C/D) to reduce headers size and/or fragment the packet. The resulting information is sent over a layer two (L2) frame to a LPWAN Radio Gateway (RG) which forwards the frame to a Network Gateway (NGW).

#### 4. SCHC over Sigfox

In the case of the global Sigfox network, RGs (or base stations) are distributed over the multiple countries where the Sigfox LPWAN service is provided. On the other hand, the NGW (or Cloud-based Core network) is a single entity that connects to all Sigfox base stations in the world.

Uplink transmissions occur in repetitions over different times and frequencies. Besides these time and frequency diversities, the Sigfox network also provides space diversity, as potentially an uplink message will be received by several base stations. Since all messages are self-contained and base stations forward them all back to the same Core network (NGW), multiple input copies can be combined at the NGW and hence provide for extra reliability based on the triple diversity.

The NGW communicates with the Network SCHC C/D for compression/decompression (and/or fragmentation/reassembly). The Network SCHC C/D shares the same set of rules as the Dev SCHC C/D. The Network SCHC C/D can be collocated with the NGW or in another place, as long as a tunnel is established between the NGW and the SCHC C/D. After decompression (and/or reassembly), the packet can be forwarded over the Internet to one (or several) LPWAN Application Server(s) (App).

The SCHC C/D process is bidirectional, so the same principles can be applied on both uplink and downlink.

##### 4.1. SCHC Rules

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

##### 4.2. Packet processing

TBD

#### 5. Fragmentation

The SCHC specification [I-D.ietf-lpwan-ipv6-static-context-hc] defines a generic fragmentation functionality that allows sending data packets larger than the maximum size of a Sigfox data frame. The functionality also defines a mechanism to send reliably multiple frames, by allowing to resend selectively any lost frames.

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

### 5.1. Fragmentation headers

A list of fragmentation header fields, their sizes as well as recommended modes for SCHC fragmentation over Sigfox are provided in this section.

### 5.2. Uplink fragment transmissions

Uplink transmissions are completely asynchronous and can take place in any random frequency of the allowed uplink bandwidth allocation. Hence, devices can go to deep sleep mode, and then wake up and transmit whenever there is a need to send any information to the network. In that way, there is no need to perform any network attachment, synchronization, or other procedure before transmitting a data packet. All data packets are self contained with all the required information for the network to process them accordingly.

Since uplink transmissions occur asynchronously, an SCHC fragment can be transmitted at any given time by the Dev.

#### 5.2.1. Uplink No-ACK mode

No-ACK is RECOMMENDED to be used for transmitting short, non-critical packets that require fragmentation.

Fragmentation Header size: 8 bits

The recommended Rule ID size is: 3 bits

The recommended DTag size (T) is: 0 bits, as the number of available Rule IDs are sufficient to interleave fragmented packets.

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

As per [REF SCHC], in the No-ACK mode the W (window) 1-bit field is not present.

When fragmentation is used to transport IP frames, the Message Integrity Check (MIC) size, M: TBD bits

The algorithm for computing the MIC field MUST be TBD.

### 5.2.2. Uplink ACK-Always mode

TBD

### 5.2.3. Uplink ACK-on-Error mode

ACK-on-Error is RECOMMENDED for larger packets, since it leads to a reduced number of ACKs to be sent in the lower capacity downlink channel.

The recommended Fragmentation Header size is: 8 bits

The recommended Rule ID size is: 3 bits.

The recommended DTag size (T) is: 0 bits, as the number of available Rule IDs are sufficient to interleave fragmented packets.

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

As per [REF SCHC], in the ACK-on-Error mode the Window (W) 1-bit field must be present.

For the ACK-on-Error fragmentation mode(s), a single window size is RECOMMENDED.

The value of MAX\_ACK\_REQUESTS SHOULD be 2, and the value of MAX\_WIND\_FCN SHOULD be 14 (which allows a maximum window size with 15 fragments).

When fragmentation is used to transport IP frames, the Message Integrity Check (MIC) size, M: TBD bits

The algorithm for computing the MIC field MUST be TBD.

### 5.3. Downlink fragment transmissions

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

When there are multiple fragments to be transmitted in the downlink, an uplink message is required to trigger the downlink communication. In order to avoid potentially high delay for fragmented datagram transmission in the downlink, the fragment receiver MAY perform an

uplink transmission as soon as possible after reception of a fragment that is not the last one. Such uplink transmission MAY be triggered by sending a SCHC message, such as an ACK. In this sense, ACK-Always is the preferred fragmentation mode for downlink communications.

For downlink fragment transmission, the ACK-Always mode MUST be supported.

The recommended Fragmentation Header size is: 8 bits

The recommended Rule ID size is: 3 bits.

The recommended DTag size (T) is: 0 bits, as the number of available Rule IDs are sufficient to interleave fragmented packets.

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

As per [REF SCHC], in the ACK-on-Error mode the Window (W) 1-bit field must be present.

For the ACK-Always fragmentation mode(s), a single window size is RECOMMENDED.

The value of MAX\_ACK\_REQUESTS SHOULD be 2, and the value of MAX\_WIND\_FCN SHOULD be 14 (which allows a maximum window size with 15 fragments).

When fragmentation is used to transport IP frames, the Message Integrity Check (MIC) size, M: TBD bits

The algorithm for computing the MIC field MUST be TBD.

Sigfox downlink frames have a fixed length of 8 bytes, which means that default SCHC algorithm for padding cannot be used. Therefore, the 3 last bits of the fragmentation header are used to indicate in bytes the size of the padding. A size of 000 means that the full remaining frame is used to carry payload, a value of 001 indicates that the last byte contains padding, and so on.

## 6. Padding

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

Uplink frames can contain a payload from 0 to 96 bits (i.e. 12 bytes). The radio protocol allows sending zero bits or one single bit of information for binary applications (e.g. status). However, for 2 or more bits of payload it is required to add padding to the

next integer number of bytes. The reason for this flexibility is to optimize transmission time and hence save battery consumption at the device.

Downlink frames on the other hand have a fixed length. The payload length must be 64 bits (i.e. 8 bytes). Hence, if less information bits are to be transmitted padding would be necessary and it should be performed as described in the previous section.

## 7. Security considerations

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

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

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

## 8. Acknowledgements

Carles Gomez has been funded in part by the ERDF and the Spanish Government through project TEC2016-79988-P.

## 9. Informative References

[I-D.ietf-lpwan-ipv6-static-context-hc]

Minaburo, A., Toutain, L., and C. Gomez, "LPWAN Static Context Header Compression (SCHC) and fragmentation for IPv6 and UDP", draft-ietf-lpwan-ipv6-static-context-hc-07 (work in progress), October 2017.

[I-D.ietf-lpwan-overview]

Farrell, S., "LPWAN Overview", draft-ietf-lpwan-overview-07 (work in progress), October 2017.

Authors' Addresses

Juan Carlos Zuniga  
SIGFOX  
425 rue Jean Rostand  
Labège 31670  
France

Email: [JuanCarlos.Zuniga@sigfox.com](mailto:JuanCarlos.Zuniga@sigfox.com)  
URI: <http://www.sigfox.com/>

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

Email: [carlesgo@entel.upc.edu](mailto:carlesgo@entel.upc.edu)

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

Email: [Laurent.Toutain@imt-atlantique.fr](mailto:Laurent.Toutain@imt-atlantique.fr)