

lpwan
Internet-Draft
Intended status: Informational
Expires: May 1, 2017

S. Farrell
Trinity College Dublin
A. Yegin
Actility
October 28, 2016

LoRaWAN Overview
draft-farrell-lpwan-lora-overview-01

Abstract

Low Power Wide Area Networks (LPWAN) are wireless technologies covering different Internet of Things (IoT) applications. The common characteristics for LPWANs are large coverage, low bandwidth, small packet and application layer data sizes and long battery life operation. One of these technologies is LoRaWAN developed by the LoRa Alliance. LoRaWAN targets key requirements of the Internet of things such as secure bi-directional communication, mobility and localization services. This memo is an informational overview of LoRaWAN and gives the principal characteristics of this technology in order to help with the IETF work for providing IPv6 connectivity over LoRaWAN along with other LPWANs.

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 <http://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 May 1, 2017.

Copyright Notice

Copyright (c) 2016 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

(<http://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. Radio Spectrum	4
4. MAC Layer	6
5. Names and Addressing	8
6. Security Considerations	10
6.1. Payload Encryption and Data Integrity	10
6.2. Key Derivation	10
7. IANA Considerations	11
8. Acknowledgements	11
9. Contributors	11
10. Informative References	12
Authors' Addresses	12

1. Introduction

LoRaWAN is a wireless technology for long-range low-power low-data-rate applications developed by the LoRa Alliance, a membership consortium. [<https://www.lora-alliance.org/>](https://www.lora-alliance.org/) LoRaWAN networks are typically organized in a star-of-stars topology in which gateways relay messages between end-devices and a central "network server" in the backend. Gateways are connected to the network server via IP links while end-devices use single-hop LoRaWAN communication that can be received at one or more gateways. All communication is generally bi-directional, although uplink communication from end-devices to the network server are favoured in terms of overall bandwidth availability.

In LoRaWAN networks, end-device transmissions may be received at multiple gateways, so during nominal operation a network server may see multiple instances of the same uplink message from an end-device.

To maximize both battery life of end-devices and overall network capacity, the LoRaWAN network infrastructure manages the data rate and RF output power for each end-device individually by means of an adaptive data rate (ADR) scheme. End-devices may transmit on any channel allowed by local regulation at any time, using any of the currently available data rates.

This memo provides an overview of the LoRaWAN technology for the Internet community, but the definitive specification [LoRaSpec] is that produced by the LoRa Alliance. This draft is based on version 1.0.2 of the LoRa specification. (Note that version 1.0.2 is expected to be published in a few weeks. We will update this draft when that has happened. For now, version 1.0 is available at [LoRaSpec1.0])

2. Terminology

This section introduces some LoRaWAN terms. Figure 1 shows the entities involved in a LoRaWAN network.

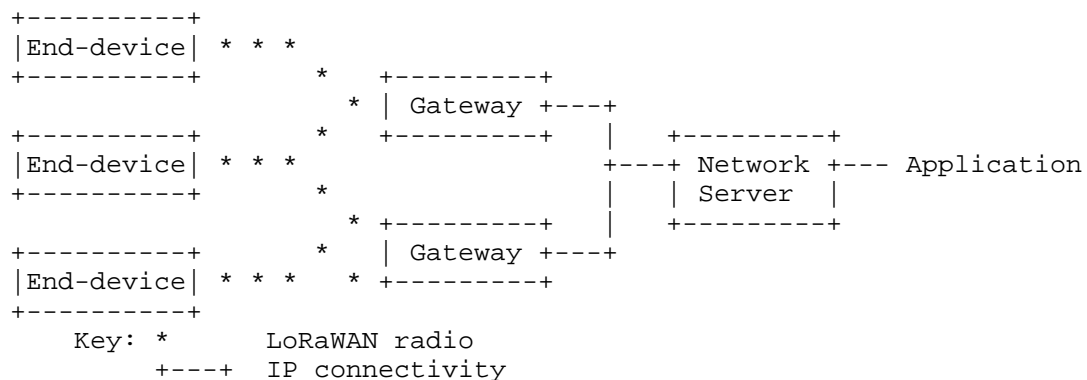


Figure 1: LoRaWAN architecture

- o End-device: a LoRa client device, sometimes called a mote. Communicates with gateways.
- o Gateway: a radio on the infrastructure-side, sometimes called a concentrator or base-station. Communicates with end-devices and, via IP, with a network server.
- o Network Server: The Network Server (NS) terminates the LoRaWAN MAC layer for the end-devices connected to the network. It is the center of the star topology.
- o Uplink message: refers to communications from end-device to network server or application via one or more gateways.
- o Downlink message: refers to communications from network server or application via one gateway to a single end-device or a group of end-devices (considering multicasting).

- o Application: refers to application layer code both on the end-device and running "behind" the network server. For LoRaWAN, there will generally only be one application running on most end-devices. Interfaces between the network server and application are not further described here.
- o Classes A, B and C define different device capabilities and modes of operation for end-devices. End-devices can transmit uplink messages at any time in any mode of operation (so long as e.g., ISM band restrictions are honoured). An end-device in Class A can only receive downlink messages at predetermined timeslots after each uplink message transmission. Class B allows the end-device to receive downlink messages at periodically scheduled timeslots. Class C allows receipt of downlink messages at anytime. Class selection is based on the end-devices' application use case and its power supply. (While Classes B and C are not further described here, readers may have seen those terms elsewhere so we include them for clarity.)

3. Radio Spectrum

LoRaWAN radios make use of ISM bands, for example, 433MHz and 868MHz within the European Union and 915MHz in the Americas.

The end-device changes channel in a pseudo-random fashion for every transmission to help make the system more robust to interference and/or to conform to local regulations.

As with other LPWAN radio technologies, LoRaWAN end-devices respect the frequency, power and maximum transmit duty cycle requirements for the sub-band imposed by local regulators. In most cases, this means an end-device is only transmitting for 1% of the time, as specified by ISM band regulations. And in some cases the LoRaWAN specification calls for end-devices to transmit less often than is called for by the ISM band regulations in order to avoid congestion.

Figure 2 below shows that after a transmission slot a Class A device turns on its receiver for two short receive windows that are offset from the end of the transmission window. The frequencies and data rate chosen for the first of these receive windows match those used for the transmit window. The frequency and data-rate for the second receive window are configurable. If a downlink message preamble is detected during a receive window, then the end-device keeps the radio on in order to receive the frame.

End-devices can only transmit a subsequent uplink frame after the end of the associated receive windows. When a device joins a LoRaWAN

network (see Section 4 for details), there are similar timeouts on parts of that process.

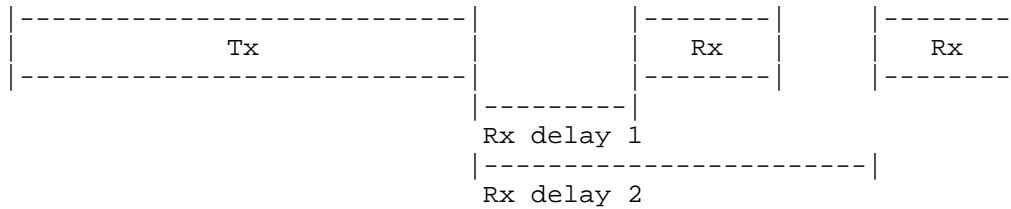


Figure 2: LoRaWAN Class A transmission and reception window

Given the different regional requirements the detailed specification for the LoRaWAN physical layer (taking up more than 30 pages of the specification) is not reproduced here. Instead and mainly to illustrate the kinds of issue encountered, in Table 1 we present some of the default settings for one ISM band (without fully explaining those here) and in Table 2 we describe maxima and minima for some parameters of interest to those defining ways to use IETF protocols over the LoRaWAN MAC layer.

Parameters	Default Value
Rx delay 1	1 s
Rx delay 2	2 s (must be RECEIVE_DELAY1 + 1s)
join delay 1	5 s
join delay 2	6 s
868MHz Default channels	3 (868.1,868.2,868.3), data rate: 0.3-5 kbps

Table 1: Default settings for EU868MHz band

Parameter/Notes	Min	Max
Duty Cycle: some but not all ISM bands impose a limit in terms of how often an end-device can transmit. In some cases LoRaWAN is more stringent in an attempt to avoid congestion.	1%	no-limit
EU 868MHz band data rate/frame-size	250 bits/s : 59 octets	50000 bits/s : 250 octets
US 915MHz band data rate/frame-size	980 bits/s : 19 octets	21900 bits/s : 250 octets

Table 2: Minima and Maxima for various LoRaWAN Parameters

Note that in the case of the smallest frame size (19 octets), 8 octets are required for LoRa MAC layer headers leaving only 11 octets for payload (including MAC layer options). However, those settings do not apply for the join procedure - end-devices are required to use a channel that can send the 23 byte Join-request message for the join procedure.

4. MAC Layer

Uplink and downlink higher layer data is carried in a MACPayload. There is a concept of "ports" (an optional 8 bit value) to handle different applications on an end-device. Port zero is reserved for LoRaWAN specific messaging, such as the join procedure.

The header also distinguishes the uplink/downlink directions and whether or not an acknowledgement ("confirmation") is required from the peer.

All payloads are encrypted and ciphertexts are protected with a cryptographic Message Integrity Check (MIC) - see Section 6 for details.

In addition to carrying higher layer PDUs there are Join-Request and Join-Response (aka Join-Accept) messages for handling network access. And so-called "MAC commands" (see below) up to 15 bytes long can be piggybacked in an options field ("FOpts").

LoRaWAN end-devices can choose various different data rates from a menu of available rates (dependent on the frequencies in use). It is however, recommended that end-devices set the Adaptive Data Rate ("ADR") bit in the MAC layer which is a signal that the network should control the data rate (via MAC commands to the end-device). The network can also assert the ADR bit and control data rates at its discretion. The goal is to ensure minimal on-time for radios whilst increasing throughput and reliability when possible. Other things being equal, the effect should be that end-devices closer to a gateway can successfully use higher data rates, whereas end-devices further from all gateways still receive connectivity though at a lower data rate.

Data rate changes can be validated via a scheme of acks from the network with a fall-back to lower rates in the event that downlink acks go missing.

There are 16 (or 32) bit frame counters maintained in each direction that are incremented on each transmission (but not re-transmissions) that are not re-used for a given key. When the device supports a 32 bit counter, then only the least significant 16 bits are sent in the MAC header, but all 32 bits are used in cryptographic operations. (If an end-device only supports a 16 bit counter internally, then the topmost 16 bits are set to zero.)

There are a number of MAC commands for: Link and device status checking, ADR and duty-cycle negotiation, managing the RX windows and radio channel settings. For example, the link check response message allows the network server (in response to a request from an end-device) to inform an end-device about the signal attenuation seen most recently at a gateway, and to also tell the end-device how many gateways received the corresponding link request MAC command.

Some MAC commands are initiated by the network server. For example, one command allows the network server to ask an end-device to reduce its duty-cycle to only use a proportion of the maximum allowed in a region. Another allows the network server to query the end-device's power status with the response from the end-device specifying whether it has an external power source or is battery powered (in which case a relative battery level is also sent to the network server).

The network server can also inform an end-device about channel assignments (mid-point frequencies and data rates). Of course, these must also remain within the bands assigned by local regulation.

5. Names and Addressing

A LoRaWAN network has a short network identifier ("NwkID") which is a seven bit value. A private network (common for LoRaWAN) can use the value zero. If a network wishes to support "foreign" end-devices then the NwkID needs to be registered with the LoRA Alliance, in which case the NwkID is the seven least significant bits of a registered 24-bit NetID. (Note however, that the methods for "roaming" are currently being enhanced within the LoRA Alliance, so the situation here is somewhat fluid.)

In order to operate nominally on a LoRaWAN network, a device needs a 32-bit device address, which is the catentation of the NwkID and a 25-bit device-specific network address that is assigned when the device "joins" the network (see below for the join procedure) or that is pre-provisioned into the device.

End-devices are assumed to work with one or a quite limited number of applications, which matches most LoRaWAN use-cases. The applications are identified by a 64-bit AppEUI, which is assumed to be a registered IEEE EUI64 value.

In addition, a device needs to have two symmetric session keys, one for protecting network artefacts (port=0), the NwkSKey, and another for protecting application layer traffic, the AppSKey. Both keys are used for 128 bit AES cryptpgraphic operations. (See Section 6 for details.)

So, one option is for an end-device to have all of the above, plus channel information, somehow (pre-)provisioned, in which case the end-device can simply start transmitting. This is achievable in many cases via out-of-band means given the nature of LoRaWAN networks. Table 3 summarises these values.

Value	Description
DevAddr	DevAddr (32-bits) = NwkId (7-bits) + device-specific network address (25 bits)
AppEUI	IEEE EUI64 naming the application
NwkSKey	128 bit network session key for use with AES
AppSKey	128 bit application session key for use with AES

Table 3: Values required for nominal operation

As an alternative, end-devices can use the LoRaWAN join procedure in order to setup some of these values and dynamically gain access to the network.

To use the join procedure, an end-device must still know the AppEUI. In addition to the AppEUI, end-devices using the join procedure need to also know a different (long-term) symmetric key that is bound to the AppEUI - this is the application key (AppKey), and is distinct from the application session key (AppSKey). The AppKey is required to be specific to the device, that is, each end-device should have a different AppKey value. And finally the end-device also needs a long-term identifier for itself, syntactically also an EUI-64, and known as the device EUI or DevEUI. Table 4 summarises these values.

Value	Description
DevEUI	IEEE EUI64 naming the device
AppEUI	IEEE EUI64 naming the application
AppKey	128 bit long term application key for use with AES

Table 4: Values required for join procedure

The join procedure involves a special exchange where the end-device asserts the AppEUI and DevEUI (integrity protected with the long-term AppKey, but not encrypted) in a Join-request uplink message. This is then routed to the network server which interacts with an entity that knows that AppKey to verify the Join-request. All going well, a Join-accept downlink message is returned from the network server to the end-device that specifies the 24-bit NetID, 32-bit DevAddr and channel information and from which the AppSKey and NwkSKey can be derived based on knowledge of the AppKey. This provides the end-device with all the values listed in Table 3.

There is some special handling related to which channels to use and for multiple transmissions for the join-request which is intended to ensure a successful join in as many cases as possible. Join-request and Join-accept messages also include some random values (nonces) to both provide some replay protection and to help ensure the session keys are unique per run of the join procedure. If a Join-request fails validation, then no Join-accept message (indeed no message at all) is returned to the end-device. For example, if an end-device is factory-reset then it should end up in a state in which it can re-do the join procedure.

6. Security Considerations

In this section we describe the use of cryptography in LoRaWAN. This section is not intended as a full specification but to be sufficient so that future IETF specifications can encompass the required security considerations. The emphasis is on describing the externally visible characteristics of LoRaWAN.

6.1. Payload Encryption and Data Integrity

All payloads are encrypted and have data integrity. Frame options (used for MAC commands) when sent as a payload (port zero) are therefore protected. MAC commands piggy-backed as frame options ("FOpts") are however sent in clear. Since MAC commands may be sent as options and not only as payload, any values sent in that manner are visible to a passive attacker but are not malleable for an active attacker due to the use of the MIC.

For LoRaWAN version 1.0.x, the NWkSKey session key is used to provide data integrity between the end-device and the network server. The AppSKey is used to provide data confidentiality between the end-device and network server, or to the application "behind" the network server, depending on the implementation of the network.

All MAC layer messages have an outer 32-bit Message Integrity Code (MIC) calculated using AES-CMAC calculated over the ciphertext payload and other headers and using the NwkSKey.

Payloads are encrypted using AES-128, with a counter-mode derived from IEEE 802.15.4 using the AppSKey.

Gateways are not expected to be provided with the AppSKey or NwkSKey, all of the infrastructure-side cryptography happens in (or "behind") the network server.

6.2. Key Derivation

When session keys are derived from the AppKey as a result of the join procedure the Join-accept message payload is specially handled.

The long-term AppKey is directly used to protect the Join-accept message content, but the function used is not an aes-encrypt operation, but rather an aes-decrypt operation. The justification is that this means that the end-device only needs to implement the aes-encrypt operation. (The counter mode variant used for payload decryption means the end-device doesn't need an aes-decrypt primitive.)

The Join-accept plaintext is always less than 16 bytes long, so electronic code book (ECB) mode is used for protecting Join-accept messages.

The Join-accept contains an AppNonce (a 24 bit value) that is recovered on the end-device along with the other Join-accept content (e.g. DevAddr) using the aes-encrypt operation.

Once the Join-accept payload is available to the end-device the session keys are derived from the AppKey, AppNonce and other values, again using an ECB mode aes-encrypt operation, with the plaintext input being a maximum of 16 octets.

7. IANA Considerations

There are no IANA considerations related to this memo.

8. Acknowledgements

The authors re-used some text from [I-D.vilajosana-lpwan-lora-hc]

Stephen Farrell's work on this memo was supported by the Science Foundation Ireleand funded CONNECT centre
<<https://connectcentre.ie/>>.

9. Contributors

The following members of the LoRa Alliance reviewed this draft and contributed (much more than SF) to the definition of LoRaWAN.

Name, Affiliation, email (optional)

Chun-Yeow Yeoh, VADS LYFE SDN BHD, yeow@tmrnd.com.my

Olivier Hersent, Actility, olivier.hersent@actility.com

Dave Kjendal, Senet Inc, dkjendal@senetco.com

Paul Duffy, Cisco, paduffy@cisco.com

Joachim Ernst, Swisscom Broadcast Ltd, joachim.ernst@swisscom.com

Nicolas Sornin, Semtech, nsornin@semtech.com

Phillippe Christin, Orange, philippe.christin@orange.com

10. Informative References

[I-D.vilajosana-lpwan-lora-hc]

Vilajosana, X., Dohler, M., and A. Yegin, "Transmission of IPv6 Packets over LoRaWAN", draft-vilajosana-lpwan-lora-hc-00 (work in progress), July 2016.

[LoRaSpec]

LoRa Alliance, "LoRaWAN Specification Version V1.0.2", Nov 2016, <URL TBD>.

[LoRaSpec1.0]

LoRa Alliance, "LoRaWAN Specification Version V1.0", Jan 2015, <<https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf>>.

Authors' Addresses

Stephen Farrell
Trinity College Dublin
Dublin 2
Ireland

Phone: +353-1-896-2354
Email: stephen.farrell@cs.tcd.ie

Alper Yegin
Actility
Paris, Paris
FR

Email: alper.yegin@actility.com

lpwan Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2021

A. Minaburo
Acklio
L. Toutain
Institut MINES TELECOM; IMT Atlantique
R. Andreasen
Universidad de Buenos Aires
March 08, 2021

LPWAN Static Context Header Compression (SCHC) for CoAP
draft-ietf-lpwan-coap-static-context-hc-19

Abstract

This draft defines how to compress the Constrained Application Protocol (CoAP) using the Static Context Header Compression (SCHC). SCHC is a header compression mechanism adapted for Constrained Devices. SCHC uses a static description of the header to reduce the header's redundancy and size. While RFC 8724 describes the SCHC compression and fragmentation framework, and its application for IPv6/UDP headers, this document applies SCHC for CoAP headers. The CoAP header structure differs from IPv6 and UDP since CoAP uses a flexible header with a variable number of options, themselves of variable length. The CoAP protocol messages format is asymmetric: the request messages have a header format different from the one in the response messages. This specification gives guidance on applying SCHC to flexible headers and how to leverage the asymmetry for more efficient compression Rules.

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 September 9, 2021.

Copyright Notice

Copyright (c) 2021 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
1.1. Terminology	4
2. SCHC Applicability to CoAP	4
3. CoAP Headers compressed with SCHC	7
3.1. Differences between CoAP and UDP/IP Compression	8
4. Compression of CoAP header fields	9
4.1. CoAP version field	9
4.2. CoAP type field	9
4.3. CoAP code field	9
4.4. CoAP Message ID field	10
4.5. CoAP Token fields	10
5. CoAP options	10
5.1. CoAP Content and Accept options.	11
5.2. CoAP option Max-Age, Uri-Host, and Uri-Port fields	11
5.3. CoAP option Uri-Path and Uri-Query fields	11
5.3.1. Variable number of Path or Query elements	13
5.4. CoAP option Size1, Size2, Proxy-URI and Proxy-Scheme fields	13
5.5. CoAP option ETag, If-Match, If-None-Match, Location-Path, and Location-Query fields	13
6. SCHC compression of CoAP extension RFCs	13
6.1. Block	13
6.2. Observe	13
6.3. No-Response	14
6.4. OSCORE	14
7. Examples of CoAP header compression	15
7.1. Mandatory header with CON message	15
7.2. OSCORE Compression	16
7.3. Example OSCORE Compression	20
8. IANA Considerations	31
9. Security considerations	31

10. Acknowledgements	32
11. Normative References	32
Authors' Addresses	33

1. Introduction

CoAP [RFC7252] is a command/response protocol designed for micro-controllers with a small RAM and ROM and optimized for REST-based (Representative state transfer) services. Although the Constrained Devices leads the CoAP design, a CoAP header's size is still too large for LPWAN (Low Power Wide Area Networks). SCHC header compression over CoAP header is required to increase performance or use CoAP over LPWAN technologies.

The [RFC8724] defines SCHC, a header compression mechanism for the LPWAN network based on a static context. Section 5 of the [RFC8724] explains where compression and decompression occur in the architecture. The SCHC compression scheme assumes as a prerequisite that both end-points know the static context before transmission. The way the context is configured, provisioned, or exchanged is out of this document's scope.

CoAP is an application protocol, so CoAP compression requires installing common Rules between the two SCHC instances. SCHC compression may apply at two different levels: at IP and UDP in the LPWAN network and another at the application level for CoAP. These two compressions may be independent. Both follow the same principle described in [RFC8724]. As different entities manage the CoAP compression at different levels, the SCHC Rules driving the compression/decompression are also different. The [RFC8724] describes how to use SCHC for IP and UDP headers. This document specifies how to apply SCHC compression to CoAP headers.

SCHC compresses and decompresses headers based on common contexts between Devices. SCHC context includes multiple Rules. Each Rule can match the header fields to specific values or ranges of values. If a Rule matches, the matched header fields are replaced by the RuleID and the Compression Residue that contains the residual bits of the compression. Thus, different Rules may correspond to different protocol headers in the packet that a Device expects to send or receive.

A Rule describes the packets' entire header with an ordered list of fields descriptions; see section 7 of [RFC8724]. Thereby each description contains the field ID (FID), its length (FL), and its position (FP), a direction indicator (DI) (upstream, downstream, and bidirectional), and some associated Target Values (TV). The direction indicator is used for compression to give the best TV to

the FID when these values differ in the transmission direction. So a field may be described several times.

A Matching Operator (MO) is associated with each header field description. The Rule is selected if all the MOs fit the TVs for all fields of the incoming header. A Rule cannot be selected if the message contains an unknown field to the SCHC compressor.

In that case, a Compression/Decompression Action (CDA) associated with each field gives the method to compress and decompress each field. Compression mainly results in one of 4 actions:

- o send the field value (value-sent),
- o send nothing (not-sent),
- o send some least significant bits of the field (LSB) or,
- o send an index (mapping-sent).

After applying the compression, there may be some bits to be sent. These values are called Compression Residue.

SCHC is a general mechanism applied to different protocols, the exact Rules to be used depending on the protocol and the Application. Section 10 of the [RFC8724] describes the compression scheme for IPv6 and UDP headers. This document targets the CoAP header compression using SCHC.

1.1. Terminology

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

2. SCHC Applicability to CoAP

SCHC Compression for CoAP header MAY be done in conjunction with the lower layers (IPv6/UDP) or independently. The SCHC adaptation layers, described in Section 5 of [RFC8724], may be used as shown in Figure 1, Figure 2, and Figure 3.

In the first example, Figure 1, a Rule compresses the complete header stack from IPv6 to CoAP. In this case, the Device and the NGW perform SCHC C/D (Static Context Header Compression Compressor/

Decompressor). The Application communicating with the Device does not implement SCHC C/D.

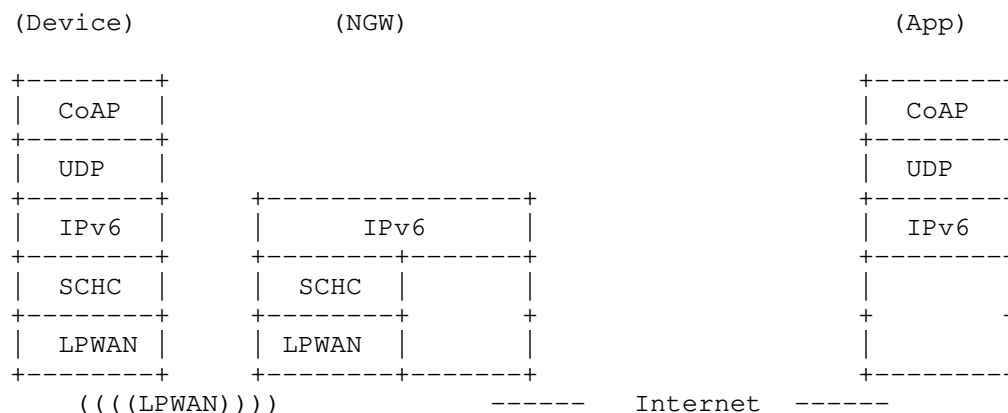


Figure 1: Compression/Decompression at the LPWAN boundary.

Figure 1 shows the use of SCHC header compression above layer 2 in the Device and the NGW. The SCHC layer receives non-encrypted packets and can apply compression Rules to all the headers in the stack. On the other end, the NGW receives the SCHC packet and reconstructs the headers using the Rule and the Compression Residue. After the decompression, the NGW forwards the IPv6 packet toward the destination. The same process applies in the other direction when a non-encrypted packet arrives at the NGW. Thanks to the IP forwarding based on the IPv6 prefix, the NGW identifies the Device and compresses headers using the Device's Rules.

In the second example, Figure 2, the SCHC compression is applied in the CoAP layer, compressing the CoAP header independently of the other layers. The RuleID, the Compression Residue, and CoAP payload are encrypted using a mechanism such as DTLS. Only the other end (App) can decipher the information. If needed, layers below use SCHC to compress the header as defined in [RFC8724] (represented in dotted lines).

This use case needs an end-to-end context initialization between the Device and the Application. The context initialization is out of the scope of this document.

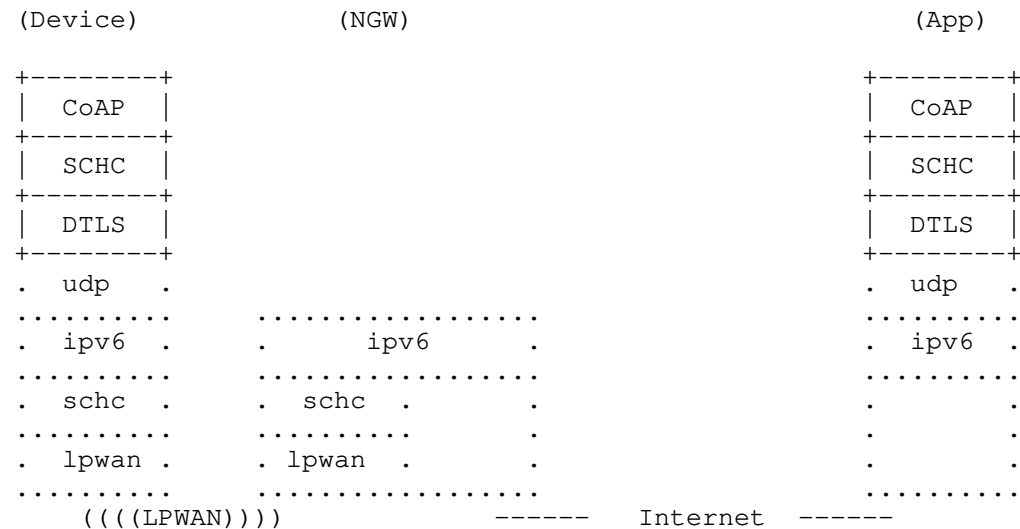


Figure 2: Standalone CoAP end-to-end Compression/Decompression

The third example, Figure 3, shows the use of Object Security for Constrained RESTful Environments (OSCORE) [RFC8613]. In this case, SCHC needs two Rules to compress the CoAP header. A first Rule focused on the inner header. The result of this first compression is encrypted using the OSCORE mechanism. Then a second Rule compresses the outer header, including the OSCORE Options.

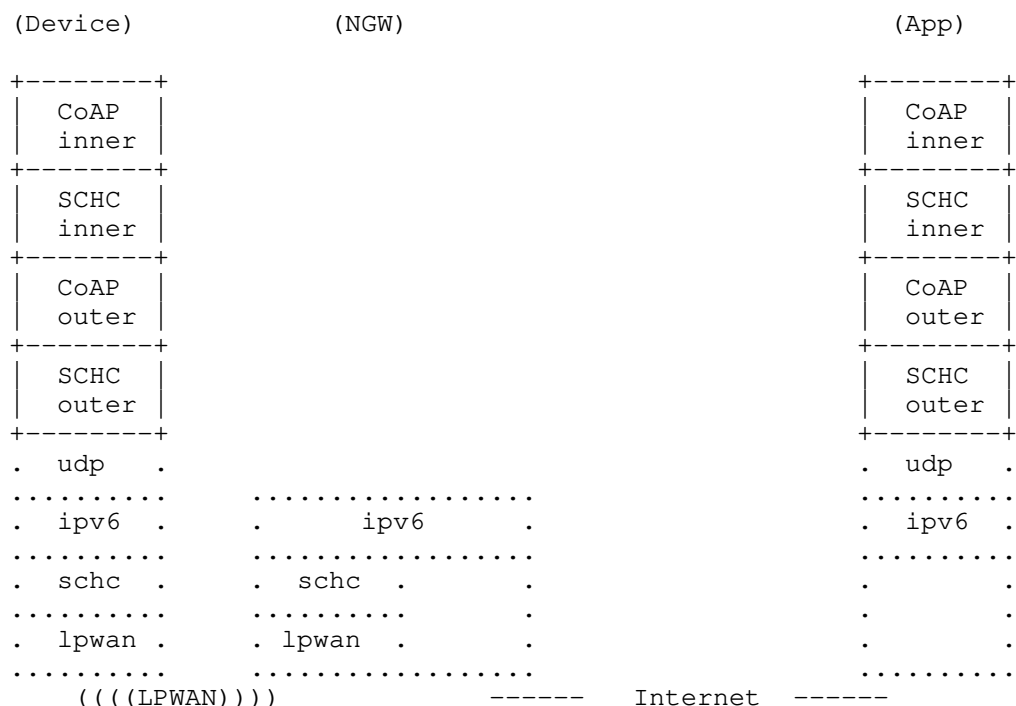


Figure 3: OSCORE compression/decompression.

In the case of several SCHC instances, as shown in Figure 2 and Figure 3, the Rules may come from different provisioning domains.

This document focuses on CoAP compression represented in the dashed boxes in the previous figures.

3. CoAP Headers compressed with SCHC

The use of SCHC over the CoAP header uses the same description, and compression/decompression techniques like the one for IP and UDP explained in the [RFC8724]. For CoAP, the SCHC Rules description uses the direction information to optimize the compression by reducing the number of Rules needed to compress headers. The field description MAY define both request/response headers and target values in the same Rule, using the DI (direction indicator) to make the difference.

As for other header compression protocols, when the compressor does not find a correct Rule to compress the header, the packet MUST be

sent uncompressed using the RuleID dedicated to this purpose. Where the Compression Residue is the complete header of the packet. See section 6 of [RFC8724].

3.1. Differences between CoAP and UDP/IP Compression

CoAP compression differs from IPv6 and UDP compression in the following aspects:

- o The CoAP protocol is asymmetric; the headers are different for a request or a response. For example, the URI-Path option is mandatory in the request, and it might not be present in the response. A request might contain an Accept option, and the response might include a Content-Format option. In comparison, IPv6 and UDP return path swap the value of some fields in the header. However, all the directions have the same fields (e.g., source and destination address fields).

The [RFC8724] defines the use of a direction indicator (DI) in the Field Descriptor, which allows a single Rule to process a message header differently depending on the direction.

- o Even when a field is "symmetric" (i.e., found in both directions), the values carried in each direction are different. The compression may use a "match-mapping" MO to limit the range of expected values in a particular direction and reduce the Compression Residue's size. Through the direction indicator (DI), a field description in the Rules splits the possible field value into two parts, one for each direction. For instance, if a client sends only CON requests, the Type can be elided by compression, and the answer may use one single bit to carry either the ACK or RST type. The field Code has the same behavior, the 0.0X code format value in the request, and the Y.ZZ code format in the response.
- o In SCHC, the Rule defines the different header fields' length, so SCHC does not need to send it. In IPv6 and UDP headers, the fields have a fixed size, known by definition. On the other hand, some CoAP header fields have variable lengths, and the Rule description specifies it. For example, in a URI-path or URI-query, the Token size may vary from 0 to 8 bytes, and the CoAP options use the Type-Length-Value encoding format.

When doing SCHC compression of a variable-length field, Section 7.5.2 from [RFC8724] offers the possibility to define a function for the Field length in the Field Description to know the length before compression. If the field length is unknown, the

Rule will set it as a variable, and SCHC will send the compressed field's length in the Compression Residue.

- o A field can appear several times in the CoAP headers. It is found typically for elements of a URI (path or queries). The SCHC specification [RFC8724] allows a Field ID to appear several times in the Rule and uses the Field Position (FP) to identify the correct instance, thereby removing the matching operation's ambiguity.
- o Field lengths defined in the CoAP protocol can be too large regarding LPWAN traffic constraints. For instance, this is particularly true for the Message-ID field and the Token field. SCHC uses different Matching operators (MO) to perform the compression. See section 7.4 of [RFC8724]. In this case, SCHC can apply the Most Significant Bits (MSB) MO to reduce the information carried on LPWANs.

4. Compression of CoAP header fields

This section discusses the compression of the different CoAP header fields. The CoAP compression with SCHC follows Section 7.1 of [RFC8724].

4.1. CoAP version field

CoAP version is bidirectional and MUST be elided during the SCHC compression since it always contains the same value. In the future, or if a new version of CoAP is defined, new Rules will be needed to avoid ambiguities between versions.

4.2. CoAP type field

The CoAP protocol [RFC7252] has four types of messages: two requests (CON, NON), one response (ACK), and one empty message (RST).

The SCHC compression SHOULD elide this field if, for instance, a client is sending only NON or only CON messages. For the RST message, SCHC may use a dedicated Rule. For other usages, SCHC can use a "match-mapping" MO.

4.3. CoAP code field

The code field is an IANA registry [RFC7252], and it indicates the Request Method used in CoAP. The compression of the CoAP code field follows the same principle as that of the CoAP type field. If the Device plays a specific role, SCHC may split the code values into two fields description, the request codes with the 0 class and the

response values. SCHC will use the direction indicator to identify the correct value in the packet.

If the Device only implements a CoAP client, SCHC compression may reduce the request code to the set of requests the client can process.

For known values, SCHC can use a "match-mapping" MO. If SCHC cannot compress the code field, it will send the values in the Compression Residue.

4.4. CoAP Message ID field

SCHC can compress the Message ID field with the "MSB" MO and the "LSB" CDA. See section 7.4 of [RFC8724].

4.5. CoAP Token fields

CoAP defines the Token using two CoAP fields, Token Length in the mandatory header and Token Value directly following the mandatory CoAP header.

SCHC processes the Token length as any header field. If the value does not change, the size can be stored in the TV and elided during the transmission. Otherwise, SCHC will send the token length in the Compression Residue.

For the Token Value, SCHC MUST NOT send it as a variable-length in the Compression Residue to avoid ambiguity with Token Length. Therefore, SCHC MUST use the Token length value to define the size of the Compression Residue. SCHC designates a specific function "tkl" that the Rule MUST use to complete the field description. During the decompression, this function returns the value contained in the Token Length field.

5. CoAP options

CoAP defines options placed after the basic header in Option Numbers order; see [RFC7252]. Each Option instance in a message uses the format Delta-Type (D-T), Length (L), Value (V). The SCHC Rule builds the description of the option by using in the Field ID the Option Number built from D-T; in TV, the Option Value; and the Option Length uses section 7.4 of [RFC8724]. When the Option Length has a well-known size, the Rule may keep the length value. Therefore, SCHC compression does not send it. Otherwise, SCHC Compression carries the length of the Compression Residue, in addition to the Compression Residue value.

CoAP requests and responses do not include the same options. So Compression Rules may reflect this asymmetry by tagging the direction indicator.

Note that length coding differs between CoAP options and SCHC variable size Compression Residue.

The following sections present how SCHC compresses some specific CoAP options.

If CoAP introduces a new option, the SCHC Rules MAY be updated, and the new Field ID description MUST be assigned to allow its compression. Otherwise, if no Rule describes this new option, the SCHC compression is not achieved, and SCHC sends the CoAP header without compression.

5.1. CoAP Content and Accept options.

If the client expects a single value, it can be stored in the TV and elided during the transmission. Otherwise, if the client expects several possible values, a "match-mapping" SHOULD be used to limit the Compression Residue's size. If not, SCHC has to send the option value in the Compression Residue (fixed or variable length).

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

SCHC compresses these three fields in the same way. When the value of these options is known, SCHC can elide these fields. If the option uses well-known values, SCHC can use a "match-mapping" MO. Otherwise, SCHC will use "value-sent" MO, and the Compression Residue will send these options' values.

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

The Uri-Path and Uri-Query fields are repeatable options; this means that in the CoAP header, they may appear several times with different values. SCHC Rule description uses the Field Position (FP) to distinguish the different instances in the path.

To compress repeatable field values, SCHC may use a "match-mapping" MO to reduce the size of variable Paths or Queries. In these cases, to optimize the compression, several elements can be regrouped into a single entry. The Numbering of elements does not change, and the first matching element sets the MO comparison.

Field	FL	FP	DI	Target Value	Matching Operator	CDA
Uri-Path		1	up	["/a/b", "/c/d"]	match-mapping	mapping-sent
Uri-Path	var	3	up		ignore	value-sent

Figure 4: complex path example

In Figure 4, SCHC can use a single bit in the Compression Residue to code one of the two paths. If regrouping were not allowed, 2 bits in the Compression Residue would be needed. SCHC sends the third path element as a variable size in the Compression Residue.

The length of URI-Path and URI-Query may be known when the rule is defined. In any case, SCHC MUST set the field length to variable. The unit to indicate the Compression Residue size is in Byte.

SCHC compression can use the MSB MO to a Uri-Path or Uri-Query element. However, attention to the length is important because the MSB value is in bits, and the size MUST always be a multiple of 8 bits.

The length sent at the beginning of a variable-length Compression Residue indicates the LSB's size in bytes.

For instance, for a CORECONF path /c/X6?k="eth0" the Rule description can be:

Field	FL	FP	DI	Target Value	Match Opera.	CDA
Uri-Path		1	up	"c"	equal	not-sent
Uri-Path	var	2	up		ignore	value-sent
Uri-Query	var	1	up	"k=\""	MSB(24)	LSB

Figure 5: CORECONF URI compression

Figure 5 shows the Rule description for a URI-Path and a URI-Query. SCHC compresses the first part of the URI-Path with a "not-sent" CDA. SCHC will send the second element of the URI-Path with the length (i.e., 0x2 X 6) followed by the query option (i.e., 0x05 eth0").

5.3.1. Variable number of Path or Query elements

SCHC fixed the number of Uri-Path or Uri-Query elements in a Rule at the Rule creation time. If the number varies, SCHC SHOULD create several Rules to cover all the possibilities. Another one is to define the length of Uri-Path to variable and sends a Compression Residue with a length of 0 to indicate that this Uri-Path is empty. However, this adds 4 bits to the variable Compression Residue size. See section 7.5.2 [RFC8724].

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

The SCHC Rule description MAY define sending some field values by setting the TV to "not-sent," MO to "ignore," and CDA to "value-sent." A Rule MAY also use a "match-mapping" when there are different options for the same FID. Otherwise, the Rule sets the TV to the value, MO to "equal," and CDA to "not-sent."

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

A Rule entry cannot store these fields' values. The Rule description MUST always send these values in the Compression Residue.

6. SCHC compression of CoAP extension RFCs

6.1. Block

When a packet uses a Block [RFC7959] option, SCHC compression MUST send its content in the Compression Residue. The SCHC Rule describes an empty TV with a MO set to "ignore" and a CDA to "value-sent." Block option allows fragmentation at the CoAP level that is compatible with SCHC fragmentation. Both fragmentation mechanisms are complementary, and the node may use them for the same packet as needed.

6.2. Observe

The [RFC7641] defines the Observe option. The SCHC Rule description will not define the TV, but MO to "ignore," and the CDA to "value-sent." SCHC does not limit the maximum size for this option (3 bytes). To reduce the transmission size, either the Device implementation MAY limit the delta between two consecutive values, or a proxy can modify the increment.

Since the Observe option MAY use an RST message to inform a server that the client does not require the Observe response, a specific

SCHC Rule SHOULD exist to allow the message's compression with the RST type.

6.3. No-Response

The [RFC7967] defines a No-Response option limiting the responses made by a server to a request. Different behaviors exist while using this option to limit the responses made by a server to a request. If both ends know the value, then the SCHC Rule will describe a TV to this value, with a MO set to "equal" and CDA set to "not-sent."

Otherwise, if the value is changing over time, the SCHC Rule will set the MO to "ignore" and CDA to "value-sent." The Rule may also use a "match-mapping" to compress this option.

6.4. OSCORE

OSCORE [RFC8613] defines end-to-end protection for CoAP messages. This section describes how SCHC Rules can be applied to compress OSCORE-protected messages.

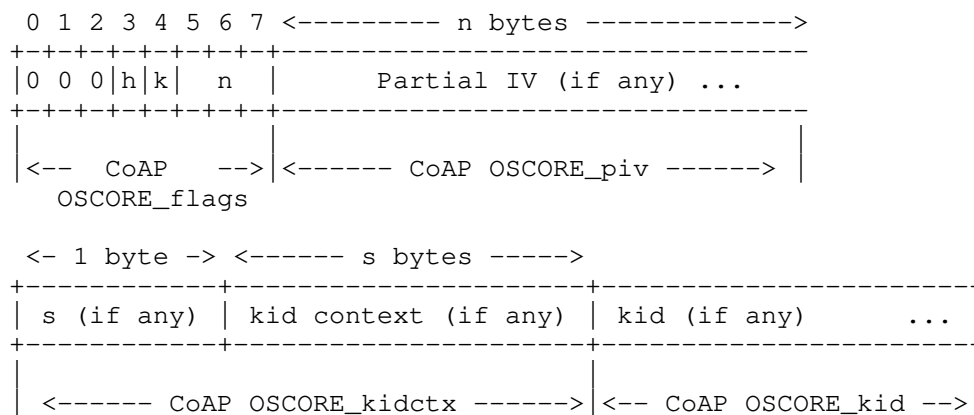


Figure 6: OSCORE Option

The Figure 6 shows the OSCORE Option Value encoding defined in Section 6.1 of [RFC8613], where the first byte specifies the Content of the OSCORE options using flags. The three most significant bits of this byte 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 three least significant bits n indicate the length of the piv (Partial Initialization Vector) field in bytes. When n = 0, no piv is present.

The flag byte is followed by the piv field, kid context field, and kid field in this order, and if present, the kid context field's length is encoded in the first byte denoting by 's' the length of the kid context in bytes.

To better perform OSCORE SCHC compression, the Rule description needs to identify the OSCORE Option and the fields it contains. Conceptually, it discerns up to 4 distinct pieces of information within the OSCORE option: the flag bits, the piv, the kid context, and the kid. The SCHC Rule splits into four field descriptions the OSCORE option to compress them:

- o CoAP OSCORE_flags,
- o CoAP OSCORE_piv,
- o CoAP OSCORE_kidctx,
- o CoAP OSCORE_kid.

Figure 6 shows the OSCORE Option format with those four fields superimposed on it. Note that the CoAP OSCORE_kidctx field directly includes the size octet s.

7. Examples of CoAP header compression

7.1. Mandatory header with CON message

In this first scenario, the SCHC Compressor at the Network Gateway side receives a POST message from an Internet client, which is immediately acknowledged by the Device. Figure 7 describes the SCHC Rule descriptions for this scenario.

RuleID 1

Field	FL	FP	DI	Target Value	Match Opera.	CDA	Sent [bits]
CoAP version	2	1	bi	01	equal	not-sent	T
CoAP Type	2	1	dw	CON	equal	not-sent	
CoAP Type	2	1	up	[ACK, RST]	match-mapping	matching-sent	
CoAP TKL	4	1	bi	0	equal	not-sent	
CoAP Code	8	1	bi	[0.00, ... 5.05]	match-mapping	matching-sent	CC CCC M-ID
CoAP MID	16	1	bi	0000	MSB(7)	LSB	
CoAP Uri-Path	var	1	dw	path	equal 1	not-sent	

Figure 7: CoAP Context to compress header without Token

In this example, SCHC compression elides the version and the Token Length fields. The 26 method and response codes defined in [RFC7252] has been shrunk to 5 bits using a "match-mapping" MO. The Uri-Path contains a single element indicated in the TV and elided with the CDA "not-sent."

SCHC Compression reduces the header sending only the Type, a mapped code, and the least significant bits of Message ID (9 bits in the example above).

Note that a client located in an Application Server sending a request to a server located in the Device may not be compressed through this Rule since the MID might not start with 7 bits equal to 0. A CoAP proxy placed before the SCHC C/D can rewrite the message ID to fit the value and match the Rule.

7.2. OSCORE Compression

OSCORE aims to solve the problem of end-to-end encryption for CoAP messages. Therefore, the goal is to hide as much as possible the message 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 sensitive information that is not necessary for proxy operation. However, it is part of the message that can be encrypted

until it reaches its end destination. The Outer Message acts as a shell matching the regular CoAP message format and includes all Options and information needed for proxy operation and caching. Figure 8 illustrates this analysis.

The CoAP protocol arranges the options 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.

These classes point out that the Outer option contains the OSCORE Option and that the message is OSCORE protected; this option carries the information necessary to retrieve the Security Context. The endpoint will use this Security Context to decrypt the message correctly.

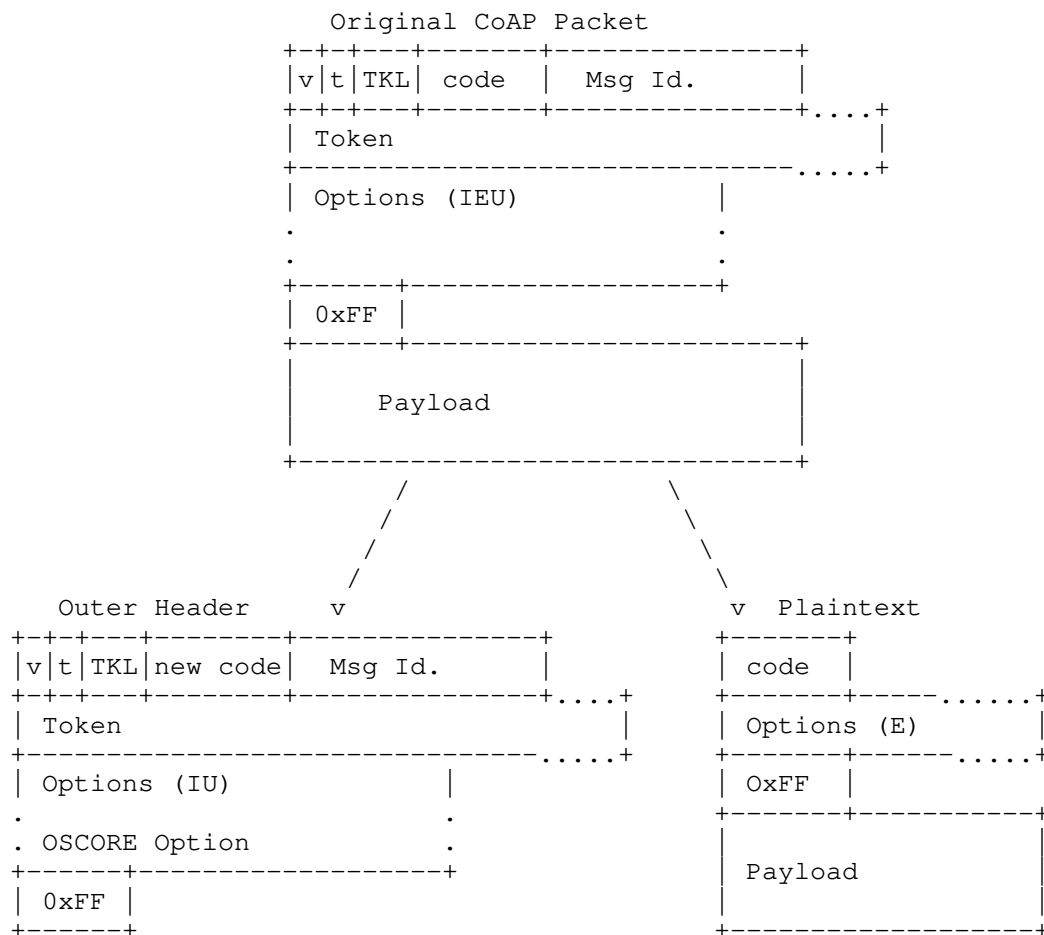


Figure 8: A CoAP packet is split into an OSCORE outer and plaintext

Figure 8 shows the packet format for the OSCORE Outer header and Plaintext.

In the Outer Header, the original header code is hidden and replaced by a default dummy value. As seen in Sections 4.1.3.5 and 4.2 of [RFC8613], the message code is replaced by POST for requests and Changed for responses when CoAP is not using the Observe option. If CoAP uses Observe, the OSCORE message code is replaced by FETCH for requests and Content for responses.

The first byte of the Plaintext contains the original packet code, followed by the message code, the class E options, and, if present, the original message Payload preceded by its payload marker.

An AEAD algorithm now encrypts the Plaintext. This integrity protects the Security Context parameters and, eventually, any class I options from the Outer Header. The resulting Ciphertext becomes the new payload of the OSCORE message, as illustrated in Figure 9.

As defined in [RFC5116], this Ciphertext is the encrypted Plaintext's concatenation of the authentication tag. Note that Inner Compression only affects the Plaintext before encryption. Thus only the first variable-length of the Ciphertext can be reduced. The authentication tag is fixed in length and is considered part of the cost of protection.

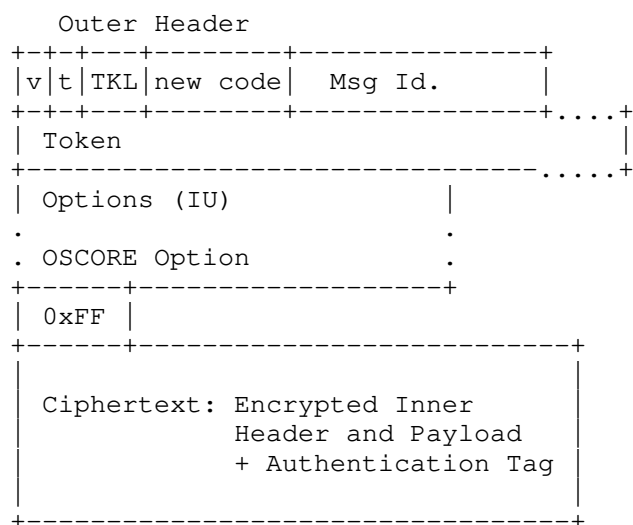


Figure 9: OSCORE message

The SCHC Compression scheme consists of compressing both the Plaintext before encryption and the resulting OSCORE message after encryption, see Figure 10.

The OSCORE message translates into a segmented process where SCHC compression is applied independently in 2 stages, each with its corresponding set of Rules, with the Inner SCHC Rules and the Outer

SCHC Rules. This way, compression is applied to all fields of the original CoAP message.

Note that since the corresponding end-point can only decrypt the Inner part of the message, this end-point will also have to implement Inner SCHC Compression/Decompression.

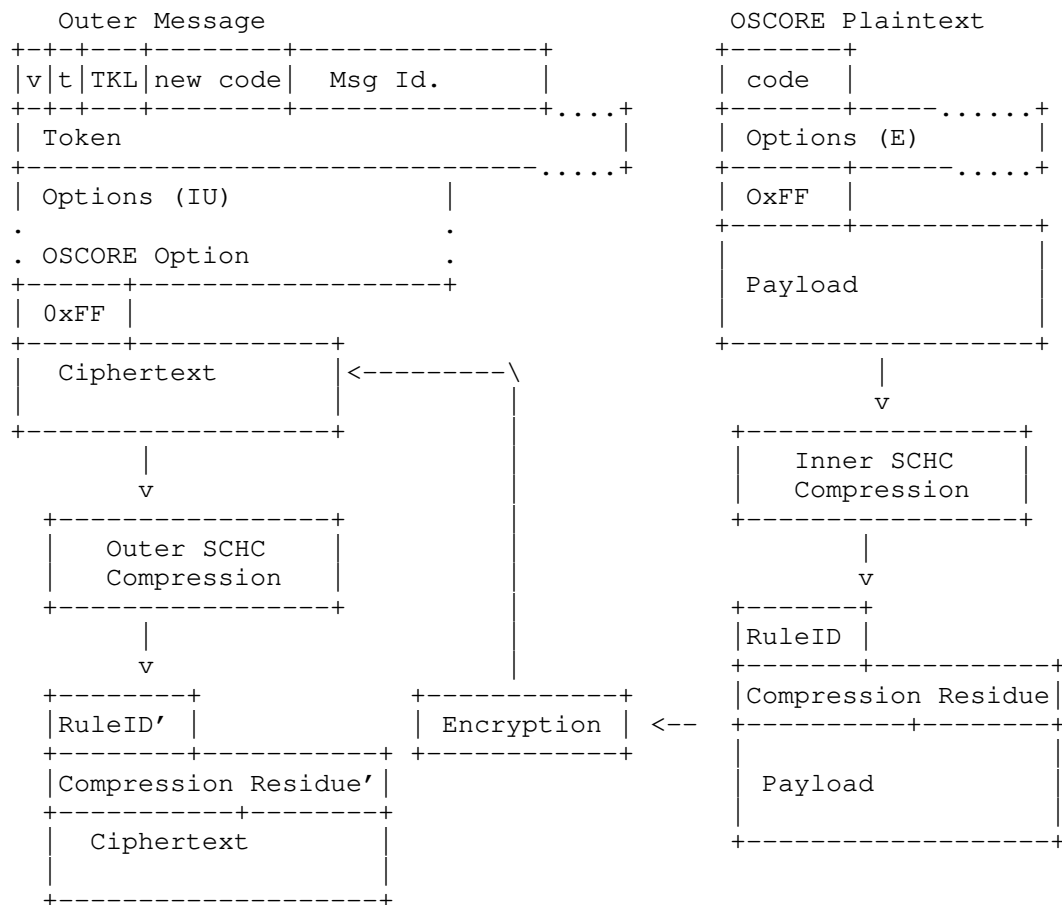


Figure 10: OSCORE Compression Diagram

7.3. Example OSCORE Compression

This section gives an example with a GET Request and its consequent Content Response from a Device-based CoAP client to a cloud-based CoAP server. The example also describes a possible set of Rules for the Inner and Outer SCHC Compression. A dump of the results and a

contrast between SCHC + OSCORE performance with SCHC + COAP performance is also listed. This example gives an approximation of the cost of security with SCHC-OSCORE.

Our first CoAP message is the GET request in Figure 11.

Original message:

=====

0x4101000182bb74656d7065726174757265

Header:

0x4101

01 Ver

00 CON

0001 TKL

00000001 Request Code 1 "GET"

0x0001 = mid

0x82 = token

Options:

0xbb74656d7065726174757265

Option 11: URI_PATH

Value = temperature

Original msg length: 17 bytes.

Figure 11: CoAP GET Request

Its corresponding response is the CONTENT Response in Figure 12.

Original message:

=====

0x6145000182ff32332043

Header:

0x6145

01 Ver

10 ACK

0001 TKL

01000101 Successful Response Code 69 "2.05 Content"

0x0001 = mid

0x82 = token

0xFF Payload marker

Payload:

0x32332043

Original msg length: 10

Figure 12: CoAP CONTENT Response

The SCHC Rules for the Inner Compression include all fields already present in a regular CoAP message. The methods described in Section 4 apply to these fields. As an example, see Figure 13.

RuleID 0

Field	FL	FP	DI	Target Value	MO	CDA	Sent [bits]
CoAP Code	8	1	up	1	equal	not-sent	
CoAP Code	8	1	dw	[69, 132]	match-mapping	mapping-sent	c
CoAP Uri-Path		1	up	temperature	equal	not-sent	

Figure 13: Inner SCHC Rules

Figure 14 shows the Plaintext obtained for the example GET request. The packet follows the process of Inner Compression and Encryption until the payload. The outer OSCORE Message adds the result of the Inner process.

In this case, the original message has no payload, and its resulting Plaintext compressed up to only 1 byte (size of the RuleID). The AEAD algorithm preserves this length in its first output and yields a

fixed-size tag. SCHC cannot compress the tag, and the OSCORE message must include it without compression. The use of integrity protection translates into an overhead in total message length, limiting the amount of compression that can be achieved and plays into the cost of adding security to the exchange.

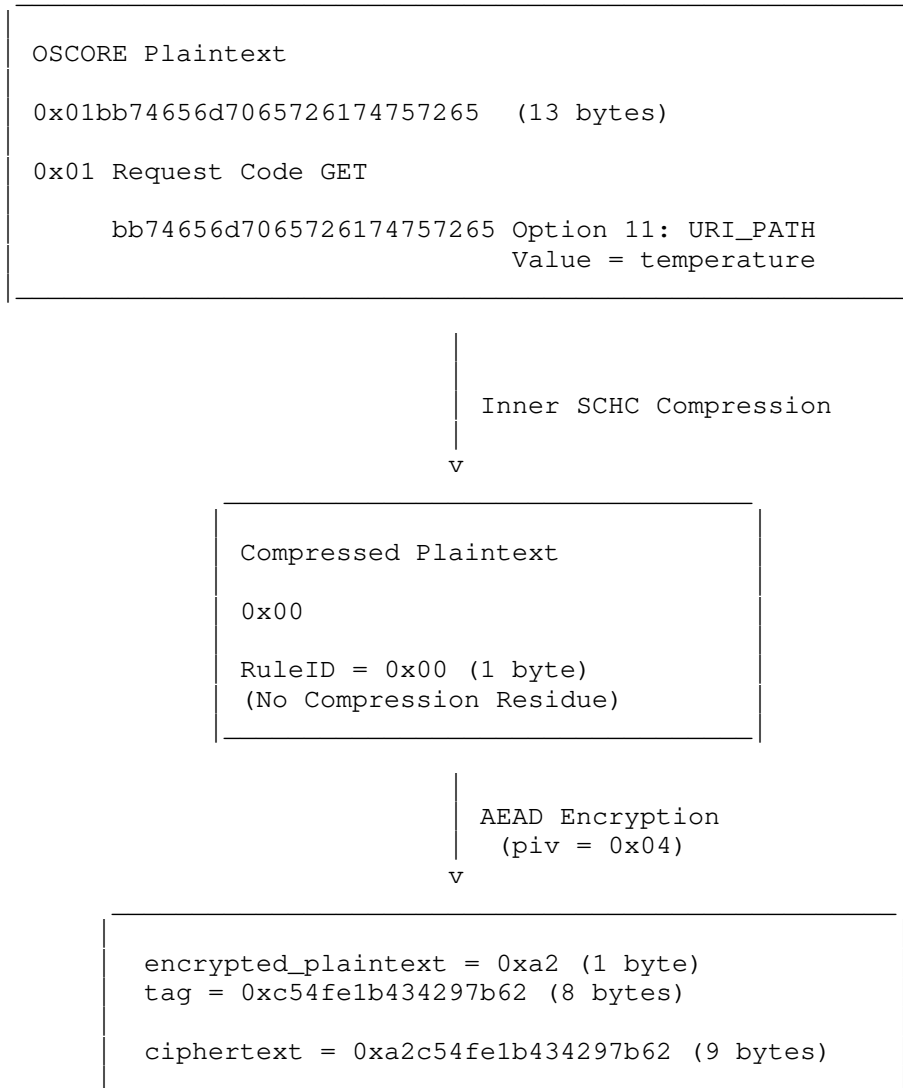


Figure 14: Plaintext compression and encryption for GET Request

Figure 15 shows the process for the example CONTENT Response. The Compression Residue is 1 bit long. Note that since SCHC adds padding after the payload, this misalignment causes the hexadecimal code from the payload to differ from the original, even if SCHC cannot compress the tag. The overhead for the tag bytes limits the SCHC's performance but brings security to the transmission.

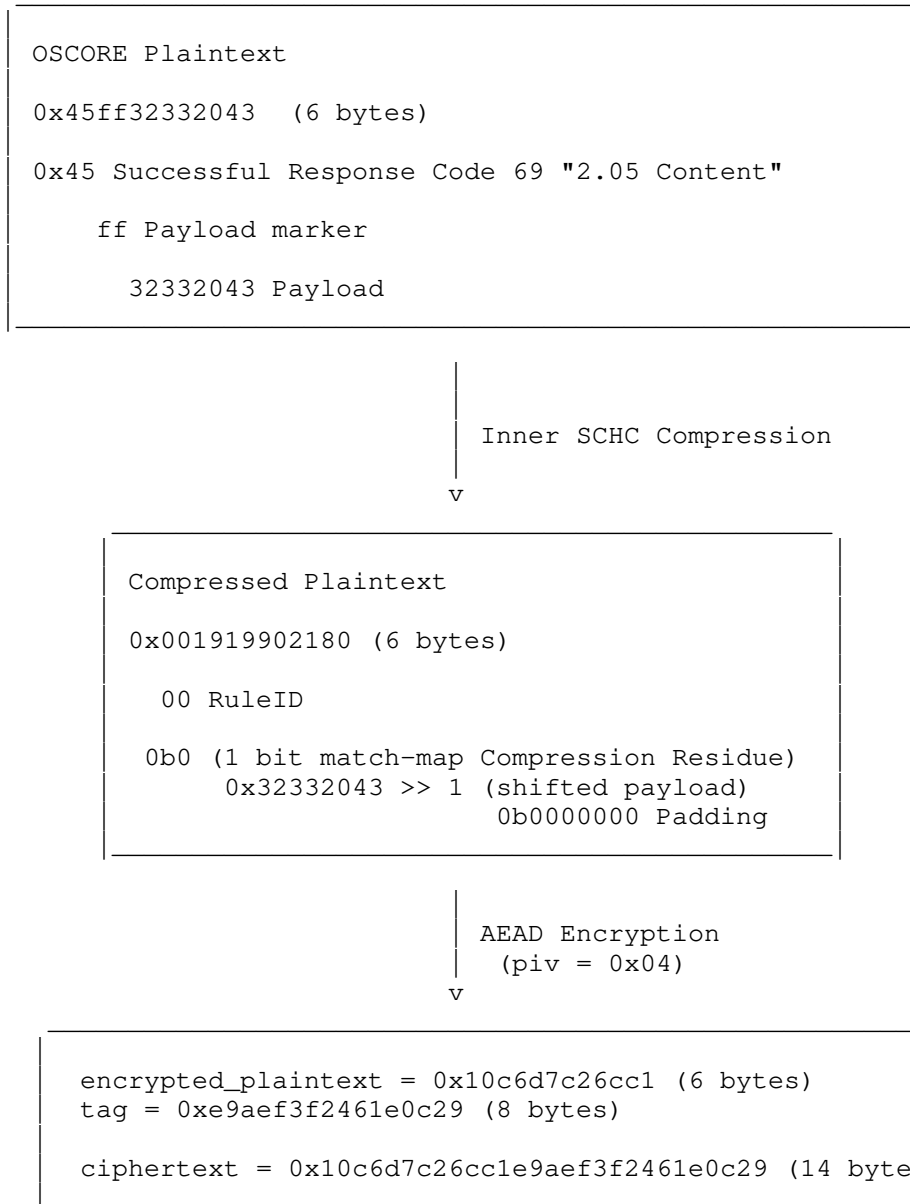


Figure 15: Plaintext compression and encryption for CONTENT Response

The Outer SCHC Rules (Figure 18) must process the OSCORE Options fields. Figure 16 and Figure 17 shows a dump of the OSCORE Messages

generated from the example messages. They include the Inner Compressed Ciphertext in the payload. These are the messages that have to be compressed by the Outer SCHC Compression.

Protected message:

=====

0x4102000182d8080904636c69656e74ffa2c54fe1b434297b62
(25 bytes)

Header:

0x4102

01 Ver

00 CON

0001 TKL

00000010 Request Code 2 "POST"

0x0001 = mid

0x82 = token

Options:

0xd8080904636c69656e74 (10 bytes)

Option 21: OBJECT_SECURITY

Value = 0x0904636c69656e74

09 = 000 0 1 001 Flag byte

h k n

04 piv

636c69656e74 kid

0xFF Payload marker

Payload:

0xa2c54fe1b434297b62 (9 bytes)

Figure 16: Protected and Inner SCHC Compressed GET Request

```

Protected message:
=====
0x6144000182d008ff10c6d7c26cc1e9aef3f2461e0c29
(22 bytes)

Header:
0x6144
01    Ver
    10    ACK
        0001    TKL
            01000100    Successful Response Code 68 "2.04 Changed"

0x0001 = mid
0x82 = token

Options:
0xd008 (2 bytes)
Option 21: OBJECT_SECURITY
Value = b''

0xFF    Payload marker
Payload:
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)

```

Figure 17: Protected and Inner SCHC Compressed CONTENT Response

For the flag bits, some SCHC compression methods are useful, depending on the Application. The most straightforward alternative is to provide a fixed value for the flags, combining MO "equal" and CDA "not-sent." This SCHC definition saves most bits but could prevent flexibility. Otherwise, SCHC could use a "match-mapping" MO to choose from several configurations for the exchange. If not, the SCHC description may use an "MSB" MO to mask off the three hard-coded most significant bits.

Note that fixing a flag bit will limit CoAP Options choice that can be used in the exchange since their values are dependent on specific options.

The piv field lends itself to having some bits masked off with "MSB" MO and "LSB" CDA. This SCHC description could be useful in applications where the message frequency is low such as LPWAN technologies. Note that compressing the sequence numbers may reduce the maximum number of sequence numbers that can be used in an exchange. Once the sequence number exceeds the maximum value, the OSCORE keys need to be re-established.

The size *s* included in the kid context field MAY be masked off with "LSB" CDA. The rest of the field could have additional bits masked off or have the whole field fixed with MO "equal" and CDA "not-sent." The same holds for the kid field.

Figure 18 shows a possible set of Outer Rules to compress the Outer Header.

RuleID 0

Field	FL	FP	DI	Target Value	MO	CDA	Sent [bits]
CoAP version	2	1	bi	01	equal	not-sent	
CoAP Type	2	1	up	0	equal	not-sent	
CoAP Type	2	1	dw	2	equal	not-sent	
CoAP TKL	4	1	bi	1	equal	not-sent	
CoAP Code	8	1	up	2	equal	not-sent	
CoAP Code	8	1	dw	68	equal	not-sent	
CoAP MID	16	1	bi	0000	MSB(12)	LSB	MMMM
CoAP Token	tkl	1	bi	0x80	MSB(5)	LSB	TTT
CoAP OSCORE_flags	8	1	up	0x09	equal	not-sent	
CoAP OSCORE_piv	var	1	up	0x00	MSB(4)	LSB	PPPP
COAP OSCORE_kid	var	1	up	0x636c69656e70	MSB(52)	LSB	KKKK
COAP OSCORE_kidctx	var	1	bi	b''	equal	not-sent	
CoAP OSCORE_flags	8	1	dw	b''	equal	not-sent	
CoAP OSCORE_piv	var	1	dw	b''	equal	not-sent	
CoAP OSCORE_kid	var	1	dw	b''	equal	not-sent	

Figure 18: Outer SCHC Rules

The Outer Rule of Figure 18 is applied to the example GET Request and CONTENT Response. Figure 19 and Figure 20 show the resulting messages.

Compressed message:

=====

0x001489458a9fc3686852f6c4 (12 bytes)

0x00 RuleID

1489 Compression Residue

458a9fc3686852f6c4 Padded payload

Compression Residue:

0b 0001 010 0100 0100 (15 bits -> 2 bytes with padding)

mid tkn piv kid

Payload

0xa2c54fe1b434297b62 (9 bytes)

Compressed message length: 12 bytes

Figure 19: SCHC-OSCORE Compressed GET Request

Compressed message:

=====

0x0014218daf84d983d35de7e48c3c1852 (16 bytes)

0x00 RuleID

14 Compression Residue

218daf84d983d35de7e48c3c1852 Padded payload

Compression Residue:

0b0001 010 (7 bits -> 1 byte with padding)

mid tkn

Payload

0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)

Compressed msg length: 16 bytes

Figure 20: SCHC-OSCORE Compressed CONTENT Response

In contrast, comparing these results with what would be obtained by SCHC compressing the original CoAP messages without protecting them with OSCORE is done by compressing the CoAP messages according to the SCHC Rules in Figure 21.

RuleID 1

Field	FL	FP	DI	Target Value	MO	CDA	Sent [bits]
CoAP version	2	1	bi	01	equal	not-sent	
CoAP Type	2	1	up	0	equal	not-sent	
CoAP Type	2	1	dw	2	equal	not-sent	
CoAP TKL	4	1	bi	1	equal	not-sent	
CoAP Code	8	1	up	2	equal	not-sent	
CoAP Code	8	1	dw	[69,132]	match-mapping	mapping-sent	
CoAP MID	16	1	bi	0000	MSB(12)	LSB	C MMMM
CoAP Token	tkl	1	bi	0x80	MSB(5)	LSB	TTT
CoAP Uri-Path		1	up	temperature	equal	not-sent	

Figure 21: SCHC-CoAP Rules (No OSCORE)

Figure 21 Rule yields the SCHC compression results in Figure 22 for request, and Figure 23 for the response.

Compressed message:

=====

0x0114

0x01 = RuleID

Compression Residue:

0b00010100 (1 byte)

Compressed msg length: 2

Figure 22: CoAP GET Compressed without OSCORE

Compressed message:

=====

0x010a32332043

0x01 = RuleID

Compression Residue:

0b00001010 (1 byte)

Payload

0x32332043

Compressed msg length: 6

Figure 23: 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.

8. IANA Considerations

This document has no request to IANA.

9. Security considerations

The use of SCHC header compression for CoAP header fields only affects the representation of the header information. SCHC header compression itself does not increase or decrease the overall level of security of the communication. When the connection does not use a security protocol (such as OSCORE, DTLS, etc.), it is necessary to use a layer-two security mechanism to protect the SCHC messages.

If LPWAN is the layer-two technology, the SCHC security considerations of [RFC8724] continue to apply. When using another layer-two protocol, use of a cryptographic integrity-protection mechanisms to protect the SCHC headers is REQUIRED. Such cryptographic integrity protection is necessary in order to continue to provide the properties that [RFC8724] relies upon.

When SCHC is used with OSCORE, the security considerations of [RFC8613] continue to apply.

When SCHC is used with the OSCORE outer headers, the Initialization Vector (IV) size in the Compression Residue must be carefully selected. There is a tradeoff between compression efficiency (with a longer "MSB" MO prefix) and the frequency at which the Device must renew its key material (in order to prevent the IV from expanding to

an uncompressable value). The key renewal operation itself requires several message exchanges and requires energy-intensive computation, but the optimal tradeoff will depend on the specifics of the device and expected usage patterns.

If an attacker can introduce a corrupted SCHC-compressed packet onto a link, DoS attacks are possible by causing excessive resource consumption at the decompressor. However, an attacker able to inject packets at the link layer is also capable of other, potentially more damaging, attacks.

SCHC compression emits variable-length Compression Residues for some CoAP fields. In the compressed header representation, the length field that is sent is not the length of the original header field but rather the length of the Compression Residue that is being transmitted. If a corrupted packet arrives at the decompressor with a longer or shorter length than the original compressed representation possessed, the SCHC decompression procedures will detect an error and drop the packet.

SCHC header compression rules MUST remain tightly coupled between compressor and decompressor. If the compression rules get out of sync, a Compression Residue might be decompressed differently at the receiver than the initial message submitted to compression procedures. Accordingly, any time the context Rules are updated on an OSCORE endpoint, that endpoint MUST trigger OSCORE key re-establishment. Similar procedures may be appropriate to signal Rule updates when other message-protection mechanisms are in use.

10. Acknowledgements

The authors would like to thank (in alphabetic order): Christian Amsuss, Dominique Barthel, Carsten Bormann, Theresa Enghardt, Thomas Fossati, Klaus Hartke, Benjamin Kaduk, Francesca Palombini, Alexander Pelov, Goran Selander and Eric Vyncke.

11. 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>>.
- [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <<https://www.rfc-editor.org/info/rfc5116>>.

- [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>>.
- [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>>.
- [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <<https://www.rfc-editor.org/info/rfc8613>>.
- [RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <<https://www.rfc-editor.org/info/rfc8724>>.

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: June 7, 2020

A. Minaburo
Acklio
L. Toutain
IMT-Atlantique
C. Gomez
Universitat Politecnica de Catalunya
D. Barthel
Orange Labs
JC. Zuniga
SIGFOX
December 05, 2019

Static Context Header Compression (SCHC) and fragmentation for LPWAN,
application to UDP/IPv6
draft-ietf-lpwan-ipv6-static-context-hc-24

Abstract

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

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

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

The SCHC header compression and fragmentation mechanisms are independent of the specific LPWAN technology over which they are used. This document defines generic functionalities and offers flexibility with regard to parameter settings and mechanism choices. This document standardizes the exchange over the LPWAN between two SCHC entities. Settings and choices specific to a technology or a product are expected to be grouped into profiles, which are specified in other documents. Data models for the context and profiles are out of scope.

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 June 7, 2020.

Copyright Notice

Copyright (c) 2019 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	8
5.1. SCHC Packet format	10
5.2. Functional mapping	11
6. Rule ID	12
7. Compression/Decompression	12
7.1. SCHC C/D Rules	13
7.2. Rule ID for SCHC C/D	15
7.3. Packet processing	15
7.4. Matching operators	17
7.5. Compression Decompression Actions (CDA)	18

7.5.1.	processing fixed-length fields	19
7.5.2.	processing variable-length fields	19
7.5.3.	not-sent CDA	20
7.5.4.	value-sent CDA	20
7.5.5.	mapping-sent CDA	20
7.5.6.	LSB CDA	21
7.5.7.	DevIID, AppIID CDA	21
7.5.8.	Compute-*	21
8.	Fragmentation/Reassembly	22
8.1.	Overview	22
8.2.	SCHC F/R Protocol Elements	22
8.2.1.	Messages	22
8.2.2.	Tiles, Windows, Bitmaps, Timers, Counters	23
8.2.3.	Integrity Checking	25
8.2.4.	Header Fields	26
8.3.	SCHC F/R Message Formats	28
8.3.1.	SCHC Fragment format	28
8.3.2.	SCHC ACK format	30
8.3.3.	SCHC ACK REQ format	32
8.3.4.	SCHC Sender-Abort format	33
8.3.5.	SCHC Receiver-Abort format	33
8.4.	SCHC F/R modes	34
8.4.1.	No-ACK mode	34
8.4.2.	ACK-Always mode	36
8.4.3.	ACK-on-Error mode	43
9.	Padding management	51
10.	SCHC Compression for IPv6 and UDP headers	52
10.1.	IPv6 version field	52
10.2.	IPv6 Traffic class field	52
10.3.	Flow label field	52
10.4.	Payload Length field	53
10.5.	Next Header field	53
10.6.	Hop Limit field	53
10.7.	IPv6 addresses fields	53
10.7.1.	IPv6 source and destination prefixes	54
10.7.2.	IPv6 source and destination IID	54
10.8.	IPv6 extension headers	54
10.9.	UDP source and destination ports	55
10.10.	UDP length field	55
10.11.	UDP Checksum field	55
11.	IANA Considerations	56
12.	Security considerations	56
12.1.	Security considerations for SCHC Compression/Decompression	56
12.1.1.	Forged SCHC Packet	56
12.1.2.	Compressed packet size as a side channel to guess a secret token	57
12.1.3.	Decompressed packet different from the original	

packet	58
12.2. Security considerations for SCHC	
Fragmentation/Reassembly	58
12.2.1. Buffer reservation attack	58
12.2.2. Corrupt Fragment attack	59
12.2.3. Fragmentation as a way to bypass Network Inspection	59
12.2.4. Privacy issues associated with SCHC header fields .	59
13. Acknowledgements	60
14. References	60
14.1. Normative References	60
14.2. Informative References	61
Appendix A. Compression Examples	61
Appendix B. Fragmentation Examples	64
Appendix C. Fragmentation State Machines	72
Appendix D. SCHC Parameters	78
Appendix E. Supporting multiple window sizes for fragmentation .	80
Appendix F. ACK-Always and ACK-on-Error on quasi-bidirectional	
links	80
Authors' Addresses	82

1. Introduction

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

LPWAN technologies impose some strict limitations on traffic. For instance, devices sleep most of the time and may only receive data during short periods of time after transmission, in order to preserve battery. LPWAN technologies are also characterized by a greatly reduced data unit and/or payload size (see [RFC8376]).

Header compression is needed for efficient Internet connectivity to a node within an LPWAN network. The following properties of LPWAN networks can be exploited to get an efficient header compression:

- o The network topology is star-oriented, which means that all packets between the same source-destination pair follow the same path. For the needs of this document, the architecture can simply be described as Devices (Dev) exchanging information with LPWAN Application Servers (App) through a Network Gateway (NGW).
- o Because devices embed built-in applications, the traffic flows to be compressed are known in advance. Indeed, new applications are less frequently installed in an LPWAN device, than they are in a general-purpose computer or smartphone.

SCHC compression uses a Context (a set of Rules) in which information about header fields is stored. This Context is static: the values of the header fields and the actions to do compression/decompression do not change over time. This avoids the need for complex resynchronization mechanisms. Indeed, a return path may be more restricted/expensive, sometimes completely unavailable [RFC8376]. A compression protocol that relies on feedback is not compatible with the characteristics of such LPWANs.

In most cases, a small Rule 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.

Furthermore, some LPWAN technologies do not provide a fragmentation functionality; to support the IPv6 MTU requirement of 1280 bytes [RFC8200], they require a fragmentation protocol at the adaptation layer below IPv6. Accordingly, this document defines an optional fragmentation/reassembly mechanism for LPWAN technologies to support the IPv6 MTU requirement.

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

2. Requirements Notation

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

3. LPWAN Architecture

LPWAN network architectures are similar among them, but each LPWAN technology names architecture elements differently. In this document, we use terminology from [RFC8376], which identifies the following 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) 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 Application Server (App) is the end point of the application level protocol on the Internet side.

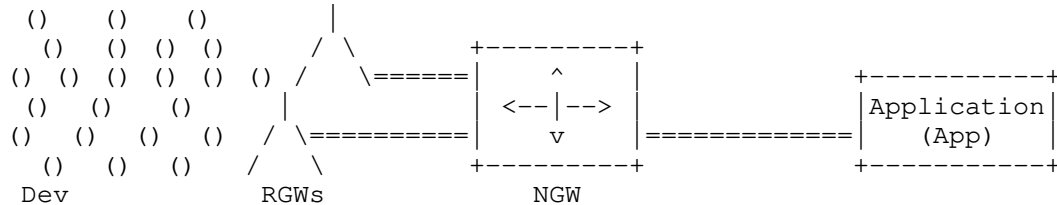


Figure 1: LPWAN Architecture, simplified from that shown in RFC8376

4. Terminology

This section defines the terminology and acronyms used in this document. It extends the terminology of [RFC8376].

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 App: LPWAN Application, as defined by [RFC8376]. An application sending/receiving packets to/from the Dev.
- o AppIID: Application Interface Identifier. The IID that identifies the application server interface.
- o Bi: Bidirectional. Characterizes a Field Descriptor that applies to headers of packets traveling in either direction (Up and Dw, see this glossary).
- o CDA: Compression/Decompression Action. Describes the pair of actions that are performed at the compressor to compress a header field and at the decompressor to recover the original value of the header field.
- o Compression Residue. The bits that remain to be sent (beyond the Rule ID itself) after applying the SCHC compression.
- o Context: A set of Rules used to compress/decompress headers.
- o Dev: Device, as defined by [RFC8376].
- 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 Field Description applies to. This allows for asymmetric processing, using the same Rule.
- o Dw: Downlink direction for compression/decompression, from SCHC C/D in the network to SCHC C/D in the Dev.
- o Field Description. A tuple containing identifier, value, matching operator and actions to be applied to a field.
- o FID: Field Identifier. This identifies the protocol and field a Field Description applies to.
- o FL: Field Length is the length of the original packet header field. It is expressed as a number of bits for header fields of fixed lengths or as a type (e.g., variable, token length, ...) for field lengths that are unknown at the time of Rule creation. The length of a header field is defined in the corresponding protocol specification (such as IPv6 or UDP).
- o FP: when a Field is expected to appear multiple times in a header, Field Position specifies the occurrence this Field Description applies to (for example, first uri-path option, second uri-path, etc. in a CoAP header), counting from 1. The value 0 is special and means "don't care", see Section 7.3.
- o IID: Interface Identifier. See the IPv6 addressing architecture [RFC7136].
- o L2: Layer two. The immediate lower layer SCHC interfaces with. It is provided by an underlying LPWAN technology. It does not necessarily correspond to the OSI model definition of Layer 2.
- 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 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 Profile: SCHC offers variations in the way it is operated, with a number of parameters listed in Appendix D. A Profile indicates a particular setting of all these parameters. Both ends of a SCHC communication must be provisioned with the same Profile information and with the same set of Rules before the communication starts, so that there is no ambiguity in how they expect to communicate.
- o Rule: A set of Field Descriptions.
- o Rule ID (Rule Identifier): 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 C/D: SCHC Compressor/Decompressor. A mechanism used on both sides, at the Dev and at the network, to achieve Compression/Decompression of headers.
- o SCHC F/R: SCHC Fragmentation / Reassembly. A mechanism used on both sides, at the Dev and at the network, to achieve Fragmentation / Reassembly of SCHC Packets.
- 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, from the Dev SCHC C/D to the network SCHC C/D.

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

5. SCHC overview

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

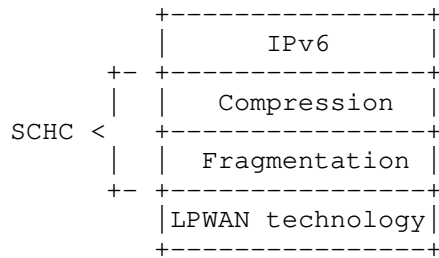
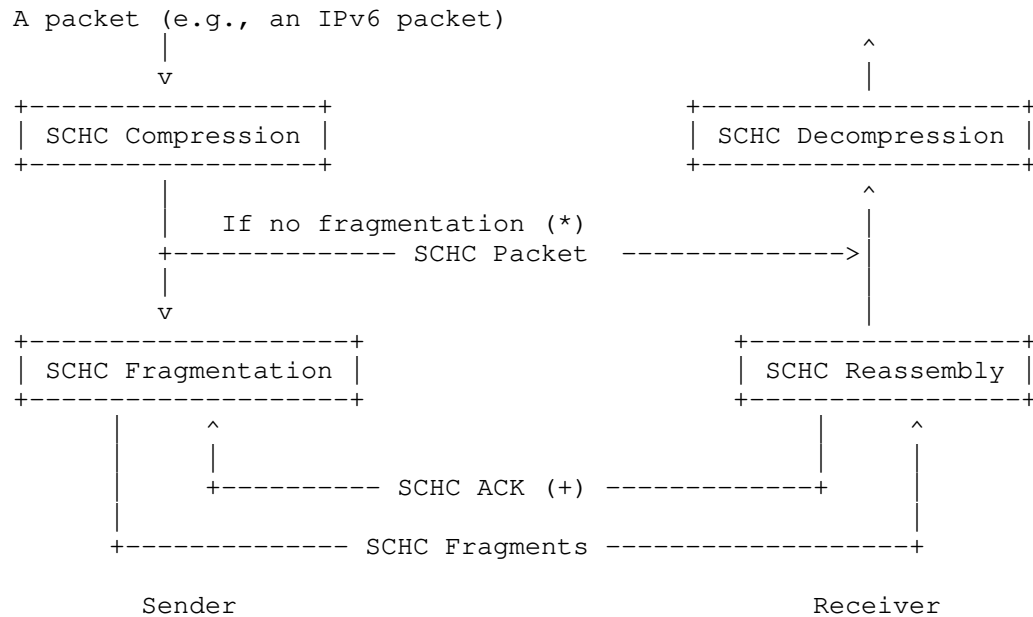


Figure 2: Protocol stack comprising IPv6, SCHC and an LPWAN technology

Before an upper layer packet (e.g., an IPv6 packet) is transmitted to the underlying layer, header compression is first attempted. The resulting packet is called a SCHC Packet, whether or not any compression is performed. If needed by the underlying layer, the optional SCHC Fragmentation MAY be applied to the SCHC Packet. The inverse operations take place at the receiver. This process is illustrated in Figure 3.



*: the decision to not use SCHC Fragmentation is left to each Profile.
 +: optional, depends on Fragmentation mode.

Figure 3: SCHC operations at the Sender and the Receiver

5.1. SCHC Packet format

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

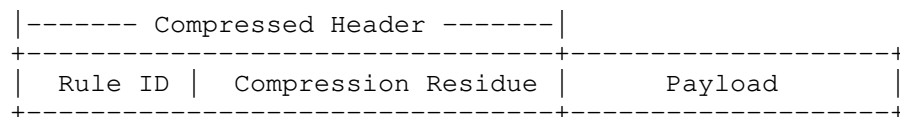


Figure 4: SCHC Packet

5.2. Functional mapping

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

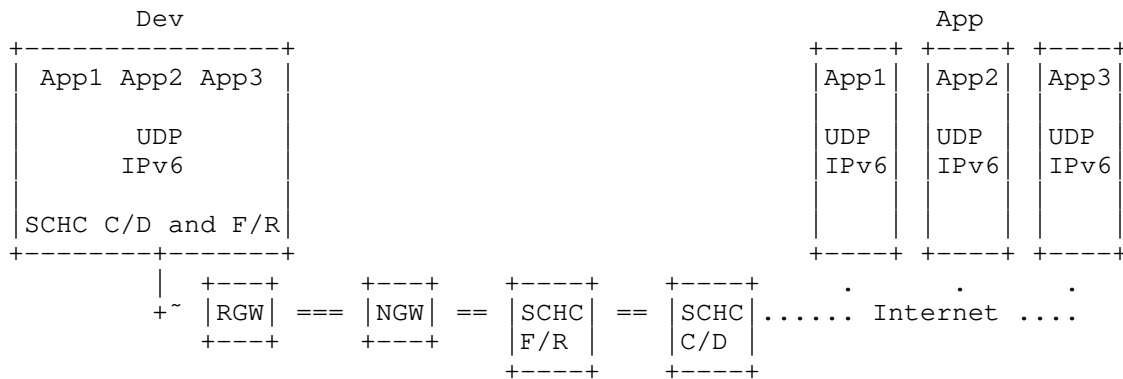


Figure 5: Architecture

SCHC C/D and SCHC F/R are located on both sides of the LPWAN transmission, hereafter called "the Dev side" and "the Network infrastructure side".

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

The SCHC F/R and C/D on the Network infrastructure side can be part of the NGW, or located in the Internet as long as a tunnel is established between them and the NGW. 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/Ds on both sides MUST share the same set of Rules. So MUST the SCHC F/Rs on both sides.

The operation in the Downlink direction is similar to that in the Uplink direction, only reversing the order in which the architecture elements are traversed.

6. Rule ID

Rule IDs identify the Rules used for Compression/Decompression or for Fragmentation/Reassembly.

The scope of the Rule ID of a Compression/Decompression Rule is the link between the SCHC C/D in a given Dev and the corresponding SCHC C/D in the Network infrastructure side. The scope of the Rule ID of a Fragmentation/Reassembly Rule is the link between the SCHC F/R in a given Dev and the corresponding SCHC F/R in the Network infrastructure side. If such a link is bidirectional, the scope includes both directions.

Inside their scopes, Rules for Compression/Decompression and Rules for Fragmentation/Reassembly share the same Rule ID space.

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

The Rule IDs are used:

- o For SCHC C/D, to identify the Rule (i.e., the set of Field Descriptions) that is used to compress a packet header.
 - * At least one Rule ID MUST be allocated to tagging packets for which SCHC compression was not possible (i.e., no matching compression Rule was found).
- o In SCHC F/R, to identify the specific mode and settings of F/R for one direction of traffic (Up or Dw).
 - * When F/R is used for both communication directions, at least two Rule ID values are needed for F/R, one per direction of traffic. This is because F/R may entail control messages flowing in the reverse direction compared to data traffic.

7. Compression/Decompression

Compression with SCHC is based on using a set of Rules, called the Context, to compress or decompress headers. SCHC avoids Context synchronization traffic, which consumes considerable bandwidth in other header compression mechanisms such as RoHC [RFC5795]. Since

the content of packets is highly predictable in LPWAN networks, static Contexts can be stored beforehand. The Contexts MUST be stored at both ends, and they can be learned by a provisioning protocol or by out of band means, or they can be pre-provisioned. The way the Contexts are provisioned is out of the scope of this document.

7.1. SCHC C/D Rules

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

The Context is a set of Rules. See Figure 6 for a high level, abstract representation of the Context. The formal specification of the representation of the Rules is outside the scope of this document.

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

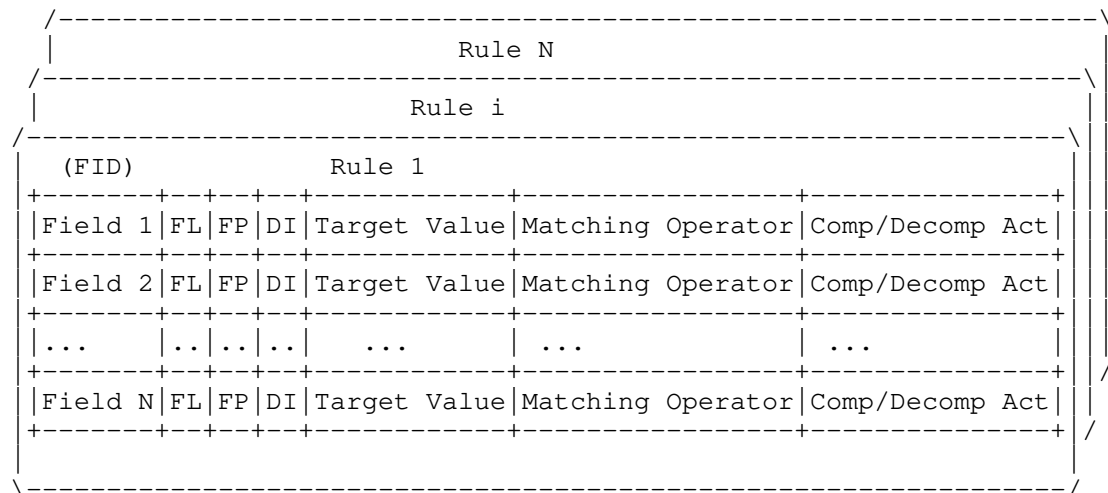


Figure 6: A Compression/Decompression Context

A Rule does not describe how the compressor parses a packet header to find and identify each field (e.g., the IPv6 Source Address, the UDP Destination Port or a CoAP URI path option). It is assumed that there is a protocol parser alongside SCHC that is able to identify all the fields encountered in the headers to be compressed, and to label them with a Field ID. Rules only describe the compression/decompression behavior for each header field, after it has been identified.

In a Rule, the Field Descriptions are listed in the order in which the fields appear in the packet header. The Field Descriptions describe the header fields with the following entries:

- o Field ID (FID) designates a protocol and field (e.g., UDP Destination Port), unambiguously among all protocols that a SCHC compressor processes. In the presence of protocol nesting, the Field ID also identifies the nesting.
- o Field Length (FL) represents the length of the original 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 by its own protocol specification (e.g., IPv6 or UDP). If the length is variable, the type defines the process to compute the length and its unit (bits, bytes...).
- o Field Position (FP): most often, a field only occurs once in a packet header. However, some fields may occur multiple times. An example is the uri-path of CoAP. FP indicates which occurrence this Field Description applies to. If FP is not specified in the Field Description, it takes the default value of 1. The value 1 designates the first occurrence. The value 0 is special. It means "don't care", see Section 7.3.
- 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 traveling both Up and Dw.
- o Target Value (TV) is the value used to match against the packet header field. The Target Value can be a scalar value of any type

(integer, strings, etc.) or a more complex structure (array, list, etc.). The types and representations are out of scope for this document.

- 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 might use 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 the Context related to one Dev. Hence, multiple Dev instances, which refer to different header compression Contexts, MAY reuse the same Rule ID for different Rules. On the Network infrastructure side, in order to identify the correct Rule to be applied, the SCHC Decompressor needs to associate the Rule ID with the Dev identifier. Similarly, the SCHC Compressor on the Network infrastructure side first identifies the destination Dev before looking for the appropriate compression Rule (and associated Rule ID) in the Context of that Dev.

7.3. Packet processing

The compression/decompression process follows several phases:

- o Compression Rule selection: the general idea is to browse the Rule set to find a Rule that has a matching Field Descriptor (given the DI and FP) for all and only those header fields that appear in the packet being compressed. The detailed algorithm is the following:
 - * The first step is to check the Field Identifiers (FID). If any header field of the packet being examined cannot be matched with a Field Description with the correct FID, the Rule MUST be disregarded. If any Field Description in the Rule has a FID that cannot be matched to one of the header fields of the packet being examined, the Rule MUST be disregarded.

- * The next step is to match the Field Descriptions by their direction, using the Direction Indicator (DI). If any field of the packet header cannot be matched with a Field Description with the correct FID and DI, the Rule MUST be disregarded.
- * Then the Field Descriptions are further selected according to Field Position (FP). If any field of the packet header cannot be matched with a Field Description with the correct FID, DI and FP, the Rule MUST be disregarded.

The value 0 for FP means "don't care", i.e. the comparison of this Field Description's FP with the position of the field of the packet header being compressed returns True, whatever that position. FP=0 can be useful to build compression Rules for protocols headers in which some fields order is irrelevant. An example could be uri-queries in CoAP. Care needs to be exercised when writing Rules containing FP=0 values. Indeed, it may result in decompressed packets having fields ordered differently compared to the original packet.

- * Once each header field has been associated with a Field Description with matching FID, DI and FP, 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 every field in the packet header satisfies the corresponding matching operators (MO) of a Rule (i.e. all MO results are True), that Rule is valid for use to compress the header. Otherwise, the Rule MUST be disregarded.

This specification does not prevent multiple Rules from matching the above steps and therefore being valid for use. Which Rule to use among multiple valid Rules is left to the implementation. As long as the same Rule set is installed at both ends, this degree of freedom does not constitute an interoperability issue.

- * If no valid compression Rule is found, then the packet MUST be sent uncompressed using the Rule ID dedicated to this purpose (see Section 6). The entire packet header is the Compression Residue (see Figure 4). Sending an uncompressed header is likely to require SCHC F/R.
- o Compression: if a valid Rule was found, each field of the header is compressed according to the Compression/Decompression Actions (CDAs) of the Rule. The fields are compressed in the order that the Field Descriptions appear in the Rule. The compression of each field results in a residue, which may be empty. The Compression Residue for the packet header is the concatenation of

the non-empty residues for each field of the header, in the order the Field Descriptions appear in the Rule. The order in which the Field Descriptions appear in the Rule is therefore semantically important.

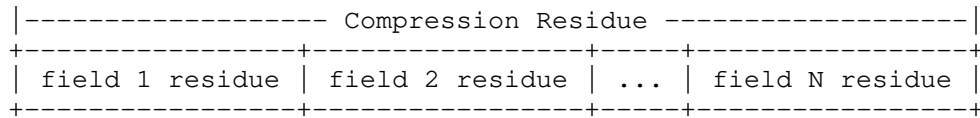


Figure 7: Compression Residue structure

- o **Sending:** The Rule ID is sent to the other end followed by the Compression Residue (which could be empty) or the uncompressed header, and directly followed by the payload (see Figure 4). The way the Rule ID is sent will be specified in the Profile and is out of the scope of the present document. For example, it could be included in an L2 header or sent as part of the L2 payload.
- o **Decompression:** when decompressing, on the Network infrastructure side the SCHC C/D needs to find the correct Rule based on the L2 address of the Dev; in this way, it can use the DevIID and the Rule ID. On the Dev side, only the Rule ID is needed to identify the correct Rule since the Dev typically only holds Rules that apply to itself.

This Rule describes the compressed header format. From this, the decompressor determines the order of the residues, the fixed-sized or variable-sized nature of each residue (see Section 7.5.2), and the size of the fixed-sized residues.

From the received compressed header, it can therefore retrieve all the residue values and associate them to the corresponding header fields.

For each field in the header, the receiver applies the CDA action associated to that field in order to reconstruct the original header field value. The CDA application order can be different from the order in which the fields are listed in the Rule. In particular, Compute-* MUST be applied after the application of the CDAs of all the fields it computes on.

7.4. Matching operators

Matching Operators (MOs) are functions used by both SCHC C/D endpoints. They are not typed and can be applied to integer, string

or any other data type. The result of the operation can either be True or False. MOs are defined as follows:

- o equal: The match result is True if the field value in the packet matches the TV.
- o ignore: No matching is attempted between the field value in the packet and the TV in the Rule. The result is always true.
- o MSB(x): A match is obtained if the most significant (leftmost) 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 x parameter must be a multiple of the FL unit. For example, x must be multiple of 8 if the unit of the variable length is bytes.
- o match-mapping: With match-mapping, the Target Value is a list of values. Each value of the list is identified by an 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 header fields and the inverse action taken by the decompressor to restore the original value.

Action	Compression	Decompression
not-sent	elided	use TV stored in Rule
value-sent	send	use received value
mapping-sent	send index	retrieve value from TV list
LSB	send LSB	concat. TV and received value
compute-*	elided	recompute at decompressor
DevIID	elided	build IID from L2 Dev addr
AppIID	elided	build IID from L2 App addr

Table 1: Compression and Decompression Actions

Table 1 summarizes the basic actions that can be used to compress and decompress a field. The first column shows the action's name. The second and third columns show the compression and decompression behaviors for each action.

7.5.1. processing fixed-length fields

If the field is identified in the Field Description as being of fixed length, then applying the CDA to compress this field results in a fixed amount of bits. The residue for that field is simply the bits resulting from applying the CDA to the field. This value may be empty (e.g., not-sent CDA), in which case the field residue is absent from the Compression Residue.

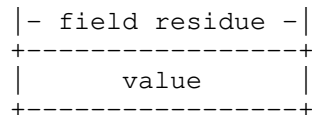


Figure 8: fixed sized field residue structure

7.5.2. processing variable-length fields

If the field is identified in the Field Description as being of variable length, then applying the CDA to compress this field may result in a value of fixed size (e.g., not-sent or mapping-sent) or of variable size (e.g., value-sent or LSB). In the latter case, the residue for that field is the bits that result from applying the CDA to the field, preceded with the size of the value. The most significant bit of the size is stored to the left (leftmost bit of the residue field).

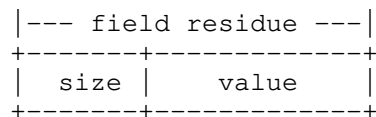


Figure 9: variable sized field residue structure

The size (using the unit defined in the FL) is encoded on 4, 12 or 28 bits as follows:

- o If the size is between 0 and 14, it is encoded as a 4 bits unsigned integer.
- o Sizes between 15 and 254 are encoded as 0b1111 followed by the 8 bits unsigned integer.
- o Larger sizes are encoded as 0xffff followed by the 16 bits unsigned integer.

If the field is identified in the Field Description as being of variable length and this field is not present in the packet header being compressed, size 0 MUST be sent to denote its absence.

7.5.3. not-sent CDA

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

The compressor does not send any 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.

7.5.4. value-sent CDA

The value-sent action can be used when the field value is not known by both the Compressor and the Decompressor. The field is sent in its entirety, using the same bit order as in the original packet header.

If this action is performed on a variable length field, the size of the residue value (using the units defined in FL) MUST be sent as described in Section 7.5.2.

This action is generally used with the "ignore" MO.

7.5.5. mapping-sent CDA

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

On the compressor side, the match-mapping Matching Operator searches the TV for a match with the header field value. The mapping-sent CDA then sends the corresponding index as the field residue. The most significant bit of the index is stored to the left (leftmost bit of the residue field).

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.

The first element in the list MUST be represented by index value 0, and successive elements in the list MUST have indices incremented by 1.

7.5.6. 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 compressor sends the Least Significant Bits as the field residue value. The number of bits sent is the original header field length minus the length specified in the MSB(x) MO. The bits appear in the residue in the same bit order as in the original packet header.

The decompressor concatenates the x most significant bits of Target Value and the received residue value.

If this action is performed on a variable length field, the size of the residue value (using the units defined in FL) MUST be sent as described in Section 7.5.2.

7.5.7. DevIID, AppIID CDA

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

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

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

7.5.8. Compute-*

Some fields can be elided at the compressor and recomputed locally at the decompressor.

Because the field is uniquely identified by its Field ID (e.g., UDP length), the relevant protocol specification unambiguously defines the algorithm for such computation.

Examples of fields that know how to recompute themselves are UDP length, IPv6 length and UDP checksum.

8. Fragmentation/Reassembly

8.1. Overview

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

The optional SCHC Fragmentation/Reassembly (SCHC F/R) functionality enables such LPWAN technologies to comply with the IPv6 MTU requirement of 1280 bytes [RFC8200]. It is OPTIONAL to implement per this specification, but Profiles may specify that it is REQUIRED.

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

The same SCHC F/R mode MUST be used for all SCHC Fragments of a given SCHC Packet. This document does not specify which mode(s) must be implemented and used over a specific LPWAN technology. That information will be given in Profiles.

SCHC allows transmitting non-fragmented SCHC Packet concurrently with fragmented SCHC Packets. In addition, SCHC F/R provides protocol elements that allow transmitting several fragmented SCHC Packets concurrently, i.e. interleaving the transmission of fragments from different fragmented SCHC Packets. A Profile MAY restrict the latter behavior.

The L2 Word size (see Section 4) determines the encoding of some messages. SCHC F/R usually generates SCHC Fragments and SCHC ACKs that are multiples of L2 Words.

8.2. SCHC F/R Protocol Elements

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

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

8.2.1. Messages

SCHC F/R defines the following messages:

- o SCHC Fragment: A message that carries part of a SCHC Packet from the sender to the receiver.
- o SCHC ACK: An acknowledgement for fragmentation, by the receiver to the sender. This message is used to indicate whether or not the reception of pieces of, or the whole of the fragmented SCHC Packet, was successful.
- o SCHC ACK REQ: A request by the sender for a SCHC ACK from the receiver.
- o SCHC Sender-Abort: A message by the sender telling the receiver that it has aborted the transmission of a fragmented SCHC Packet.
- o SCHC Receiver-Abort: A message by the receiver to tell the sender to abort the transmission of a fragmented SCHC Packet.

The format of these messages is provided in Section 8.3.

8.2.2. Tiles, Windows, Bitmaps, Timers, Counters

8.2.2.1. Tiles

The SCHC Packet is fragmented into pieces, hereafter called tiles. The tiles MUST be non-empty and pairwise disjoint. Their union MUST be equal to the SCHC Packet.

See Figure 10 for an example.

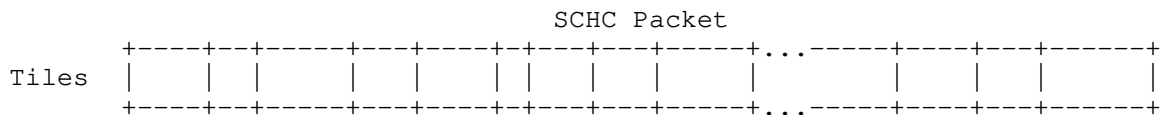


Figure 10: a SCHC Packet fragmented in tiles

Modes (see Section 8.4) MAY place additional constraints on tile sizes.

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

8.2.2.2. Windows

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

If windows are used

- o all the windows of a SCHC Packet, except the last one, MUST contain the same number of tiles. This number is WINDOW_SIZE.
- o WINDOW_SIZE MUST be specified in a Profile.
- o the windows are numbered.
- o their numbers MUST increment by 1 from 0 upward, from the start of the SCHC Packet to its end.
- o the last window MUST contain WINDOW_SIZE tiles or less.
- o tiles are numbered within each window.
- o the tile indices MUST decrement by 1 from WINDOW_SIZE - 1 downward, looking from the start of the SCHC Packet toward its end.
- o each tile of a SCHC Packet is therefore uniquely identified by a window number and a tile index within this window.

See Figure 11 for an example.

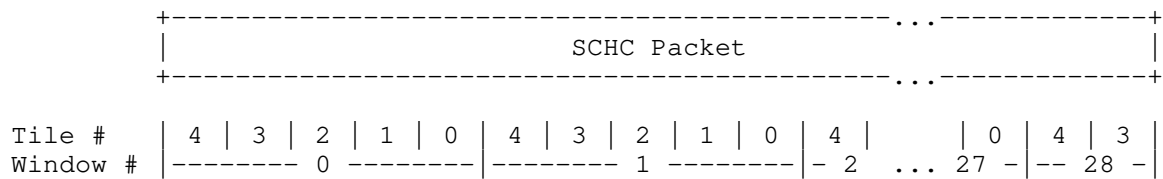


Figure 11: a SCHC Packet fragmented in tiles grouped in 29 windows, with WINDOW_SIZE = 5

Appendix E discusses the benefits of selecting one among multiple window sizes depending on the size of the SCHC Packet to be fragmented.

When windows are used

- o Bitmaps (see Section 8.2.2.3) MAY be sent back by the receiver to the sender in a SCHC ACK message.
- o A Bitmap corresponds to exactly one Window.

8.2.2.3. Bitmaps

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

At the receiver

- o a bit set to 1 in the Bitmap indicates that a tile associated with that bit position has been correctly received for that window.
- o a bit set to 0 in the Bitmap indicates that there has been no tile correctly received, associated with that bit position, for that window. Possible reasons include that the tile was not sent at all, not received, or received with errors.

8.2.2.4. Timers and counters

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

- o Inactivity Timer: a SCHC Fragment receiver uses this timer to abort waiting for a SCHC F/R message.
- o Retransmission Timer: a SCHC Fragment sender uses this timer to abort waiting for an expected SCHC ACK.
- o Attempts: this counter counts the requests for SCHC ACKs, up to MAX_ACK_REQUESTS.

8.2.3. Integrity Checking

The integrity of the fragmentation-reassembly process of a SCHC Packet MUST be checked at the receive end. A Profile MUST specify how integrity checking is performed.

It is RECOMMENDED that integrity checking be performed by computing a Reassembly Check Sequence (RCS) based on the SCHC Packet at the sender side and transmitting it to the receiver for comparison with the RCS locally computed after reassembly.

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

The CRC32 polynomial 0xEDB88320 (i.e., the reversed polynomial representation, which is used in the Ethernet standard [ETHERNET]) is RECOMMENDED as the default algorithm for computing the RCS.

The RCS MUST be computed on the full SCHC Packet concatenated with the padding bits, if any, of the SCHC Fragment carrying the last tile. The rationale is that the SCHC reassembler has no way of knowing the boundary between the last tile and the padding bits. Indeed, this requires decompressing the SCHC Packet, which is out of the scope of the SCHC reassembler.

The concatenation of the complete SCHC Packet and any padding bits, if present, of the last SCHC Fragment does not generally constitute an integer number of bytes. CRC libraries are usually byte-oriented. It is RECOMMENDED that the concatenation of the complete SCHC Packet and any last fragment padding bits be zero-extended to the next byte boundary and that the RCS be computed on that byte array.

8.2.4. Header Fields

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

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

The Rule tells apart a non-fragmented SCHC Packet from SCHC Fragments. It will also tell apart SCHC Fragments of fragmented SCHC Packets that use different SCHC F/R modes or different parameters. Interleaved transmission of these is therefore possible.

All SCHC F/R messages pertaining to the same SCHC Packet MUST bear the same Rule ID.

- o Datagram Tag (DTag). This field allows differentiating SCHC F/R messages belonging to different SCHC Packets that may be using the same Rule ID simultaneously. Hence, it allows interleaving fragments of a new SCHC Packet with fragments of a previous SCHC Packet under the same Rule ID.

The size of the DTag field (called T, in bits) is defined by each Profile for each Rule ID. When T is 0, the DTag field does not appear in the SCHC F/R messages and the DTag value is defined as 0.

When T is 0, there can be no more than one fragmented SCHC Packet in transit for each fragmentation Rule ID.

If T is not 0, DTag

- * MUST be set to the same value for all the SCHC F/R messages related to the same fragmented SCHC Packet,
 - * MUST be set to different values for SCHC F/R messages related to different SCHC Packets that are being fragmented under the same Rule ID, and whose transmission may overlap.
- o W: The W field is optional. It is only present if windows are used. Its presence and size (called M, in bits) is defined by each SCHC F/R mode and each Profile for each Rule ID.
- This field carries information pertaining to the window a SCHC F/R message relates to. If present, W MUST carry the same value for all the SCHC F/R messages related to the same window. Depending on the mode and Profile, W may carry the full window number, or just the least significant bit or any other partial representation of the window number.
- o Fragment Compressed Number (FCN). The FCN field is present in the SCHC Fragment Header. Its size (called N, in bits) is defined by each Profile for each Rule ID.

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

- * The FCN value with all the bits equal to 1 (called All-1) signals that the very last tile of a SCHC Packet has been

transmitted. By extension, if windows are used, the last window of a packet is called the All-1 window.

- * If windows are used, the FCN value with all the bits equal to 0 (called All-0) signals the last tile of a window that is not the last one of the SCHC packet. By extension, such a window is called an All-0 window.
- o Reassembly Check Sequence (RCS). This field only appears in the All-1 SCHC Fragments. Its size (called U, in bits) is defined by each Profile for each Rule ID.

See Section 8.2.3 for the RCS default size, default polynomial and details on RCS computation.

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

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

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

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

8.3. SCHC F/R Message Formats

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

8.3.1. SCHC Fragment format

A SCHC Fragment conforms to the general format shown in Figure 12. It comprises a SCHC Fragment Header and a SCHC Fragment Payload. The SCHC Fragment Payload carries one or several tile(s).

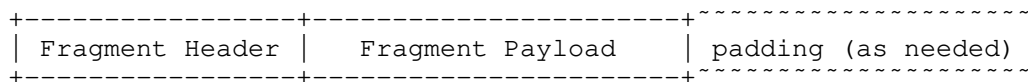


Figure 12: SCHC Fragment general format

8.3.1.1. Regular SCHC Fragment

The Regular SCHC Fragment format is shown in Figure 13. Regular SCHC Fragments are generally used to carry tiles that are not the last one of a SCHC Packet. The DTag field and the W field are OPTIONAL, their presence is specified by each mode and Profile.

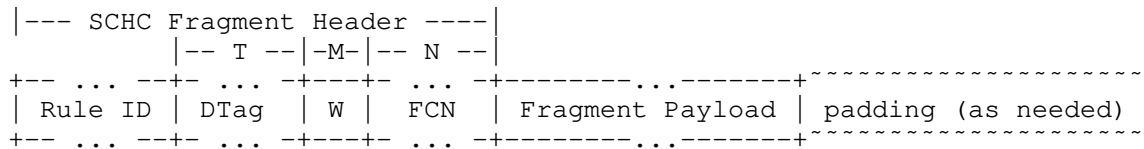


Figure 13: Detailed Header Format for Regular SCHC Fragments

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

Profiles MUST ensure that a SCHC Fragment with FCN equal to 0 (called an All-0 SCHC Fragment) is distinguishable by size, even in the presence of padding, from a SCHC ACK REQ message (see Section 8.3.3) with the same Rule ID value and with the same T, M and N values. This condition is met if the Payload is at least the size of an L2 Word. This condition is also met if the SCHC Fragment Header is a multiple of L2 Words.

8.3.1.2. All-1 SCHC Fragment

The All-1 SCHC Fragment format is shown in Figure 14. The sender uses the All-1 SCHC Fragment format for the message that completes the emission of a fragmented SCHC Packet. The DTag field, the W field, the RCS field and the Payload are OPTIONAL, their presence is specified by each mode and Profile. At least one of RCS field or Payload MUST be present. The FCN field is all ones.

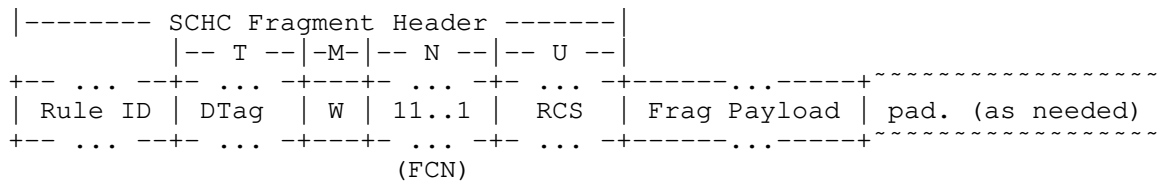


Figure 14: Detailed Header Format for the All-1 SCHC Fragment

Profiles MUST ensure that an All-1 SCHC Fragment message is distinguishable by size, even in the presence of padding, from a SCHC Sender-Abort message (see Section 8.3.4) with the same Rule ID value and with the same T, M and N values. This condition is met if the RCS is present and is at least the size of an L2 Word, or if the

Payload is present and at least the size an L2 Word. This condition is also met if the SCHC Sender-Absort Header is a multiple of L2 Words.

8.3.2. SCHC ACK format

The SCHC ACK message is shown in Figure 15. The DTag field and the W field are OPTIONAL, their presence is specified by each mode and Profile. The Compressed Bitmap field MUST be present in SCHC F/R modes that use windows, and MUST NOT be present in other modes.

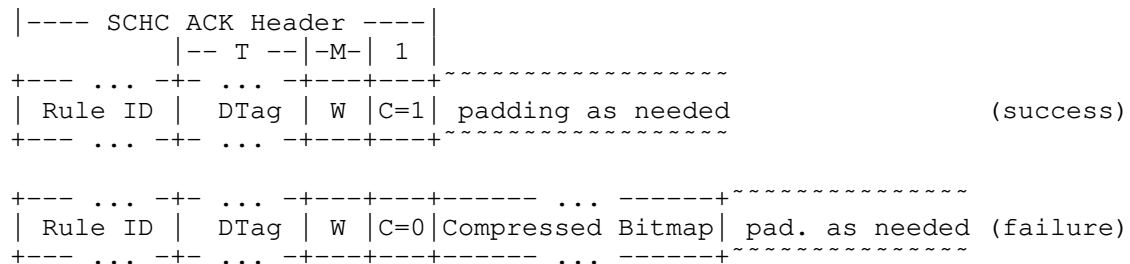


Figure 15: Format of the SCHC ACK message

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

If the C bit is set to 1 (integrity check successful), no Bitmap is carried.

If the C bit is set to 0 (integrity check not performed or failed) and if windows are used, a Compressed Bitmap for the window referred to by the W field is transmitted as specified in Section 8.3.2.1.

8.3.2.1. Bitmap Compression

For transmission, the Compressed Bitmap in the SCHC ACK message is defined by the following algorithm (see Figure 16 for a follow-along example):

- o Build a temporary SCHC ACK message that contains the Header followed by the original Bitmap (see Section 8.2.2.3 for a description of Bitmaps).
- o Position scissors at the end of the Bitmap, after its last bit.
- o While the bit on the left of the scissors is 1 and belongs to the Bitmap, keep moving left, then stop. When this is done,

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

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

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

Figure 16 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 -----|
|  -- T --  -M-  1  |
+--- ... +- ... +-----+-----+
| Rule ID | DTag | W | C=0 | 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |
+--- ... +- ... +-----+-----+
                        next L2 Word boundary ->|

```

Figure 16: SCHC ACK Header plus uncompressed Bitmap

Figure 17 shows that the last 14 bits are not sent.

```

|---- SCHC ACK Header ----|CpBmp|
|  -- T --  -M-  1  |
+--- ... +- ... +-----+-----+
| Rule ID | DTag | W | C=0 | 1 0 1 |
+--- ... +- ... +-----+-----+
                        next L2 Word boundary ->|

```

Figure 17: Resulting SCHC ACK message with Compressed Bitmap

Figure 18 shows an example of a SCHC ACK with tile indices ranging from 6 down to 0, where the Bitmap indicates that the second and the fourth tile of the window have not been correctly received.

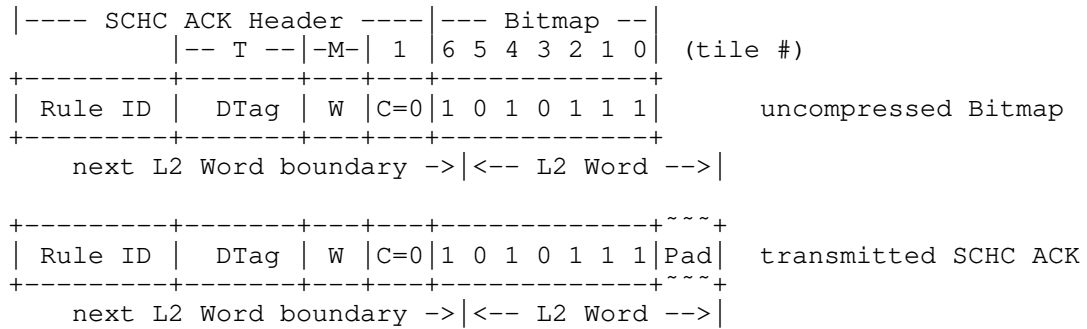


Figure 18: Example of a SCHC ACK message, missing tiles

Figure 19 shows an example of a SCHC ACK with FCN ranging from 6 down to 0, where integrity check has not been performed or has failed and the Bitmap indicates that there is no missing tile in that window.

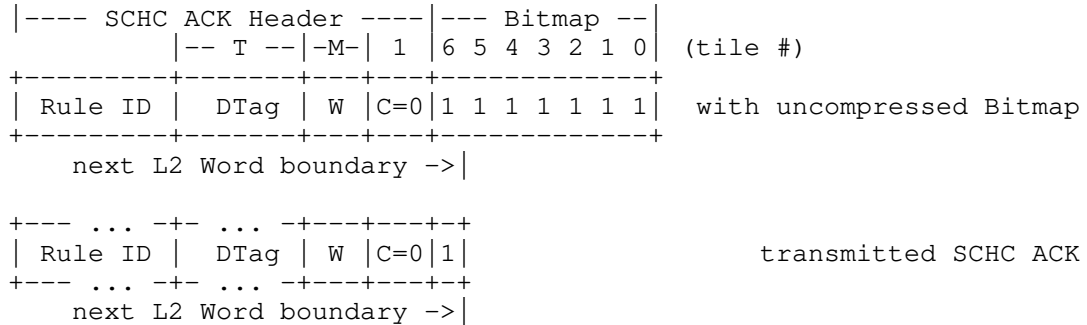


Figure 19: Example of a SCHC ACK message, no missing tile

8.3.3. SCHC ACK REQ format

The SCHC ACK REQ is used by a sender to request a SCHC ACK from the receiver. Its format is shown in Figure 20. The DTag field and the W field are OPTIONAL, their presence is specified by each mode and Profile. The FCN field is all zero.

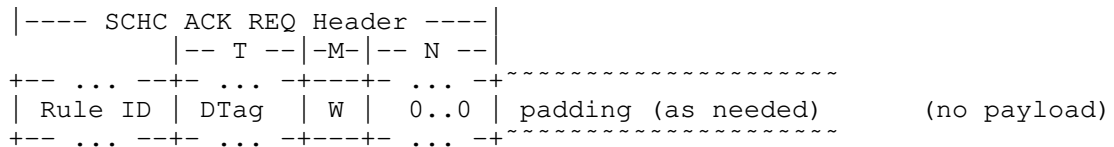


Figure 20: SCHC ACK REQ format

8.3.4. SCHC Sender-Abort format

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

The SCHC Sender-Abort format is shown in Figure 21. The DTag field and the W field are OPTIONAL, their presence is specified by each mode and Profile. The FCN field is all ones.

```
|---- Sender-Abort Header ----|
|      |-- T --|-M-|-N --|
+--- ... ---+ ... +-----+ ... +-----+
| Rule ID | DTag | W | 11..1 | padding (as needed)
+--- ... ---+ ... +-----+ ... +-----+
```

Figure 21: SCHC Sender-Abort format

If the W field is present,

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

The SCHC Sender-Abort MUST NOT be acknowledged.

8.3.5. SCHC Receiver-Abort format

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

The SCHC Receiver-Abort format is shown in Figure 22. The DTag field and the W field are OPTIONAL, their presence is specified by each mode and Profile.

```
|--- Receiver-Abort Header ---|
|      |--- T ---|-M-| 1 |
+--- ... ---+ ... +-----+-----+-----+-----+-----+
| Rule ID | DTag | W | C=1 | 1..1 | 1..1 |
+--- ... ---+ ... +-----+-----+-----+-----+-----+
| next L2 Word boundary -> | <-- L2 Word --> |
```

Figure 22: SCHC Receiver-Abort format

If the W field is present,

- o the fragment receiver MUST set it to all ones. Other values are RESERVED.
- o if the value is different from all ones, the fragment sender MUST ignore the message.

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

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

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

The SCHC Receiver-Abort MUST NOT be acknowledged.

8.4. SCHC F/R modes

This specification includes several SCHC F/R modes, which

- o allow for a range of reliability options, such as optional SCHC Fragment retransmission
- o support various LPWAN characteristics, such as links with variable MTU or unidirectional links.

More modes may be defined in the future.

Appendix B provides examples of fragmentation sessions based on the modes described hereafter.

Appendix C provides examples of Finite State Machines implementing the SCHC F/R modes described hereafter.

8.4.1. No-ACK mode

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

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

without expecting any acknowledgement. Therefore, No-ACK does not require bidirectional links: unidirectional links are just fine.

In No-ACK mode, only the All-1 SCHC Fragment is padded as needed. The other SCHC Fragments are intrinsically aligned to L2 Words.

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

Each Profile MUST specify which Rule ID value(s) correspond to SCHC F/R messages operating in this mode.

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

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

Each Profile, for each Rule ID value, MUST define

- o the size of the DTag field,
- o the size and algorithm for the RCS field,
- o the expiration time of the Inactivity Timer

Each Profile, for each Rule ID value, MAY define

- o a value of N different from the recommended one,
- o the meaning of values sent in the FCN field, for values different from the All-1 value.

For each active pair of Rule ID and DTag values, the receiver MUST maintain an Inactivity Timer. If the receiver is under-resourced to do this, it MUST silently drop the related messages.

8.4.1.1. Sender behavior

At the beginning of the fragmentation of a new SCHC Packet, the fragment sender MUST select a Rule ID and DTag value pair for this SCHC Packet.

Each SCHC Fragment MUST contain exactly one tile in its Payload. The tile MUST be at least the size of an L2 Word. The sender MUST transmit the SCHC Fragments messages in the order that the tiles appear in the SCHC Packet. Except for the last tile of a SCHC

Packet, each tile MUST be of a size that complements the SCHC Fragment Header so that the SCHC Fragment is a multiple of L2 Words without the need for padding bits. Except for the last one, the SCHC Fragments MUST use the Regular SCHC Fragment format specified in Section 8.3.1.1. The SCHC Fragment that carries the last tile MUST be an All-1 SCHC Fragment, described in Section 8.3.1.2.

The sender MAY transmit a SCHC Sender-Abort.

Figure 37 shows an example of a corresponding state machine.

8.4.1.2. Receiver behavior

Upon receiving each Regular SCHC Fragment,

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

On receiving an All-1 SCHC Fragment,

- o the receiver MUST append the All-1 SCHC Fragment Payload and the padding bits to the previously received SCHC Fragment Payloads for this SCHC Packet
- o the receiver MUST perform the integrity check
- o if integrity checking fails, the receiver MUST drop the reassembled SCHC Packet
- o the reassembly operation concludes.

On expiration of the Inactivity Timer, the receiver MUST drop the SCHC Packet being reassembled.

On receiving a SCHC Sender-Abort, the receiver MAY drop the SCHC Packet being reassembled.

Figure 38 shows an example of a corresponding state machine.

8.4.2. ACK-Always mode

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

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

- o The L2 MTU value does not change while the fragments of a SCHC Packet are being transmitted.
- o There is a feedback path from the reassembler to the fragmenter. See Appendix F for a discussion on using ACK-Always mode on quasi-bidirectional links.

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

The tiles are not required to be of uniform size. In ACK-Always mode, only the All-1 SCHC Fragment is padded as needed. The other SCHC Fragments are intrinsically aligned to L2 Words.

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

Each Profile MUST specify which Rule ID value(s) correspond to SCHC F/R messages operating in this mode.

The W field MUST be present and its size M MUST be 1 bit.

Each Profile, for each Rule ID value, MUST define

- o the value of N (size of the FCN field),
- o the value of WINDOW_SIZE, which MUST be strictly less than 2^N ,
- o the size and algorithm for the RCS field,
- o the size of the DTag field,
- o the value of MAX_ACK_REQUESTS,
- o the expiration time of the Retransmission Timer
- o the expiration time of the Inactivity Timer

For each active pair of Rule ID and DTag values, the sender MUST maintain

- o one Attempts counter
- o one Retransmission Timer

For each active pair of Rule ID and DTag values, the receiver MUST maintain

- o one Inactivity Timer
- o one Attempts counter

8.4.2.1. Sender behavior

At the beginning of the fragmentation of a new SCHC Packet, the fragment sender MUST select a Rule ID and DTag value pair for this SCHC Packet.

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

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

For a SCHC Fragment that carries a tile other than the last one of the SCHC Packet,

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

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

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

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

The sender starts by a "blind transmission" phase, in which it MUST transmit all the tiles composing the window, in decreasing tile index order.

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

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

At any time,

- o on receiving a SCHC Receiver-Abort, the fragment sender MAY exit with an error condition.
- o on receiving a SCHC ACK that bears a W value different from the W value that it currently uses, the fragment sender MUST silently discard and ignore that SCHC ACK.

Figure 39 shows an example of a corresponding state machine.

8.4.2.2. Receiver behavior

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

- o the receiver SHOULD check if the DTag value has not recently been used for that Rule ID value, thereby ensuring that the received SCHC Fragment is not a remnant of a prior fragmented SCHC Packet transmission. The initial value of the Inactivity Timer is the RECOMMENDED lifetime for the DTag value at the receiver. If the SCHC Fragment is determined to be such a remnant, the receiver MAY silently ignore it and discard it.
- o the receiver MUST start a process to assemble a new SCHC Packet with that Rule ID and DTag value pair.
- o the receiver MUST start an Inactivity Timer for that RuleID and DTag pair. It MUST initialize an Attempts counter to 0 for that RuleID and DTag pair. It MUST initialize a window counter to 0. If the receiver is under-resourced to do this, it MUST respond to the sender with a SCHC Receiver Abort.

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

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

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

The receiver MUST first initialize an empty Bitmap for the first window, then enter an "acceptance phase", in which

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

- o on receiving a SCHC ACK REQ with the W bit equal to the local W bit, the receiver MUST send a SCHC ACK for this window.
- o on receiving a SCHC Fragment with the W bit equal to the local W bit, the receiver MUST assemble the received tile based on the window counter and on the FCN field in the SCHC Fragment and it MUST update the Bitmap.
 - * if the SCHC Fragment received is an All-0 SCHC Fragment, the current window is determined to be a not-last window, the receiver MUST send a SCHC ACK for this window and it MUST enter the "retransmission phase" for this window.
 - * if the SCHC Fragment received is an All-1 SCHC Fragment, the padding bits of the All-1 SCHC Fragment MUST be assembled after the received tile, the current window is determined to be the last window, the receiver MUST perform the integrity check and it MUST send a SCHC ACK for this window. Then,
 - + If the integrity check indicates that the full SCHC Packet has been correctly reassembled, the receiver MUST enter the "clean-up phase" for this window.
 - + If the integrity check indicates that the full SCHC Packet has not been correctly reassembled, the receiver enters the "retransmission phase" for this window.

In the "retransmission phase":

- o if the window is a not-last window
 - * on receiving a SCHC Fragment that is not All-0 or All-1 and that has a W bit different from the local W bit, the receiver MUST increment its window counter and allocate a fresh Bitmap, it MUST assemble the tile received and update the Bitmap and it MUST enter the "acceptance phase" for that new window.
 - * on receiving a SCHC ACK REQ with a W bit different from the local W bit, the receiver MUST increment its window counter and allocate a fresh Bitmap, it MUST send a SCHC ACK for that new window and it MUST enter the "acceptance phase" for that new window.
 - * on receiving a SCHC All-0 Fragment with a W bit different from the local W bit, the receiver MUST increment its window counter and allocate a fresh Bitmap, it MUST assemble the tile received and update the Bitmap, it MUST send a SCHC ACK for that new

window and it MUST stay in the "retransmission phase" for that new window.

- * on receiving a SCHC All-1 Fragment with a W bit different from the local W bit, the receiver MUST increment its window counter and allocate a fresh Bitmap, it MUST assemble the tile received, including the padding bits, it MUST update the Bitmap and perform the integrity check, it MUST send a SCHC ACK for the new window, which is determined to be the last window. Then,
 - + If the integrity check indicates that the full SCHC Packet has been correctly reassembled, the receiver MUST enter the "clean-up phase" for that new window.
 - + If the integrity check indicates that the full SCHC Packet has not been correctly reassembled, the receiver enters the "retransmission phase" for that new window.
- * on receiving a SCHC Fragment with a W bit equal to the local W bit,
 - + if the SCHC Fragment received is an All-1 SCHC Fragment, the receiver MUST silently ignore it and discard it.
 - + otherwise, the receiver MUST assemble the tile received and update the Bitmap. If the Bitmap becomes fully populated with 1's or if the SCHC Fragment is an All-0, the receiver MUST send a SCHC ACK for this window.
- * on receiving a SCHC ACK REQ with the W bit equal to the local W bit, the receiver MUST send a SCHC ACK for this window.
- o if the window is the last window
 - * on receiving a SCHC Fragment or a SCHC ACK REQ, either one having a W bit different from the local W bit, the receiver MUST silently ignore and discard that message.
 - * on receiving a SCHC ACK REQ with the W bit equal to the local W bit, the receiver MUST send a SCHC ACK for this window.
 - * on receiving a SCHC Fragment with a W bit equal to the local W bit,
 - + if the SCHC Fragment received is an All-0 SCHC Fragment, the receiver MUST silently ignore it and discard it.

- + otherwise, the receiver MUST update the Bitmap and it MUST assemble the tile received. If the SCHC Fragment received is an All-1 SCHC Fragment, the receiver MUST assemble the padding bits of the All-1 SCHC Fragment after the received tile, it MUST perform the integrity check and
 - if the integrity check indicates that the full SCHC Packet has been correctly reassembled, the receiver MUST send a SCHC ACK and it enters the "clean-up phase".
 - if the integrity check indicates that the full SCHC Packet has not been correctly reassembled,
 - o if the SCHC Fragment received was an All-1 SCHC Fragment, the receiver MUST send a SCHC ACK for this window.

In the "clean-up phase":

- o On receiving an All-1 SCHC Fragment or a SCHC ACK REQ, either one having the W bit equal to the local W bit, the receiver MUST send a SCHC ACK.
- o Any other SCHC Fragment received MUST be silently ignored and discarded.

At any time, on sending a SCHC ACK, the receiver MUST increment the Attempts counter.

At any time, on incrementing its window counter, the receiver MUST reset the Attempts counter.

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

Figure 40 shows an example of a corresponding state machine.

8.4.3. ACK-on-Error mode

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

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

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

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

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

See Figure 23 for an example.

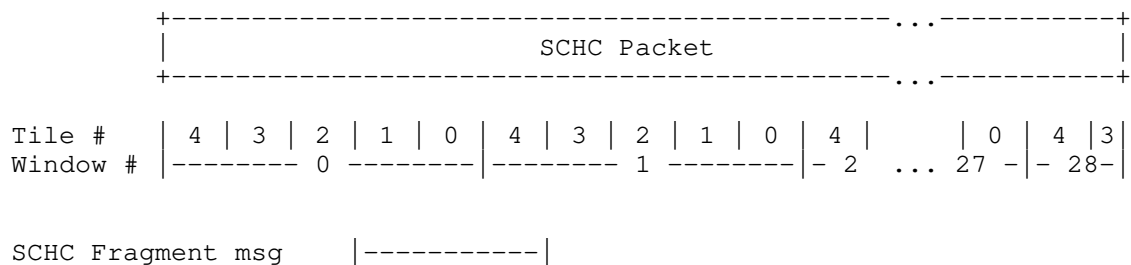


Figure 23: a SCHC Packet fragmented in tiles, ACK-on-Error mode

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

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

- o integrity checking shows that the fragmented SCHC Packet has been correctly reassembled at the receive end, and this information has been conveyed back to the sender,
- o or too many retransmission attempts were made,

- o or the receiver determines that the transmission of this fragmented SCHC Packet has been inactive for too long.

Each Profile MUST specify which Rule ID value(s) correspond to SCHC F/R messages operating in this mode.

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

Each Profile, for each Rule ID value, MUST define

- o the tile size (a tile does not need to be multiple of an L2 Word, but it MUST be at least the size of an L2 Word)
- o the value of M (size of the W field),
- o the value of N (size of the FCN field),
- o the value of WINDOW_SIZE, which MUST be strictly less than 2^N ,
- o the size and algorithm for the RCS field,
- o the size of the DTag field,
- o the value of MAX_ACK_REQUESTS,
- o the expiration time of the Retransmission Timer
- o the expiration time of the Inactivity Timer
- o whether the last tile is carried in a Regular SCHC Fragment or an All-1 SCHC Fragment (see Section 8.4.3.1)
- o if the penultimate tile MAY be one L2 Word smaller than the regular tile size. In this case, the regular tile size MUST be at least twice the L2 Word size.

For each active pair of Rule ID and DTag values, the sender MUST maintain

- o one Attempts counter
- o one Retransmission Timer

For each active pair of Rule ID and DTag values, the receiver MUST maintain

- o one Inactivity Timer

- o one Attempts counter

8.4.3.1. Sender behavior

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

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

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

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

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

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

Depending on the Profile, the last tile of a SCHC Packet MUST be sent either

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

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

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

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

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

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

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

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

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

On Retransmission Timer expiration

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

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

On receiving a SCHC ACK,

- o if the W field in the SCHC ACK corresponds to the last window of the SCHC Packet,
 - * if the C bit is set, the sender MAY exit successfully
 - * otherwise,
 - + if the Profile mandates that the last tile be sent in an All-1 SCHC Fragment,
 - if the SCHC ACK shows no missing tile at the receiver, the sender
 - o MUST send a SCHC Sender-Abort
 - o MAY exit with an error condition
 - otherwise

- o the fragment sender MUST send SCHC Fragment messages containing all the tiles that are reported missing in the SCHC ACK.
- o if the last of these SCHC Fragment messages is not an All-1 SCHC Fragment, then the fragment sender MUST in addition send after it a SCHC ACK REQ with the W field corresponding to the last window.
- + otherwise,
 - if the SCHC ACK shows no missing tile at the receiver, the sender MUST send the All-1 SCHC Fragment
 - otherwise
 - o the fragment sender MUST send SCHC Fragment messages containing all the tiles that are reported missing in the SCHC ACK.
 - o the fragment sender MUST then send either the All-1 SCHC Fragment or a SCHC ACK REQ with the W field corresponding to the last window.
- o otherwise, the fragment sender
 - * MUST send SCHC Fragment messages containing the tiles that are reported missing in the SCHC ACK
 - * then it MAY send a SCHC ACK REQ with the W field corresponding to the last window

See Figure 41 for one among several possible examples of a Finite State Machine implementing a sender behavior obeying this specification.

8.4.3.2. Receiver behavior

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

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

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

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

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

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

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

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

- o if the receiver knows of any windows with missing tiles for the packet being reassembled, it MUST return a SCHC ACK for the lowest-numbered such window,
- o otherwise,
 - * if it has received at least one tile, it MUST return a SCHC ACK for the highest-numbered window it currently has tiles for
 - * otherwise it MUST return a SCHC ACK for window numbered 0

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

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

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

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

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

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

Reassembly of the SCHC Packet concludes when

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

See Figure 42 for one among several possible examples of a Finite State Machine implementing a receiver behavior obeying this specification, and that is meant to match the sender Finite State Machine of Figure 41.

9. Padding management

SCHC C/D and SCHC F/R operate on bits, not bytes. SCHC itself does not have any alignment prerequisite. The size of SCHC Packets can be any number of bits.

If the layer below SCHC constrains the payload to align to some boundary, called L2 Words (for example, bytes), the SCHC messages MUST be padded. When padding occurs, the number of appended bits MUST be strictly less than the L2 Word size.

If a SCHC Packet is sent unfragmented (see Figure 24), it is padded as needed for transmission.

If a SCHC Packet needs to be fragmented for transmission, it is not padded in itself. Only the SCHC F/R messages are padded as needed for transmission. Some SCHC F/R messages are intrinsically aligned to L2 Words.

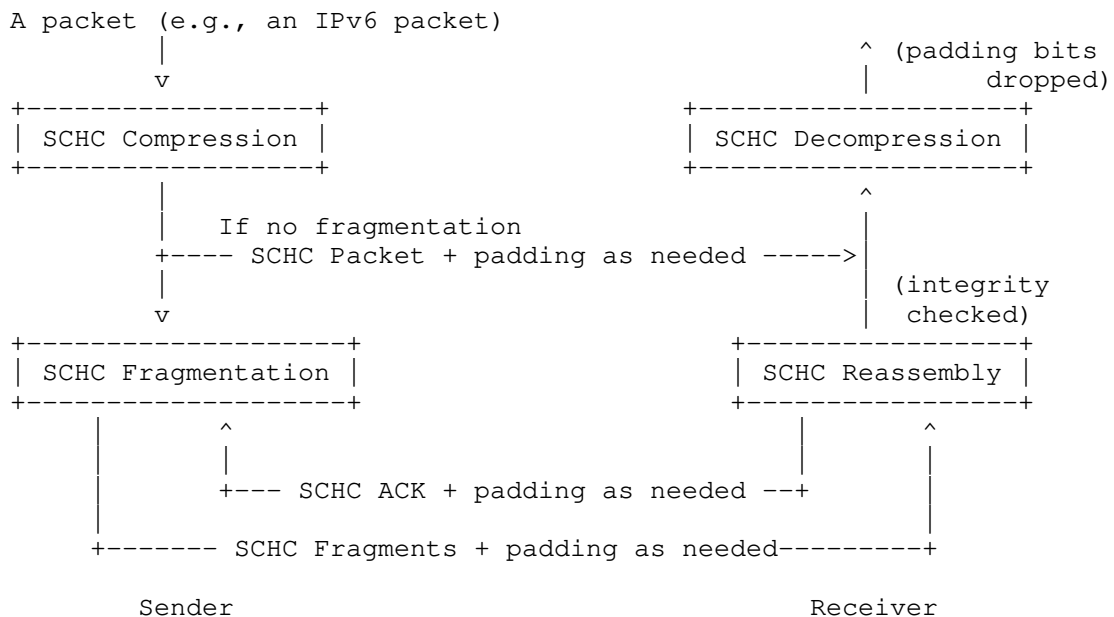


Figure 24: SCHC operations, including padding as needed

Each Profile MUST specify the size of the L2 Word. The L2 Word might actually be a single bit, in which case no padding will take place at all.

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

10. SCHC Compression for IPv6 and UDP headers

This section lists the IPv6 and UDP header fields and describes how they can be compressed. An example of a set of Rules for UDP/IPv6 header compression is provided in Appendix A.

10.1. IPv6 version field

The IPv6 version field is labeled by the protocol parser as being the "version" field of the IPv6 protocol. Therefore, it only exists for IPv6 packets. In the Rule, TV is set to 6, MO to "ignore" and CDA to "not-sent".

10.2. IPv6 Traffic class field

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

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

- 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.

ECN functionality depends on both bits of the ECN field, which are the 2 LSBs of this field, hence sending only a single LSB of this field is NOT RECOMMENDED.

10.3. Flow label field

If the flow label is not set, i.e. its value is zero, the Field Descriptor in the Rule SHOULD contain a TV set to zero, an "equal" MO and a "not-sent" CDA.

If the flow label is set to a pseudo-random value according to [RFC6437], in the Rule, TV is not set to any particular value, MO is set to "ignore" and CDA is set to "value-sent".

If the flow label is set according to some prior agreement, i.e. by a flow state establishment method as allowed by [RFC6437], the Field Descriptor in the Rule SHOULD contain a TV with this agreed-upon value, an "equal" MO and a "not-sent" CDA.

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-*".

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 (Up) and downlink (Dw). In Up, since there is no IP forwarding between the Dev and the SCHC C/D, the value is relatively constant. On the other hand, the Dw value depends on Internet routing and can change more frequently. The Direction Indicator (DI) can be used to distinguish both directions:

- o in the Up, 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 Dw, the Hop Limit is elided for transmission and forced to 1 at the receiver, by setting TV to 1, MO to "ignore" and CDA to "not-sent". This prevents any further forwarding.

10.7. IPv6 addresses fields

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

10.7.1. IPv6 source and destination prefixes

Both ends MUST be configured with the appropriate prefixes. For a specific flow, the source and destination prefixes can be unique and stored in the Context. 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 26.

Otherwise, the TV is not set, the MO is set to "ignore" 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". On LPWAN technologies where the frames carry a single identifier (corresponding to the Dev.), AppIID cannot be used.

As described in [RFC8065], it may be undesirable to build the Dev IPv6 IID out of the Dev address. Another static value is used instead. In that case, the TV contains the static value, the MO operator is set to "equal" and the CDA is set to "not-sent".

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 its entirety 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 extension headers

This document does not provide recommendations on how to compress IPv6 extension headers.

10.9. UDP source and destination ports

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

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

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

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

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

10.10. UDP length field

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

10.11. UDP Checksum field

The UDP checksum operation is mandatory with IPv6 for most packets but there are exceptions [RFC8200].

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] requires any node implementing zero-checksum mode to follow the requirements specified in "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums" [RFC6936].

6LoWPAN Header Compression [RFC6282] also specifies that a UDP checksum can be elided by the compressor and re-computed by the decompressor when an upper layer guarantees the integrity of the UDP payload and pseudo-header. A specific example of this is when a message integrity check protects the compressed message between the compressor that elides the UDP checksum and the decompressor that

computes it, with a strength that is identical or better to the UDP checksum.

Similarly, a SCHC compressor MAY elide the UDP checksum when another layer guarantees at least equal 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-*".

In particular, when SCHC fragmentation is used, a fragmentation RCS 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, and if the SCHC Packet is fragmented even when it would fit unfragmented in the L2 MTU, then the compressor MAY verify and then elide the UDP checksum. Whether and when the UDP Checksum is elided is to be specified in the Profile.

Since the compression happens before the fragmentation, implementers 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

As explained in Section 5, SCHC is expected to be implemented on top of LPWAN technologies, which are expected to implement security measures.

In this section, we analyze the potential security threats that could be introduced into an LPWAN by adding the SCHC functionalities.

12.1. Security considerations for SCHC Compression/Decompression

12.1.1. Forged SCHC Packet

Let's assume that an attacker is able to send a forged SCHC Packet to a SCHC Decompressor.

Let's first consider the case where the Rule ID contained in that forged SCHC Packet does not correspond to a Rule allocated in the

Rule table. An implementation should detect that the Rule ID is invalid and should silently drop the offending SCHC Packet.

Let's now consider that the Rule ID corresponds to a Rule in the table. With the CDAs defined in this document, the reconstructed packet is at most a constant number of bits bigger than the SCHC Packet that was received. This assumes that the compute-* decompression actions produce a bounded number of bits, irrespective of the incoming SCHC Packet. This property is true for IPv6 Length, UDP Length and UDP Checksum, for which the compute-* CDA is recommended by this document.

As a consequence, SCHC Decompression does not amplify attacks, beyond adding a bounded number of bits to the SCHC Packet received. This bound is determined by the Rule stored in the receiving device.

As a general safety measure, a SCHC Decompressor should never re-construct a packet larger than MAX_PACKET_SIZE (defined in a Profile, with 1500 bytes as generic default).

12.1.2. Compressed packet size as a side channel to guess a secret token

Some packet compression methods are known to be susceptible to attacks, such as BREACH and CRIME. The attack involves injecting arbitrary data into the packet and observing the resulting compressed packet size. The observed size potentially reflects correlation between the arbitrary data and some content that was meant to remain secret, such as a security token, thereby allowing the attacker to get at the secret.

By contrast, SCHC Compression takes place header field by header field, with the SCHC Packet being a mere concatenation of the compression residues of each of the individual field. Any correlation between header fields does not result in a change in the SCHC Packet size compressed under the same Rule.

If SCHC C/D is used to compress packets that include a secret information field, such as a token, the Rule set should be designed so that the size of the compression residue for the field to remain secret is the same irrespective of the value of the secret information. This is achieved by e.g., sending this field in extenso with the "ignore" MO and the "value-sent" CDA. This recommendation is disputable if it is ascertained that the Rule set itself will remain secret.

12.1.1.3. Decompressed packet different from the original packet

As explained in Section 7.3, using FPs with value 0 in Field Descriptors in a Rule may result in header fields appearing in the decompressed packet in an order different from that in the original packet. Likewise, as stated in Section 7.5.3, using an "ignore" MO together with a "not-sent" CDA will result in the header field taking the TV value, which is likely to be different from the original value.

Depending on the protocol, the order of header fields in the packet may be functionally significant or not.

Furthermore, if the packet is protected by a checksum or a similar integrity protection mechanism, and if the checksum is transmitted instead of being recomputed as part of the decompression, these situations may result in the packet being considered corrupt and dropped.

12.2. Security considerations for SCHC Fragmentation/Reassembly

12.2.1. Buffer reservation attack

Let's assume that an attacker is able to send a forged SCHC Fragment to a SCHC Reassembler.

A node can perform a buffer reservation attack: the receiver will reserve buffer space for the SCHC Packet. If the implementation has only one buffer, 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. An implementation may have multiple reassembly buffers. The cost to mount this attack is linear with the number of buffers at the target node. Better, the cost for an attacker can be increased if individual fragments of multiple SCHC Packets can be stored in the reassembly buffer. The finer grained the reassembly buffer (down to the smallest tile size), the higher the cost of the attack. If buffer overload does occur, a smart receiver could selectively discard SCHC Packets being reassembled based on the sender behavior, which may help identify which SCHC Fragments have been sent by the attacker. Another mild counter-measure is for the target to abort the fragmentation/reassembly session as early as it detects a non-identical SCHC Fragment duplicate, anticipating for an eventual corrupt SCHC Packet, so as to save the sender the hassle of sending the rest of the fragments for this SCHC Packet.

12.2.2. Corrupt Fragment attack

Let's assume that an attacker is able to send a forged SCHC Fragment to a SCHC Reassembler. The malicious node is additionally assumed to be able to hear an incoming communication destined to the target node.

It can then send a forged SCHC Fragment that looks like it belongs to a SCHC Packet already being reassembled at the target node. This can cause the SCHC Packet to be considered corrupt and be dropped by the receiver. The amplification happens here by a single spoofed SCHC Fragment rendering a full sequence of legit SCHC Fragments useless. If the target uses ACK-Always or ACK-on-Error mode, such a malicious node can also interfere with the acknowledgement and repetition algorithm of SCHC F/R. A single spoofed ACK, with all bitmap bits set to 0, will trigger the repetition of WINDOW_SIZE tiles. This protocol loop amplification depletes the energy source of the target node and consumes the channel bandwidth. Similarly, a spoofed ACK REQ will trigger the sending of a SCHC ACK, which may be much larger than the ACK REQ if WINDOW_SIZE is large. These consequences should be borne in mind when defining profiles for SCHC over specific LPWAN technologies.

12.2.3. Fragmentation as a way to bypass Network Inspection

Fragmentation is known for potentially allowing to force through a Network Inspection device (e.g., firewall) packets that would be rejected if unfragmented. This involves sending overlapping fragments to rewrite fields whose initial value led the Network Inspection device to allow the flow go through.

SCHC F/R is expected to be used over one LPWAN link, where no Network Inspection device is expected to sit. As described in Section 5.2, even if the SCHC F/R on the Network infrastructure side is located in the Internet, a tunnel is to be established between it and the NGW.

12.2.4. Privacy issues associated with SCHC header fields

SCHC F/R allocates a DTag value to fragments belonging to the same SCHC Packet. Concerns were raised that, if DTag is a wide counter that is incremented in a predictable fashion for each new fragmented SCHC Packet, it might lead to a privacy issue, such as enabling tracking of a device across LPWANs.

However, SCHC F/R is expected to be used over exactly one LPWAN link. As described in Section 5.2, even if the SCHC F/R on the Network infrastructure side is located in the Internet, a tunnel is to be established between it and the NGW. Therefore, assuming the tunnel

provides confidentiality, neither the DTag field nor any other SCHC-introduced field is visible over the Internet.

13. Acknowledgements

Thanks to (in alphabetical order) Sergio Aguilar Romero, David Black, Carsten Bormann, Deborah Brungard, Brian Carpenter, Philippe Clavier, Alissa Cooper, Roman Danyliw, Daniel Ducuara Beltran, Diego Dujovne, Eduardo Ingles Sanchez, Rahul Jadhav, Benjamin Kaduk, Arunprabhu Kandasamy, Suresh Krishnan, Mirja Kuehlewind, Barry Leiba, Sergio Lopez Bernal, Antoni Markovski, Alexey Melnikov, Georgios Papadopoulos, Alexander Pelov, Charles Perkins, Edgar Ramos, Alvaro Retana, Adam Roach, Shoichi Sakane, Joseph Salowey, Pascal Thubert, and Eric Vyncke for useful design considerations, reviews and comments.

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.

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>>.
- [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>>.
- [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>>.
- [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>>.

14.2. Informative References

- [ETHERNET] "IEEE Standard for Ethernet", IEEE standard, DOI 10.1109/ieeestd.2018.8457469, n.d..
- [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>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<https://www.rfc-editor.org/info/rfc6437>>.
- [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>>.
- [RFC8065] Thaler, D., "Privacy Considerations for IPv6 Adaptation-Layer Mechanisms", RFC 8065, DOI 10.17487/RFC8065, February 2017, <<https://www.rfc-editor.org/info/rfc8065>>.

Appendix A. Compression Examples

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

The mechanisms defined in this document can be applied to a 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 Dev (using ports 5683) and Global IPv6 Address prefixes alpha::IID/64 to beta::1/64. The last flow is for legacy applications using different ports numbers, the destination IPv6 address prefix is gamma::1/64.

Figure 25 presents the protocol stack. IPv6 and UDP are represented with dotted lines since these protocols are compressed on the radio link.

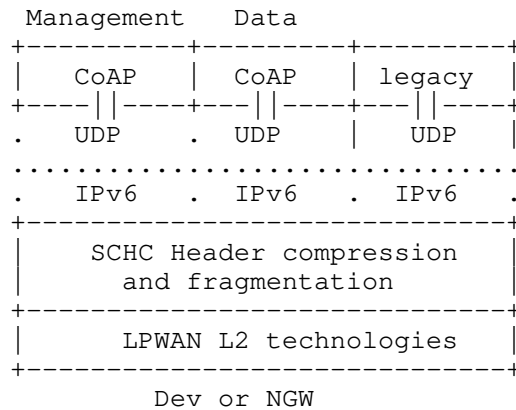


Figure 25: Simplified Protocol Stack for LP-WAN

In some LPWAN technologies, only the Devs have a device ID. 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

Special Rule ID used to tag an uncompressed UDP/IPV6 packet.

Rule 1

Field	FL	FP	DI	Value	Match Opera.	Comp Decomp Action	Sent [bits]
IPv6 Version	4	1	Bi	6	ignore	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	compute-*	
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	compute-*		
UDP checksum	16	1	Bi		ignore	compute-*		

Rule 2

Field	FL	FP	DI	Value	Match Opera.	Action Action	Sent [bits]
IPv6 Version	4	1	Bi	6	ignore	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	compute-*	
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	compute-*	
UDP checksum	16	1	Bi		ignore	compute-*	

Rule 3

Field	FL	FP	DI	Value	Match Opera.	Action Action	Sent [bits]
IPv6 Version	4	1	Bi	6	ignore	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	compute-*	
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	compute-*		
UDP checksum	16	1	Bi		ignore	compute-*		

Figure 26: Context Rules

Figure 26 describes a example of a Rule set.

In this example, 0 was chosen as the special Rule ID that tags packets that cannot be compressed with any compression Rule.

All the fields described in Rules 1-3 are present in the IPv6 and UDP headers. The DevIID-DID value is found in the L2 header.

Rules 2-3 use global addresses. The way the Dev learns the prefix is not in the scope of the document.

Rule 3 compresses each port number to 4 bits.

Appendix B. Fragmentation Examples

This section provides examples for the various fragment reliability modes specified in this document. In the drawings, Bitmaps are shown in their uncompressed form.

Figure 27 illustrates the transmission in No-ACK mode of a SCHC Packet that needs 11 SCHC 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=1 + RCS ---->| Integrity check: success
(End)

```

Figure 27: No-ACK mode, 11 SCHC Fragments

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

Figure 28 illustrates the transmission in ACK-on-Error mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, WINDOW_SIZE=7 and no lost SCHC Fragment.

```

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 + RCS-->| Integrity check: success
<-- ACK, W=1, C=1 ---| C=1
(End)

```

Figure 28: ACK-on-Error mode, 11 tiles, one tile per SCHC Fragment, no lost SCHC Fragment.

Figure 29 illustrates the transmission in ACK-on-Error mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, WINDOW_SIZE=7 and three lost SCHC 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, C=0 ---	6543210
-----W=0, FCN=4----->	Bitmap:1101011
-----W=0, FCN=2----->	
(no ACK)	
-----W=1, FCN=6----->	
-----W=1, FCN=5----->	
-----W=1, FCN=4--X-->	
- W=1, FCN=7 + RCS ->	Integrity check: failure
<-- ACK, W=1, C=0 ---	C=0, Bitmap:1100001
-----W=1, FCN=4----->	Integrity check: success
<-- ACK, W=1, C=1 ---	C=1
(End)	

Figure 29: ACK-on-Error mode, 11 tiles, one tile per SCHC Fragment, lost SCHC Fragments.

Figure 30 shows an example of a transmission in ACK-on-Error mode of a SCHC Packet fragmented in 73 tiles, with N=5, WINDOW_SIZE=28, M=2 and 3 lost SCHC Fragments.

Sender	Receiver
-----W=0, FCN=27----->	4 tiles sent
-----W=0, FCN=23----->	4 tiles sent
-----W=0, FCN=19----->	4 tiles sent
-----W=0, FCN=15--X-->	4 tiles sent (not received)
-----W=0, FCN=11----->	4 tiles sent
-----W=0, FCN=7 ----->	4 tiles sent
-----W=0, FCN=3 ----->	4 tiles sent
-----W=1, FCN=27----->	4 tiles sent
-----W=1, FCN=23----->	4 tiles sent
-----W=1, FCN=19----->	4 tiles sent
-----W=1, FCN=15----->	4 tiles sent
-----W=1, FCN=11----->	4 tiles sent
-----W=1, FCN=7 ----->	4 tiles sent
-----W=1, FCN=3 --X-->	4 tiles sent (not received)
-----W=2, FCN=27----->	4 tiles sent
-----W=2, FCN=23----->	4 tiles sent
-----W=2, FCN=19----->	1 tile sent
-----W=2, FCN=18----->	1 tile sent
-----W=2, FCN=17----->	1 tile sent
-----W=2, FCN=16----->	1 tile sent
-----W=2, FCN=15----->	1 tile sent
-----W=2, FCN=14----->	1 tile sent
-----W=2, FCN=13--X-->	1 tile sent (not received)
-----W=2, FCN=12----->	1 tile sent
---W=2, FCN=31 + RCS->	Integrity check: failure
<---- ACK, W=0, C=0 ---	C=0, Bitmap:1111111111110000111111111111
-----W=0, FCN=15----->	1 tile sent
-----W=0, FCN=14----->	1 tile sent
-----W=0, FCN=13----->	1 tile sent
-----W=0, FCN=12----->	1 tile sent
<---- ACK, W=1, C=0 ---	C=0, Bitmap:11111111111111111111111111110000
-----W=1, FCN=3 ----->	1 tile sent
-----W=1, FCN=2 ----->	1 tile sent
-----W=1, FCN=1 ----->	1 tile sent
-----W=1, FCN=0 ----->	1 tile sent
<---- ACK, W=2, C=0 ---	C=0, Bitmap:11111111111111111010000000000001
-----W=2, FCN=13----->	Integrity check: success
<---- ACK, W=2, C=1 ---	C=1

(End)

Figure 30: ACK-on-Error mode, variable MTU.

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

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

Figure 31 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, with $N=3$, $WINDOW_SIZE=7$ 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, C=0 ---	Bitmap:1111111
-----W=1, FCN=6----->	
-----W=1, FCN=5----->	
-----W=1, FCN=4----->	
--W=1, FCN=7 + RCS-->	Integrity check: success
<-- ACK, W=1, C=1 ---	C=1
(End)	

Figure 31: ACK-Always mode, 11 tiles, one tile per SCHC Fragment, no loss.

Figure 32 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, $N=3$, $WINDOW_SIZE=7$ and three lost SCHC 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----->	6543210
<-- ACK, W=0, C=0 ---	Bitmap:1101011
-----W=0, FCN=4----->	
-----W=0, FCN=2----->	
<-- ACK, W=0, C=0 ---	Bitmap:1111111
-----W=1, FCN=6----->	
-----W=1, FCN=5----->	
-----W=1, FCN=4--X-->	
--W=1, FCN=7 + RCS-->	Integrity check: failure
<-- ACK, W=1, C=0 ---	C=0, Bitmap:1100001
-----W=1, FCN=4----->	Integrity check: success
<-- ACK, W=1, C=1 ---	C=1
(End)	

Figure 32: ACK-Always mode, 11 tiles, one tile per SCHC Fragment, three lost SCHC Fragments.

Figure 33 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with one tile per SCHC Fragment, N=3, WINDOW_SIZE=7, three lost SCHC Fragments and only one retry needed to recover each lost SCHC 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 + RCS-->	Integrity check: failure
<-- ACK, W=0, C=0 ---	C=0, Bitmap:1100001
-----W=0, FCN=4----->	Integrity check: failure
-----W=0, FCN=3----->	Integrity check: failure
-----W=0, FCN=2----->	Integrity check: success
<-- ACK, W=0, C=1 ---	C=1
(End)	

Figure 33: ACK-Always mode, 6 tiles, one tile per SCHC Fragment, three lost SCHC Fragments.

Figure 34 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with one tile per SCHC Fragment, N=3,

WINDOW_SIZE=7, three lost SCHC Fragments, and the second SCHC 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 + RCS-->	Integrity check: failure
	<-- ACK, W=0, C=0 ---	C=0, Bitmap:1100001
	-----W=0, FCN=4----->	Integrity check: failure
	-----W=0, FCN=3----->	Integrity check: failure
	-----W=0, FCN=2----->	Integrity check: success
	<-X-ACK, W=0, C=1 ---	C=1
timeout	---	
	--- W=0, ACK REQ --->	ACK REQ
	<-- ACK, W=0, C=1 ---	C=1
	(End)	

Figure 34: ACK-Always mode, 6 tiles, one tile per SCHC Fragment, SCHC ACK loss.

Figure 35 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with N=3, WINDOW_SIZE=7, with three lost SCHC Fragments, and one retransmitted SCHC 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 + RCS-->	Integrity check: failure
	<-- ACK, W=0, C=0 ---	C=0, Bitmap:1100001
	-----W=0, FCN=4----->	Integrity check: failure
	-----W=0, FCN=3----->	Integrity check: failure
	-----W=0, FCN=2--X-->	
timeout	---	
	--- W=0, ACK REQ --->	ACK REQ
	<-- ACK, W=0, C=0 ---	C=0, Bitmap: 1111101
	-----W=0, FCN=2----->	Integrity check: success
	<-- ACK, W=0, C=1 ---	C=1
	(End)	

Figure 35: ACK-Always mode, 6 tiles, retransmitted SCHC Fragment lost again.

Figure 36 illustrates the transmission in ACK-Always mode of a SCHC Packet fragmented in 28 tiles, with one tile per SCHC Fragment, N=5, WINDOW_SIZE=24 and two lost SCHC Fragments.

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, C=0 ---	Bitmap:11011111111111011111111111
-----W=0, FCN=21----->	
-----W=0, FCN=10----->	
<---- ACK, W=0, C=0 ---	Bitmap:11111111111111111111111111
-----W=1, FCN=23----->	
-----W=1, FCN=22----->	
-----W=1, FCN=21----->	
--W=1, FCN=31 + RCS-->	Integrity check: success
<---- ACK, W=1, C=1 ---	C=1

(End)

Figure 36: ACK-Always mode, 28 tiles, one tile per SCHC Fragment, lost SCHC Fragments.

Appendix C. Fragmentation State Machines

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

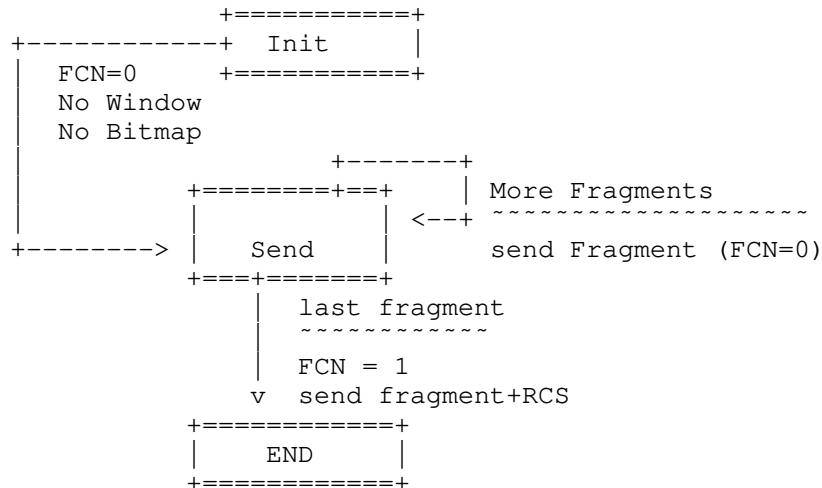


Figure 37: Sender State Machine for the No-ACK Mode

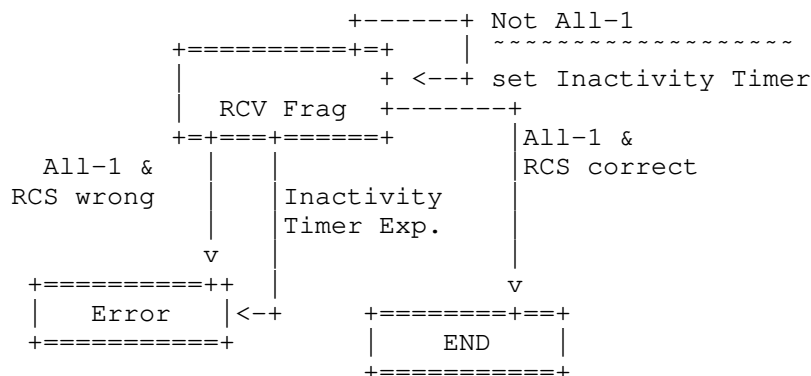


Figure 38: Receiver State Machine for the No-ACK Mode

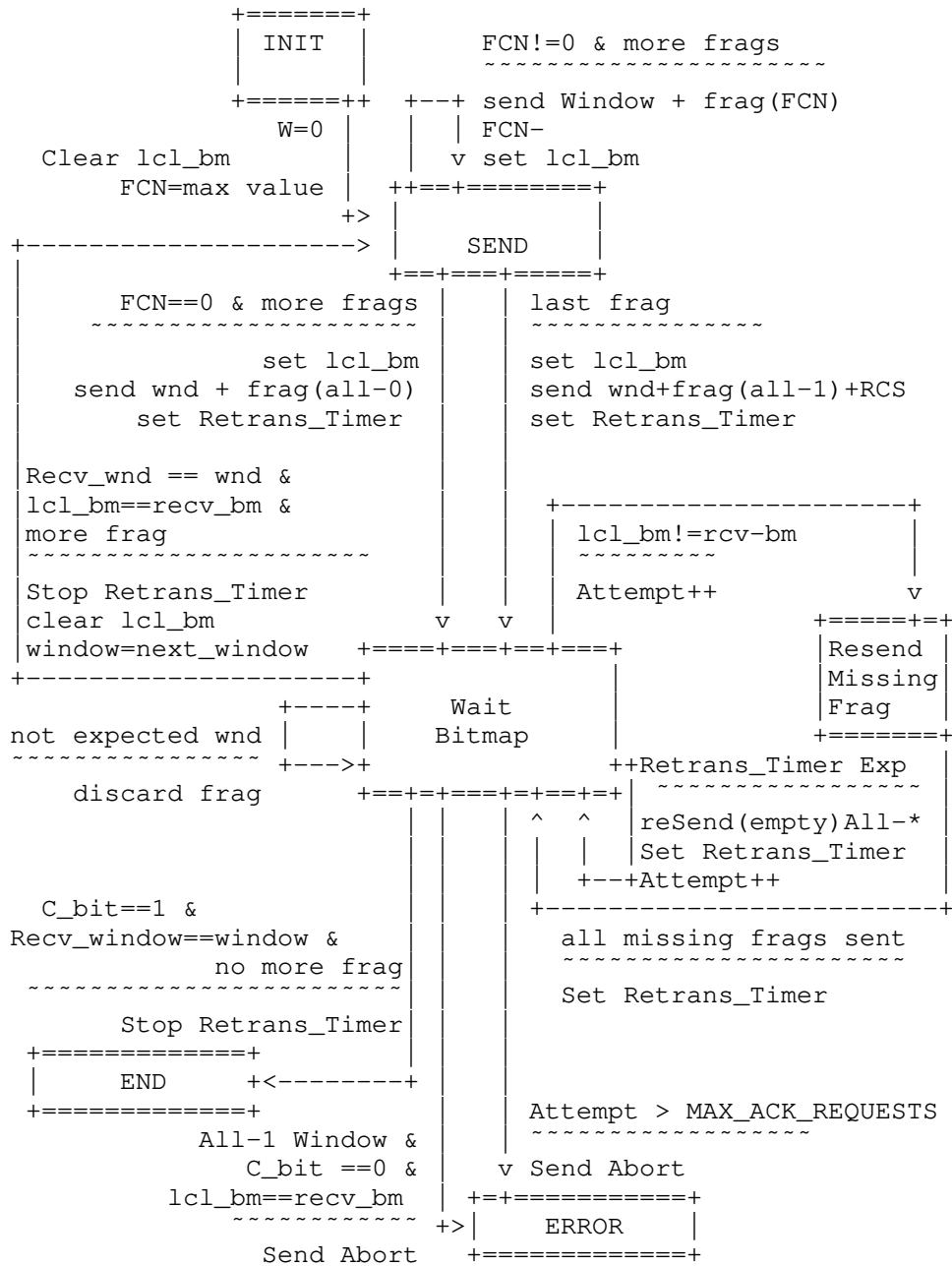
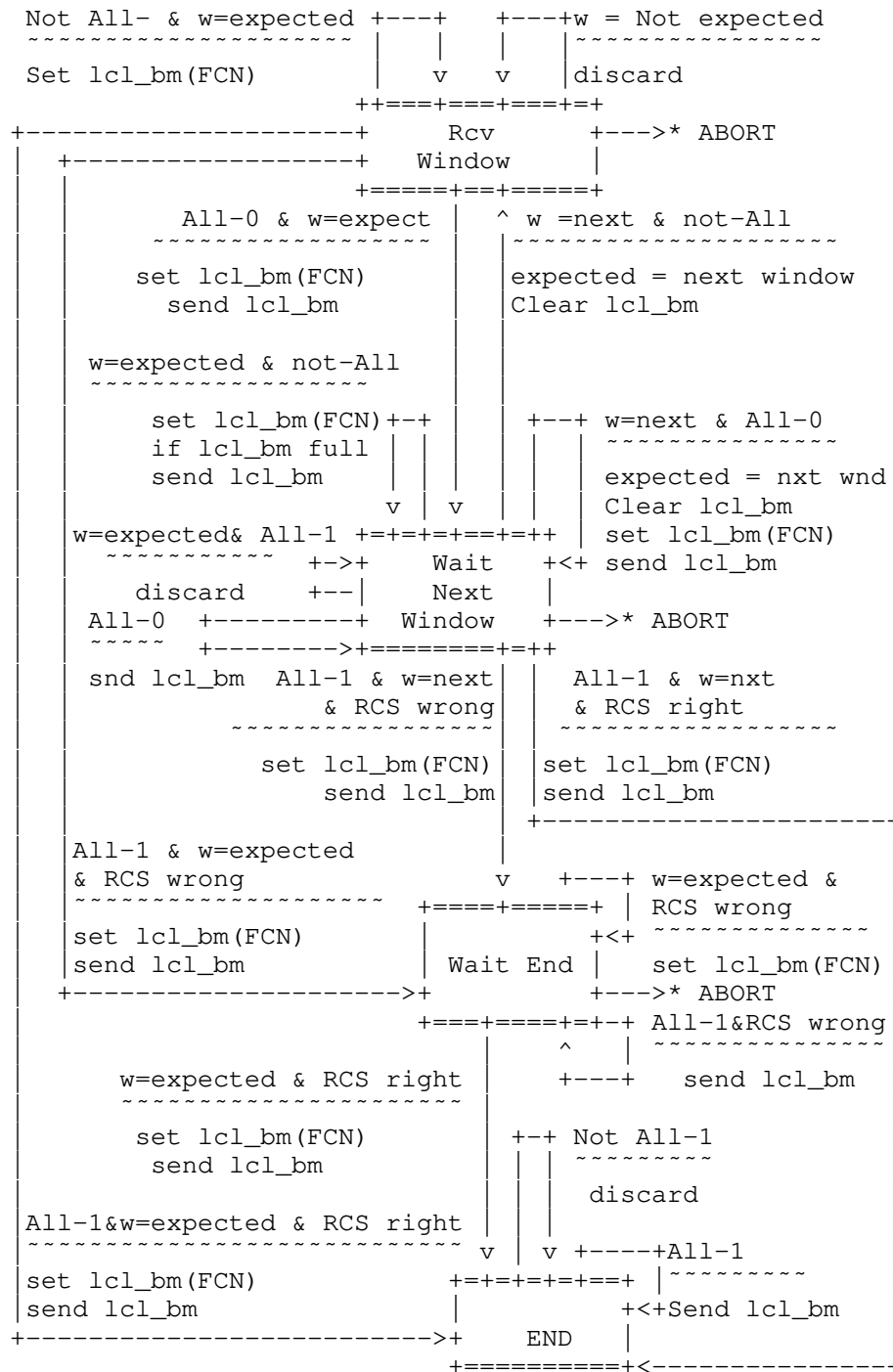


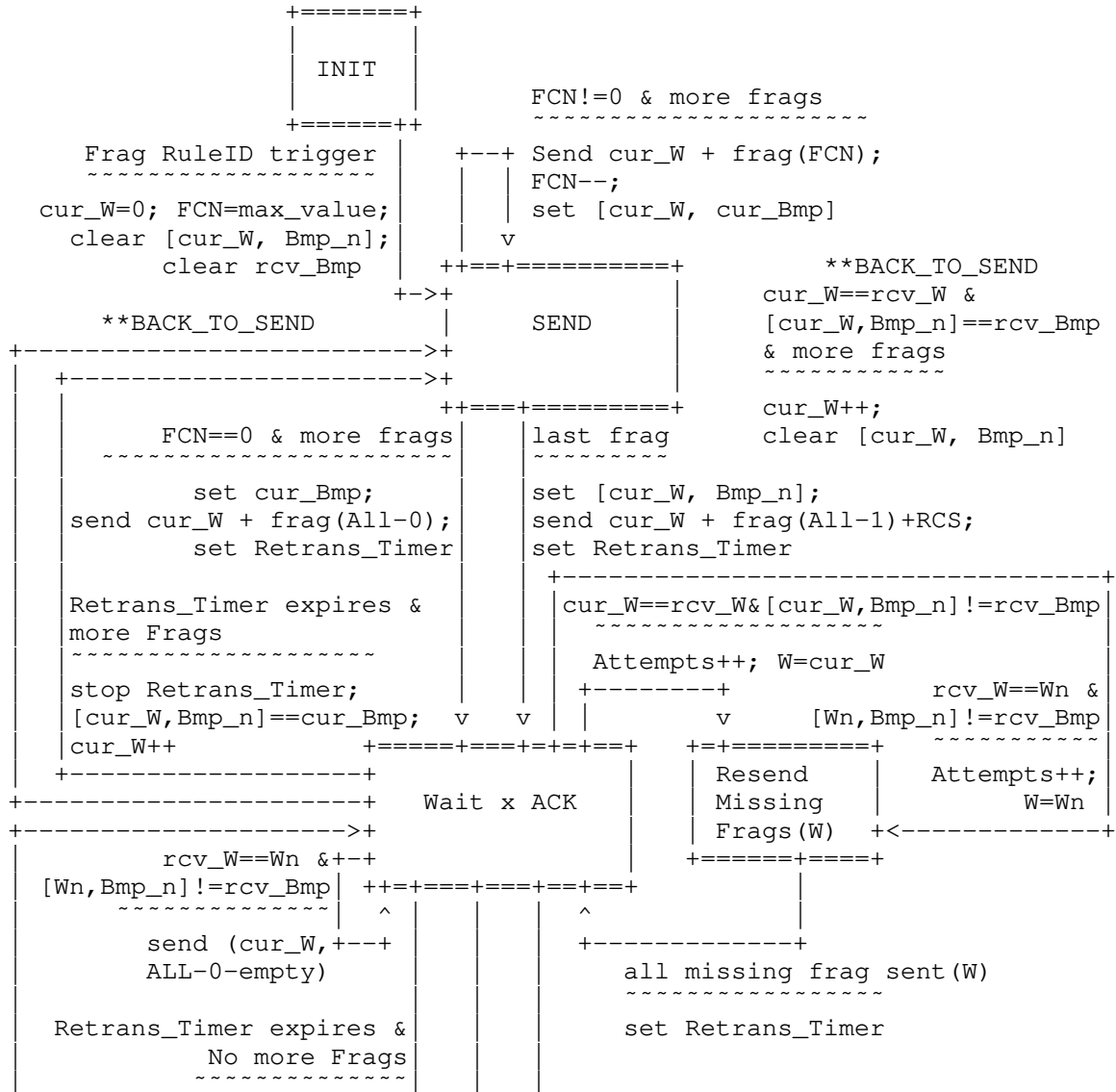
Figure 39: Sender State Machine for the ACK-Always Mode



--->* ABORT

In any state
on receiving a SCHC ACK REQ
Send a SCHC ACK for the current window

Figure 40: Receiver State Machine for the ACK-Always Mode



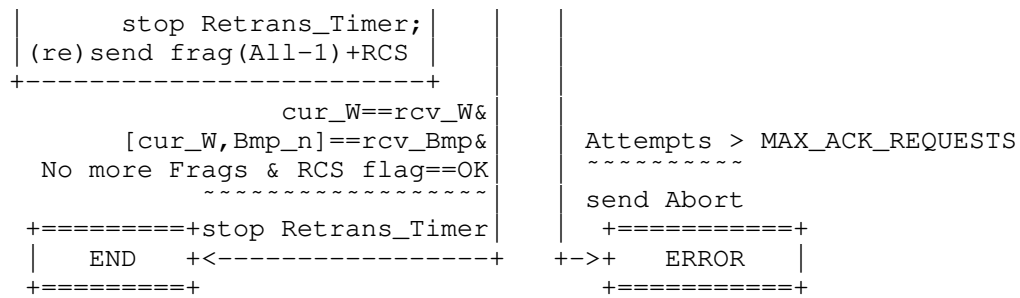
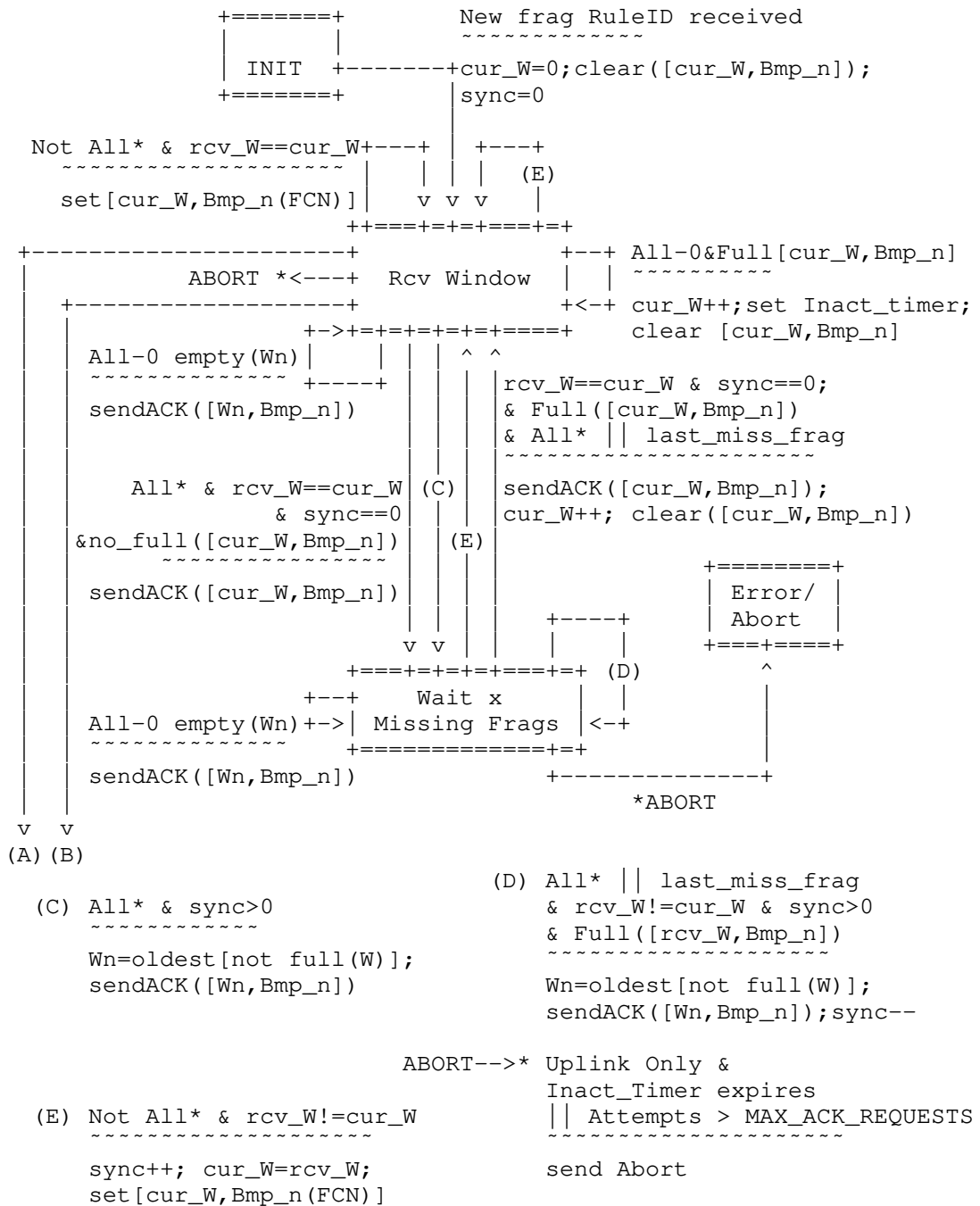


Figure 41: Sender State Machine for the ACK-on-Error Mode

This is an example only. It is not normative. The specification in Section 8.4.3.1 allows for sequences of operations different from the one shown here.



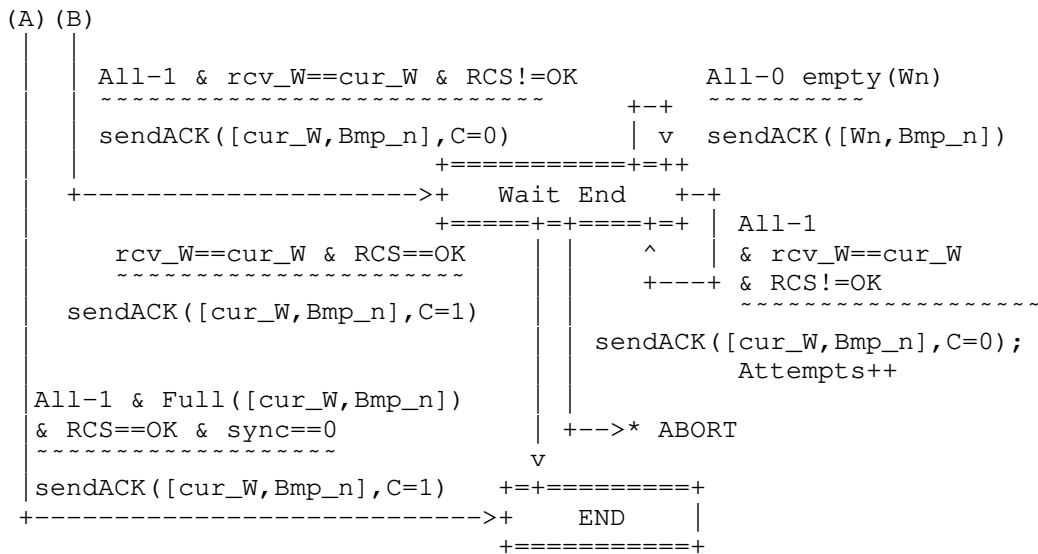


Figure 42: Receiver State Machine for the ACK-on-Error Mode

Appendix D. SCHC Parameters

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

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

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

- o Rule ID numbering scheme, fixed-sized or variable-sized Rule IDs, number of Rules, the way the Rule ID is transmitted
- o maximum packet size that should ever be reconstructed by SCHC Decompression (MAX_PACKET_SIZE). See Section 12.

- o Padding: size of the L2 Word (for most LPWAN technologies, this would be a byte; for some technologies, a bit)
- o Decision to use SCHC fragmentation mechanism or not. If yes, the document must describe:
 - * reliability mode(s) used, in which cases (e.g., based on link channel condition)
 - * Rule ID values assigned to each mode in use
 - * presence and number of bits for DTag (T) for each Rule ID value, lifetime of DTag at the receiver
 - * support for interleaved packet transmission, to what extent
 - * WINDOW_SIZE, for modes that use windows
 - * number of bits for W (M) for each Rule ID value, for modes that use windows
 - * number of bits for FCN (N) for each Rule ID value
 - * what makes an All-0 SCHC Fragment and a SCHC ACK REQ distinguishable (see Section 8.3.1.1).
 - * what makes an All-1 SCHC Fragment and a SCHC Sender-Abort distinguishable (see Section 8.3.1.2).
 - * size of RCS and algorithm for its computation, for each Rule ID, if different from the default CRC32. Byte fill-up with zeroes or other mechanism, to be specified.
 - * Retransmission Timer duration for each Rule ID value, if applicable to the SCHC F/R mode
 - * Inactivity Timer duration for each Rule ID value, if applicable to the SCHC F/R mode
 - * MAX_ACK_REQUESTS value for each Rule ID value, if applicable to the SCHC F/R mode
- o if L2 Word is wider than a bit and SCHC fragmentation is used, value of the padding bits (0 or 1). This is needed because the padding bits of the last fragment are included in the RCS computation.

A Profile may define a delay to be added after each SCHC message transmission for compliance with local regulations or other constraints imposed by the applications.

- o 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 PDU that requires an L2 ACK) or it may be triggered from an upper layer. See Appendix F.
- o the following parameters need to be addressed in documents other than this one but not necessarily in the LPWAN technology-specific documents:
 - * The way the Contexts are provisioned
 - * The way the Rules are generated

Appendix E. Supporting multiple window sizes for fragmentation

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

Appendix F. ACK-Always and ACK-on-Error on quasi-bidirectional links

The ACK-Always and ACK-on-Error modes of SCHC F/R are bidirectional protocols: they require a feedback path from the reassembler to the fragmenter.

Some LPWAN technologies provide quasi-bidirectional connectivity, whereby a downlink transmission from the Network Infrastructure can only take place right after an uplink transmission by the Dev.

When using SCHC F/R to send fragmented SCHC Packets downlink over these quasi-bidirectional links, the following situation may arise: if an uplink SCHC ACK is lost, the SCHC ACK REQ message by the sender

could be stuck indefinitely in the downlink queue at the Network Infrastructure, waiting for a transmission opportunity.

There are many ways by which this deadlock can be avoided. The Dev application might be sending recurring uplink messages such as keep-alive, or the Dev application stack might be sending other recurring uplink messages as part of its operation. However, these are out of the control of this generic SCHC specification.

In order to cope with quasi-bidirectional links, a SCHC-over-foo specification may want to amend the SCHC F/R specification to add a timer-based retransmission of the SCHC ACK. Below is an example of the suggested behavior for ACK-Always mode. Because it is an example, [RFC2119] language is deliberately not used here.

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 UplinkACK Timer) 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 UplinkACK 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 UplinkACK timer for the SCHC ACK is stopped. However, if the UplinkACK timer expires, the SCHC ACK is resent and the UplinkACK timer is reinitialized and restarted.

The default initial value for the UplinkACK Timer, as well as the maximum number of retries for a specific SCHC ACK, denoted MAX_ACK_REQUESTS, is to be defined in a Profile. The initial value of the UplinkACK 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 finds an opportunity for that transmission. One exception to this recommendation is the special case of the All-1 SCHC Fragment transmission.

When the SCHC Fragment sender transmits the All-1 SCHC Fragment, it starts its Retransmission Timer with a large timeout value (e.g., several times that of the initial UplinkACK Timer). If a SCHC ACK is received before expiration of this timer, the SCHC Fragment sender retransmits any lost SCHC Fragments as 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 SCHC 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 SCHC Fragment is long enough to allow several SCHC ACK retries if the All-1 SCHC Fragment has not been received by the SCHC Fragment receiver, and it also assumes that it is unlikely that several ACKs become all lost).

Authors' Addresses

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

Email: ana@ackl.io

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

Email: Laurent.Toutain@imt-atlantique.fr

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

Email: carlesgo@entel.upc.edu

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

Email: dominique.barthel@orange.com

Juan Carlos Zuniga
SIGFOX
425 rue Jean Rostand
Labège 31670
France

Email: JuanCarlos.Zuniga@sigfox.com

lpwan
Internet-Draft
Intended status: Informational
Expires: August 11, 2018

S. Farrell, Ed.
Trinity College Dublin
February 7, 2018

LPWAN Overview
draft-ietf-lpwan-overview-10

Abstract

Low Power Wide Area Networks (LPWAN) are wireless technologies with characteristics such as large coverage areas, low bandwidth, possibly very small packet and application layer data sizes and long battery life operation. This memo is an informational overview of the set of LPWAN technologies being considered in the IETF and of the gaps that exist between the needs of those technologies and the goal of running IP in LPWANs.

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 <http://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 August 11, 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 (<http://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. LPWAN Technologies	3
2.1. LoRaWAN	4
2.1.1. Provenance and Documents	4
2.1.2. Characteristics	4
2.2. Narrowband IoT (NB-IoT)	11
2.2.1. Provenance and Documents	11
2.2.2. Characteristics	11
2.3. SIGFOX	15
2.3.1. Provenance and Documents	15
2.3.2. Characteristics	16
2.4. Wi-SUN Alliance Field Area Network (FAN)	20
2.4.1. Provenance and Documents	20
2.4.2. Characteristics	21
3. Generic Terminology	24
4. Gap Analysis	25
4.1. Naive application of IPv6	26
4.2. 6LoWPAN	26
4.2.1. Header Compression	27
4.2.2. Address Autoconfiguration	27
4.2.3. Fragmentation	27
4.2.4. Neighbor Discovery	28
4.3. 6lo	29
4.4. 6tisch	29
4.5. RoHC	29
4.6. ROLL	30
4.7. CoAP	30
4.8. Mobility	30
4.9. DNS and LPWAN	31
5. Security Considerations	31
6. IANA Considerations	32
7. Contributors	32
8. Acknowledgments	35
9. Informative References	35
Appendix A. Changes	41
A.1. From -00 to -01	41
A.2. From -01 to -02	41
A.3. From -02 to -03	41
A.4. From -03 to -04	42
A.5. From -04 to -05	42
A.6. From -05 to -06	42
A.7. From -06 to -07	42
A.8. From -07 to -08	42

A.9. From -08 to -09	43
A.10. From -09 to -10	43
Author's Address	43

1. Introduction

This document provides background material and an overview of the technologies being considered in the IETF's Low Power Wide-Area Networking (LPWAN) working group. We also provide a gap analysis between the needs of these technologies and currently available IETF specifications.

Most technologies in this space aim for similar goals of supporting large numbers of very low-cost, low-throughput devices with very-low power consumption, so that even battery-powered devices can be deployed for years. LPWAN devices also tend to be constrained in their use of bandwidth, for example with limited frequencies being allowed to be used within limited duty-cycles (usually expressed as a percentage of time per-hour that the device is allowed to transmit.) And as the name implies, coverage of large areas is also a common goal. So, by and large, the different technologies aim for deployment in very similar circumstances.

What mainly distinguishes LPWANs from other constrained networks is that in LPWANs the balancing act related to power consumption/battery life, cost and bandwidth tends to prioritise doing better with respect to power and cost and we are more willing to live with extremely low bandwidth and constrained duty-cycles when making the various trade-offs required, in order to get the multiple-kilometre radio links implied by the "wide area" aspect of the LPWAN term.

Existing pilot deployments have shown huge potential and created much industrial interest in these technologies. As of today, essentially no LPWAN end-devices (other than for Wi-SUN) have IP capabilities. Connecting LPWANs to the Internet would provide significant benefits to these networks in terms of interoperability, application deployment, and management, among others. The goal of the IETF LPWAN working group is to, where necessary, adapt IETF-defined protocols, addressing schemes and naming to this particular constrained environment.

This document is largely the work of the people listed in Section 7.

2. LPWAN Technologies

This section provides an overview of the set of LPWAN technologies that are being considered in the LPWAN working group. The text for each was mainly contributed by proponents of each technology.

Note that this text is not intended to be normative in any sense, but simply to help the reader in finding the relevant layer 2 specifications and in understanding how those integrate with IETF-defined technologies. Similarly, there is no attempt here to set out the pros and cons of the relevant technologies.

Note that some of the technology-specific drafts referenced below may have been updated since publication of this document.

2.1. LoRaWAN

2.1.1. Provenance and Documents

LoRaWAN is an ISM-based wireless technology for long-range low-power low-data-rate applications developed by the LoRa Alliance, a membership consortium. <<https://www.lora-alliance.org/>> This draft is based on version 1.0.2 [LoRaSpec] of the LoRa specification. That specification is publicly available and has already seen several deployments across the globe.

2.1.2. Characteristics

LoRaWAN aims to support end-devices operating on a single battery for an extended period of time (e.g., 10 years or more), extended coverage through 155 dB maximum coupling loss, and reliable and efficient file download (as needed for remote software/firmware upgrade).

LoRaWAN networks are typically organized in a star-of-stars topology in which gateways relay messages between end-devices and a central "network server" in the backend. Gateways are connected to the network server via IP links while end-devices use single-hop LoRaWAN communication that can be received at one or more gateways. Communication is generally bi-directional; uplink communication from end-devices to the network server is favored in terms of overall bandwidth availability.

Figure 1 shows the entities involved in a LoRaWAN network.

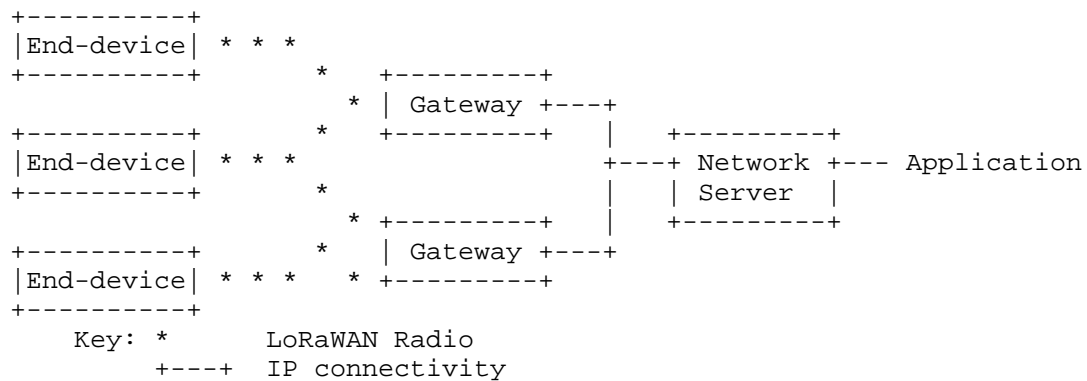


Figure 1: LoRaWAN architecture

- o End-device: a LoRa client device, sometimes called a mote. Communicates with gateways.
- o Gateway: a radio on the infrastructure-side, sometimes called a concentrator or base-station. Communicates with end-devices and, via IP, with a network server.
- o Network Server: The Network Server (NS) terminates the LoRaWAN MAC layer for the end-devices connected to the network. It is the center of the star topology.
- o Join Server: The Join Server (JS) is a server on the Internet side of an NS that processes join requests from an end-devices.
- o Uplink message: refers to communications from an end-device to a network server or application via one or more gateways.
- o Downlink message: refers to communications from a network server or application via one gateway to a single end-device or a group of end-devices (considering multicasting).
- o Application: refers to application layer code both on the end-device and running "behind" the network server. For LoRaWAN, there will generally only be one application running on most end-devices. Interfaces between the network server and application are not further described here.

In LoRaWAN networks, end-device transmissions may be received at multiple gateways, so during nominal operation a network server may see multiple instances of the same uplink message from an end-device.

The LoRaWAN network infrastructure manages the data rate and RF output power for each end-device individually by means of an adaptive data rate (ADR) scheme. End-devices may transmit on any channel allowed by local regulation at any time.

LoRaWAN radios make use of industrial, scientific and medical (ISM) bands, for example, 433MHz and 868MHz within the European Union and 915MHz in the Americas.

The end-device changes channel in a pseudo-random fashion for every transmission to help make the system more robust to interference and/or to conform to local regulations.

Figure 2 below shows that after a transmission slot a Class A device turns on its receiver for two short receive windows that are offset from the end of the transmission window. End-devices can only transmit a subsequent uplink frame after the end of the associated receive windows. When a device joins a LoRaWAN network, there are similar timeouts on parts of that process.

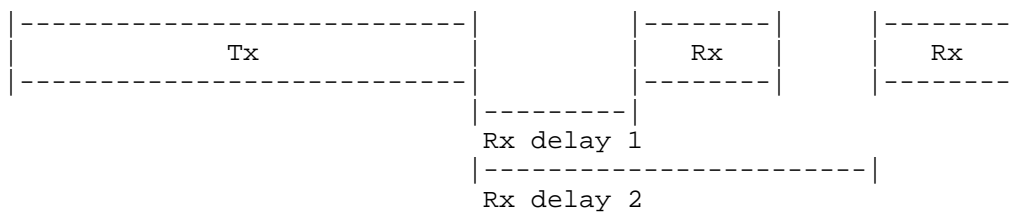


Figure 2: LoRaWAN Class A transmission and reception window

Given the different regional requirements the detailed specification for the LoRaWAN physical layer (taking up more than 30 pages of the specification) is not reproduced here. Instead and mainly to illustrate the kinds of issue encountered, in Table 1 we present some of the default settings for one ISM band (without fully explaining those here) and in Table 2 we describe maxima and minima for some parameters of interest to those defining ways to use IETF protocols over the LoRaWAN MAC layer.

Parameters	Default Value
Rx delay 1	1 s
Rx delay 2	2 s (must be RECEIVE_DELAY1 + 1s)
join delay 1	5 s
join delay 2	6 s
868MHz Default channels	3 (868.1,868.2,868.3), data rate: 0.3-50kbps

Table 1: Default settings for EU 868MHz band

Parameter/Notes	Min	Max
Duty Cycle: some but not all ISM bands impose a limit in terms of how often an end-device can transmit. In some cases LoRaWAN is more restrictive in an attempt to avoid congestion.	1%	no-limit
EU 868MHz band data rate/frame-size	250 bits/s : 59 octets	50000 bits/s : 250 octets
US 915MHz band data rate/frame-size	980 bits/s : 19 octets	21900 bits/s : 250 octets

Table 2: Minima and Maxima for various LoRaWAN Parameters

Note that in the case of the smallest frame size (19 octets), 8 octets are required for LoRa MAC layer headers leaving only 11 octets for payload (including MAC layer options). However, those settings do not apply for the join procedure - end-devices are required to use a channel and data rate that can send the 23-byte Join-request message for the join procedure.

Uplink and downlink higher layer data is carried in a MACPayload. There is a concept of "ports" (an optional 8-bit value) to handle

different applications on an end-device. Port zero is reserved for LoRaWAN specific messaging, such as the configuration of the end device's network parameters (available channels, data rates, ADR parameters, RX1/2 delay, etc.).

In addition to carrying higher layer PDUs there are Join-Request and Join-Response (aka Join-Accept) messages for handling network access. And so-called "MAC commands" (see below) up to 15 bytes long can be piggybacked in an options field ("FOpts").

There are a number of MAC commands for link and device status checking, ADR and duty-cycle negotiation, managing the RX windows and radio channel settings. For example, the link check response message allows the network server (in response to a request from an end-device) to inform an end-device about the signal attenuation seen most recently at a gateway, and to also tell the end-device how many gateways received the corresponding link request MAC command.

Some MAC commands are initiated by the network server. For example, one command allows the network server to ask an end-device to reduce its duty-cycle to only use a proportion of the maximum allowed in a region. Another allows the network server to query the end-device's power status with the response from the end-device specifying whether it has an external power source or is battery powered (in which case a relative battery level is also sent to the network server).

In order to operate nominally on a LoRaWAN network, a device needs a 32-bit device address, that is assigned when the device "joins" the network (see below for the join procedure) or that is pre-provisioned into the device. In case of roaming devices, the device address is assigned based on the 24-bit network identifier (NetID) that is allocated to the network by the LoRa Alliance. Non-roaming devices can be assigned device addresses by the network without relying on a LoRa Alliance-assigned NetID.

End-devices are assumed to work with one or a quite limited number of applications, identified by a 64-bit AppEUI, which is assumed to be a registered IEEE EUI64 value. In addition, a device needs to have two symmetric session keys, one for protecting network artifacts (port=0), the NwksKey, and another for protecting application layer traffic, the AppSKey. Both keys are used for 128-bit AES cryptographic operations. So, one option is for an end-device to have all of the above, plus channel information, somehow (pre-)provisioned, in which case the end-device can simply start transmitting. This is achievable in many cases via out-of-band means given the nature of LoRaWAN networks. Table 3 summarizes these values.

Value	Description
DevAddr	DevAddr (32-bits) = device-specific network address generated from the NetID
AppEUI	IEEE EUI64 corresponding to the join server for an application
NwksKey	128-bit network session key used with AES-CMAC
AppSKey	128-bit application session key used with AES-CTR
AppKey	128-bit application session key used with AES-ECB

Table 3: Values required for nominal operation

As an alternative, end-devices can use the LoRaWAN join procedure with a join server behind the NS in order to setup some of these values and dynamically gain access to the network. To use the join procedure, an end-device must still know the AppEUI, and in addition, a different (long-term) symmetric key that is bound to the AppEUI - this is the application key (AppKey), and is distinct from the application session key (AppSKey). The AppKey is required to be specific to the device, that is, each end-device should have a different AppKey value. And finally, the end-device also needs a long-term identifier for itself, syntactically also an EUI-64, and known as the device EUI or DevEUI. Table 4 summarizes these values.

Value	Description
DevEUI	IEEE EUI64 naming the device
AppEUI	IEEE EUI64 naming the application
AppKey	128-bit long term application key for use with AES

Table 4: Values required for join procedure

The join procedure involves a special exchange where the end-device asserts the AppEUI and DevEUI (integrity protected with the long-term AppKey, but not encrypted) in a Join-request uplink message. This is then routed to the network server which interacts with an entity that knows that AppKey to verify the Join-request. All going well, a Join-accept downlink message is returned from the network server to

the end-device that specifies the 24-bit NetID, 32-bit DevAddr and channel information and from which the AppSKey and NwkSKey can be derived based on knowledge of the AppKey. This provides the end-device with all the values listed in Table 3.

All payloads are encrypted and have data integrity. MAC commands, when sent as a payload (port zero), are therefore protected. MAC commands piggy-backed as frame options ("FOpts") are however sent in clear. Any MAC commands sent as frame options and not only as payload, are visible to a passive attacker but are not malleable for an active attacker due to the use of the Message Integrity Check (MIC) described below.

For LoRaWAN version 1.0.x, the NwkSKey session key is used to provide data integrity between the end-device and the network server. The AppSKey is used to provide data confidentiality between the end-device and network server, or to the application "behind" the network server, depending on the implementation of the network.

All MAC layer messages have an outer 32-bit MIC calculated using AES-CMAC calculated over the ciphertext payload and other headers and using the NwkSKey. Payloads are encrypted using AES-128, with a counter-mode derived from IEEE 802.15.4 using the AppSKey. Gateways are not expected to be provided with the AppSKey or NwkSKey, all of the infrastructure-side cryptography happens in (or "behind") the network server. When session keys are derived from the AppKey as a result of the join procedure the Join-accept message payload is specially handled.

The long-term AppKey is directly used to protect the Join-accept message content, but the function used is not an AES-encrypt operation, but rather an AES-decrypt operation. The justification is that this means that the end-device only needs to implement the AES-encrypt operation. (The counter mode variant used for payload decryption means the end-device doesn't need an AES-decrypt primitive.)

The Join-accept plaintext is always less than 16 bytes long, so electronic code book (ECB) mode is used for protecting Join-accept messages. The Join-accept contains an AppNonce (a 24 bit value) that is recovered on the end-device along with the other Join-accept content (e.g. DevAddr) using the AES-encrypt operation. Once the Join-accept payload is available to the end-device the session keys are derived from the AppKey, AppNonce and other values, again using an ECB mode AES-encrypt operation, with the plaintext input being a maximum of 16 octets.

2.2. Narrowband IoT (NB-IoT)

2.2.1. Provenance and Documents

Narrowband Internet of Things (NB-IoT) is developed and standardized by 3GPP. The standardization of NB-IoT was finalized with 3GPP Release 13 in June 2016, and further enhancements for NB-IoT are specified in 3GPP Release 14 in 2017, for example in the form of multicast support. Further features and improvements will be developed in the following releases, but NB-IoT has been ready to be deployed since 2016, and is rather simple to deploy especially in the existing LTE networks with a software upgrade in the operator's base stations. For more information of what has been specified for NB-IoT, 3GPP specification 36.300 [TGPP36300] provides an overview and overall description of the E-UTRAN radio interface protocol architecture, while specifications 36.321 [TGPP36321], 36.322 [TGPP36322], 36.323 [TGPP36323] and 36.331 [TGPP36331] give more detailed description of MAC, Radio Link Control (RLC), Packet Data Convergence Protocol (PDCP) and Radio Resource Control (RRC) protocol layers, respectively. Note that the description below assumes familiarity with numerous 3GPP terms.

For a general overview of NB-IoT, see [nbiot-ov].

2.2.2. Characteristics

Specific targets for NB-IoT include: Less than US\$5 module cost, extended coverage of 164 dB maximum coupling loss, battery life of over 10 years, ~55000 devices per cell and uplink reporting latency of less than 10 seconds.

NB-IoT supports Half Duplex FDD operation mode with 60 kbps peak rate in uplink and 30 kbps peak rate in downlink, and a maximum transmission unit (MTU) size of 1600 bytes limited by PDCP layer (see Figure 4 for the protocol structure), which is the highest layer in the user plane, as explained later. Any packet size up to the said MTU size can be passed to the NB-IoT stack from higher layers, segmentation of the packet is performed in the RLC layer, which can segment the data to transmission blocks with size as small as 16 bits. As the name suggests, NB-IoT uses narrowbands with bandwidth of 180 kHz in both downlink and uplink. The multiple access scheme used in the downlink is OFDMA with 15 kHz sub-carrier spacing. In uplink, SC-FDMA single tone with either 15kHz or 3.75 kHz tone spacing is used, or optionally multi-tone SC-FDMA can be used with 15 kHz tone spacing.

NB-IoT can be deployed in three ways. In-band deployment means that the narrowband is deployed inside the LTE band and radio resources

are flexibly shared between NB-IoT and normal LTE carrier. In Guard-band deployment the narrowband uses the unused resource blocks between two adjacent LTE carriers. Standalone deployment is also supported, where the narrowband can be located alone in dedicated spectrum, which makes it possible for example to reframe a GSM carrier at 850/900 MHz for NB-IoT. All three deployment modes are used in licensed frequency bands. The maximum transmission power is either 20 or 23 dBm for uplink transmissions, while for downlink transmission the eNodeB may use higher transmission power, up to 46 dBm depending on the deployment.

A maximum coupling loss (MCL) target for NB-IoT coverage enhancements defined by 3GPP is 164 dB. With this MCL, the performance of NB-IoT in downlink varies between 200 bps and 2-3 kbps, depending on the deployment mode. Stand-alone operation may achieve the highest data rates, up to few kbps, while in-band and guard-band operations may reach several hundreds of bps. NB-IoT may even operate with MCL higher than 170 dB with very low bit rates.

For signaling optimization, two options are introduced in addition to legacy LTE RRC connection setup; mandatory Data-over-NAS (Control Plane optimization, solution 2 in [TGPP23720]) and optional RRC Suspend/Resume (User Plane optimization, solution 18 in [TGPP23720]). In the control plane optimization the data is sent over Non-Access Stratum, directly to/from Mobility Management Entity (MME) (see Figure 3 for the network architecture) in the core network to the User Equipment (UE) without interaction from the base station. This means there are no Access Stratum security or header compression provided by the PDCP layer in the eNodeB, as the Access Stratum is bypassed, and only limited RRC procedures. RoHC based header compression may still optionally be provided and terminated in MME.

The RRC Suspend/Resume procedures reduce the signaling overhead required for UE state transition from RRC Idle to RRC Connected mode compared to legacy LTE operation in order to have quicker user plane transaction with the network and return to RRC Idle mode faster.

In order to prolong device battery life, both power-saving mode (PSM) and extended DRX (eDRX) are available to NB-IoT. With eDRX the RRC Connected mode DRX cycle is up to 10.24 seconds and in RRC Idle the eDRX cycle can be up to 3 hours. In PSM the device is in a deep sleep state and only wakes up for uplink reporting, after which there is a window, configured by the network, during which the device receiver is open for downlink connectivity, or for periodical "keep-alive" signaling (PSM uses periodic TAU signaling with additional reception window for downlink reachability).

Since NB-IoT operates in licensed spectrum, it has no channel access restrictions allowing up to a 100% duty-cycle.

3GPP access security is specified in [TGPP33203].

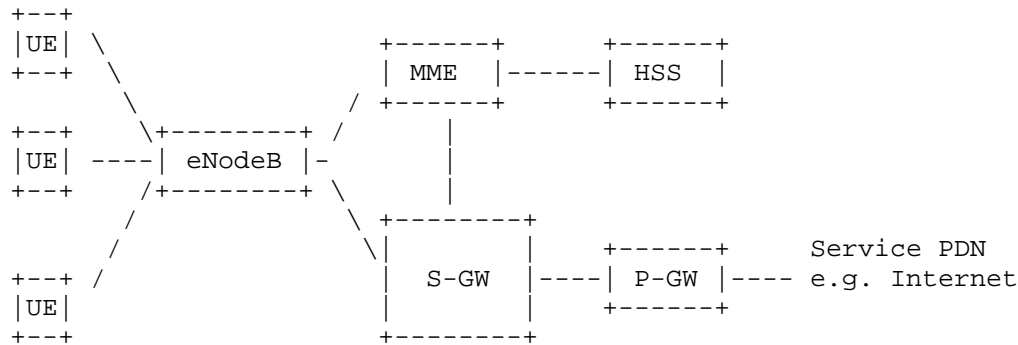


Figure 3: 3GPP network architecture

Figure 3 shows the 3GPP network architecture, which applies to NB-IoT. Mobility Management Entity (MME) is responsible for handling the mobility of the UE. MME tasks include tracking and paging UEs, session management, choosing the Serving gateway for the UE during initial attachment and authenticating the user. At MME, the Non-Access Stratum (NAS) signaling from the UE is terminated.

Serving Gateway (S-GW) routes and forwards the user data packets through the access network and acts as a mobility anchor for UEs during handover between base stations known as eNodeBs and also during handovers between NB-IoT and other 3GPP technologies.

Packet Data Network Gateway (P-GW) works as an interface between 3GPP network and external networks.

The Home Subscriber Server (HSS) contains user-related and subscription-related information. It is a database, which performs mobility management, session establishment support, user authentication and access authorization.

E-UTRAN consists of components of a single type, eNodeB. eNodeB is a base station, which controls the UEs in one or several cells.

The 3GPP radio protocol architecture is illustrated in Figure 4.

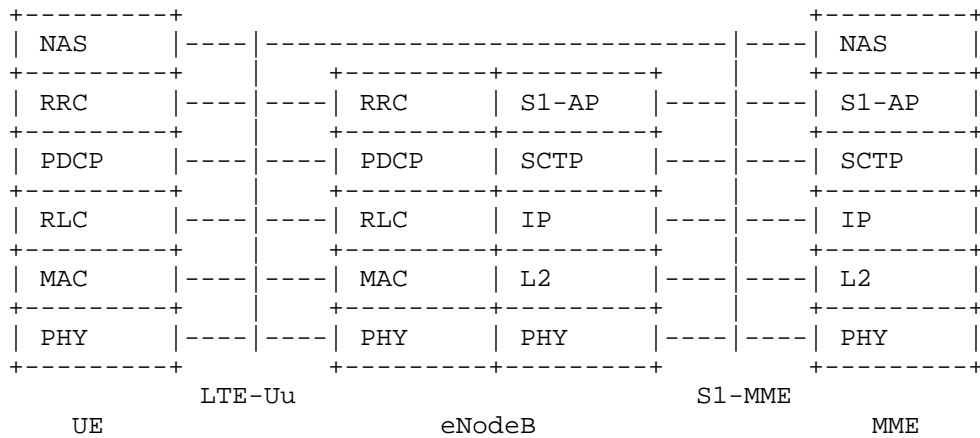


Figure 4: 3GPP radio protocol architecture for control plane

Control plane protocol stack

The radio protocol architecture of NB-IoT (and LTE) is separated into control plane and user plane. The control plane consists of protocols which control the radio access bearers and the connection between the UE and the network. The highest layer of control plane is called Non-Access Stratum (NAS), which conveys the radio signaling between the UE and the Evolved Packet Core (EPC), passing transparently through the radio network. NAS responsible for authentication, security control, mobility management and bearer management.

Access Stratum (AS) is the functional layer below NAS, and in the control plane it consists of Radio Resource Control protocol (RRC) [TGPP36331], which handles connection establishment and release functions, broadcast of system information, radio bearer establishment, reconfiguration and release. RRC configures the user and control planes according to the network status. There exists two RRC states, RRC_Idle or RRC_Connected, and RRC entity controls the switching between these states. In RRC_Idle, the network knows that the UE is present in the network and the UE can be reached in case of incoming call/downlink data. In this state, the UE monitors paging, performs cell measurements and cell selection and acquires system information. Also the UE can receive broadcast and multicast data, but it is not expected to transmit or receive unicast data. In RRC_Connected the UE has a connection to the eNodeB, the network knows the UE location on the cell level and the UE may receive and transmit unicast data. An RRC connection is established when the UE is expected to be active in the network, to transmit or receive data. The RRC connection is released, switching back to RRC_Idle, when

there is no more traffic in order to preserve UE battery life and radio resources. However, a new feature was introduced for NB-IoT, as mentioned earlier, which allows data to be transmitted from the MME directly to the UE transparently to the eNodeB, thus bypassing AS functions.

Packet Data Convergence Protocol's (PDCP) [TGPP36323] main services in control plane are transfer of control plane data, ciphering and integrity protection.

Radio Link Control protocol (RLC) [TGPP36322] performs transfer of upper layer PDUs and optionally error correction with Automatic Repeat reQuest (ARQ), concatenation, segmentation, and reassembly of RLC SDUs, in-sequence delivery of upper layer PDUs, duplicate detection, RLC SDU discard, RLC-re-establishment and protocol error detection and recovery.

Medium Access Control protocol (MAC) [TGPP36321] provides mapping between logical channels and transport channels, multiplexing of MAC SDUs, scheduling information reporting, error correction with HARQ, priority handling and transport format selection.

Physical layer [TGPP36201] provides data transport services to higher layers. These include error detection and indication to higher layers, FEC encoding, HARQ soft-combining, rate matching and mapping of the transport channels onto physical channels, power weighting and modulation of physical channels, frequency and time synchronization and radio characteristics measurements.

User plane is responsible for transferring the user data through the Access Stratum. It interfaces with IP and the highest layer of user plane is PDCP, which in user plane performs header compression using Robust Header Compression (RoHC), transfer of user plane data between eNodeB and UE, ciphering and integrity protection. Similar to control plane, lower layers in user plane include RLC, MAC and physical layer performing the same tasks as in control plane.

2.3. SIGFOX

2.3.1. Provenance and Documents

The SIGFOX LPWAN is in line with the terminology and specifications being defined by ETSI [etsi_unb]. As of today, SIGFOX's network has been fully deployed in 12 countries, with ongoing deployments on 26 other countries, giving in total a geography of 2 million square kilometers, containing 512 million people.

2.3.2. Characteristics

SIGFOX LPWAN autonomous battery-operated devices send only a few bytes per day, week or month, in principle allowing them to remain on a single battery for up to 10-15 years. Hence, the system is designed as to allow devices to last several years, sometimes even buried underground.

Since the radio protocol is connection-less and optimized for uplink communications, the capacity of a SIGFOX base station depends on the number of messages generated by devices, and not on the actual number of devices. Likewise, the battery life of devices depends on the number of messages generated by the device. Depending on the use case, devices can vary from sending less than one message per device per day, to dozens of messages per device per day.

The coverage of the cell depends on the link budget and on the type of deployment (urban, rural, etc.). The radio interface is compliant with the following regulations:

Spectrum allocation in the USA [fcc_ref]

Spectrum allocation in Europe [etsi_ref]

Spectrum allocation in Japan [arib_ref]

The SIGFOX radio interface is also compliant with the local regulations of the following countries: Australia, Brazil, Canada, Kenya, Lebanon, Mauritius, Mexico, New Zealand, Oman, Peru, Singapore, South Africa, South Korea, and Thailand.

The radio interface is based on Ultra Narrow Band (UNB) communications, which allow an increased transmission range by spending a limited amount of energy at the device. Moreover, UNB allows a large number of devices to coexist in a given cell without significantly increasing the spectrum interference.

Both uplink and downlink are supported, although the system is optimized for uplink communications. Due to spectrum optimizations, different uplink and downlink frames and time synchronization methods are needed.

The main radio characteristics of the UNB uplink transmission are:

- o Channelization mask: 100 Hz / 600 Hz (depending on the region)
- o Uplink baud rate: 100 baud / 600 baud (depending on the region)

- o Modulation scheme: DBPSK
- o Uplink transmission power: compliant with local regulation
- o Link budget: 155 dB (or better)
- o Central frequency accuracy: not relevant, provided there is no significant frequency drift within an uplink packet transmission

For example, in Europe the UNB uplink frequency band is limited to 868.00 to 868.60 MHz, with a maximum output power of 25 mW and a duty cycle of 1%.

The format of the uplink frame is the following:

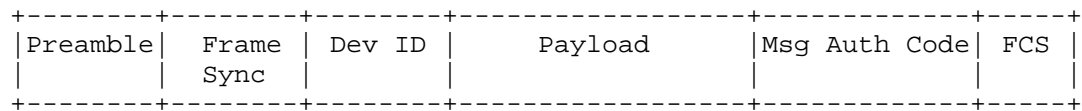


Figure 5: Uplink Frame Format

The uplink frame is composed of the following fields:

- o Preamble: 19 bits
- o Frame sync and header: 29 bits
- o Device ID: 32 bits
- o Payload: 0-96 bits
- o Authentication: 16-40 bits
- o Frame check sequence: 16 bits (CRC)

The main radio characteristics of the UNB downlink transmission are:

- o Channelization mask: 1.5 kHz
- o Downlink baud rate: 600 baud
- o Modulation scheme: GFSK
- o Downlink transmission power: 500 mW / 4W (depending on the region)
- o Link budget: 153 dB (or better)

- o Central frequency accuracy: the center frequency of downlink transmission is set by the network according to the corresponding uplink transmission

For example, in Europe the UNB downlink frequency band is limited to 869.40 to 869.65 MHz, with a maximum output power of 500 mW with 10% duty cycle.

The format of the downlink frame is the following:

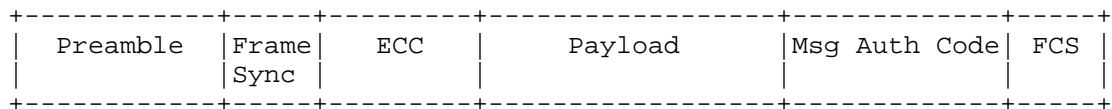


Figure 6: Downlink Frame Format

The downlink frame is composed of the following fields:

- o Preamble: 91 bits
- o Frame sync and header: 13 bits
- o Error Correcting Code (ECC): 32 bits
- o Payload: 0-64 bits
- o Authentication: 16 bits
- o Frame check sequence: 8 bits (CRC)

The radio interface is optimized for uplink transmissions, which are asynchronous. Downlink communications are achieved by devices querying the network for available data.

A device willing to receive downlink messages opens a fixed window for reception after sending an uplink transmission. The delay and duration of this window have fixed values. The network transmits the downlink message for a given device during the reception window, and the network also selects the base station (BS) for transmitting the corresponding downlink message.

Uplink and downlink transmissions are unbalanced due to the regulatory constraints on ISM bands. Under the strictest regulations, the system can allow a maximum of 140 uplink messages and 4 downlink messages per device per day. These restrictions can

be slightly relaxed depending on system conditions and the specific regulatory domain of operation.

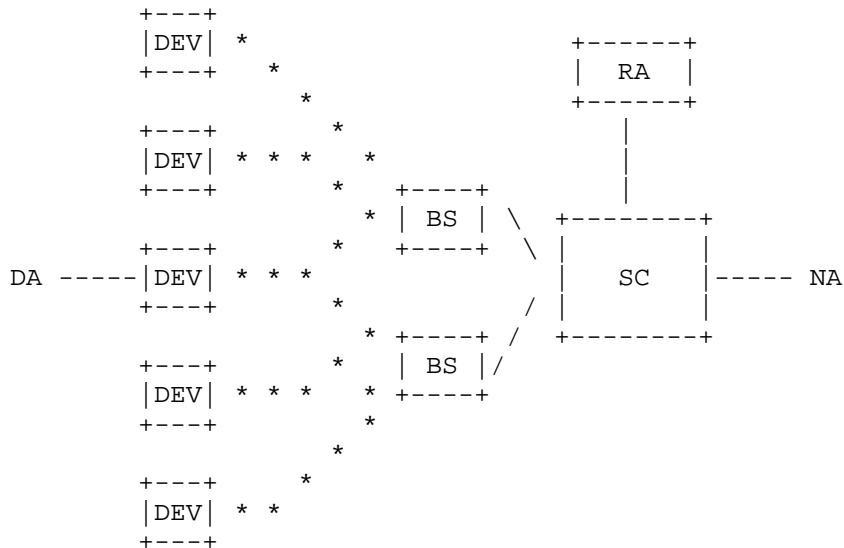


Figure 7: SIGFOX network architecture

Figure 7 depicts the different elements of the SIGFOX network architecture.

SIGFOX has a "one-contract one-network" model allowing devices to connect in any country, without any need or notion of either roaming or handover.

The architecture consists of a single cloud-based core network, which allows global connectivity with minimal impact on the end device and radio access network. The core network elements are the Service Center (SC) and the Registration Authority (RA). The SC is in charge of the data connectivity between the Base Station (BS) and the Internet, as well as the control and management of the BSs and End Points. The RA is in charge of the End Point network access authorization.

The radio access network is comprised of several BSs connected directly to the SC. Each BS performs complex L1/L2 functions, leaving some L2 and L3 functionalities to the SC.

The Devices (DEVs) or End Points (EPs) are the objects that communicate application data between local device applications (DAs) and network applications (NAs).

Devices (or EPs) can be static or nomadic, as they associate with the SC and they do not attach to any specific BS. Hence, they can communicate with the SC through one or multiple BSs.

Due to constraints in the complexity of the Device, it is assumed that Devices host only one or very few device applications, which most of the time communicate each to a single network application at a time.

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. At the time of writing the algorithms and keying details for this are not published.

2.4. Wi-SUN Alliance Field Area Network (FAN)

Text here is via personal communication from Bob Heile (bheile@ieee.org) and was authored by Bob and Sum Chin Sean. Duffy (paduffy@cisco.com) also provided additional comments/input on this section.

2.4.1. Provenance and Documents

The Wi-SUN Alliance <<https://www.wi-sun.org/>> is an industry alliance for smart city, smart grid, smart utility, and a broad set of general IoT applications. The Wi-SUN Alliance Field Area Network (FAN) profile is open standards based (primarily on IETF and IEEE802 standards) and was developed to address applications like smart municipality/city infrastructure monitoring and management, electric vehicle (EV) infrastructure, advanced metering infrastructure (AMI), distribution automation (DA), supervisory control and data acquisition (SCADA) protection/management, distributed generation monitoring and management, and many more IoT applications. Additionally, the Alliance has created a certification program to promote global multi-vendor interoperability.

The FAN profile is specified within ANSI/TIA as an extension of work previously done on Smart Utility Networks. [ANSI-4957-000]. Updates to those specifications intended to be published in 2017 will contain details of the FAN profile. A current snapshot of the work to

produce that profile is presented in [wisun-pressie1]
[wisun-pressie2] .

2.4.2. Characteristics

The FAN profile is an IPv6 wireless mesh network with support for enterprise level security. The frequency hopping wireless mesh topology aims to offer superior network robustness, reliability due to high redundancy, good scalability due to the flexible mesh configuration and good resilience to interference. Very low power modes are in development permitting long term battery operation of network nodes.

The following list contains some overall characteristics of Wi-SUN that are relevant to LPWAN applications.

- o Coverage: The range of Wi-SUN FAN is typically 2 -- 3 km in line of sight, matching the needs of neighborhood area networks, campus area networks, or corporate area networks. The range can also be extended via multi-hop networking.
- o High bandwidth, low link latency: Wi-SUN supports relatively high bandwidth, i.e. up to 300 kbps [FANTPS], enables remote update and upgrade of devices so that they can handle new applications, extending their working life. Wi-SUN supports LPWAN IoT applications that require on-demand control by providing low link latency (0.02s) and bi-directional communication.
- o Low power consumption: FAN devices draw less than 2 uA when resting and only 8 mA when listening. Such devices can maintain a long lifetime even if they are frequently listening. For instance, suppose the device transmits data for 10 ms once every 10 s; theoretically, a battery of 1000 mAh can last more than 10 years.
- o Scalability: Tens of millions Wi-SUN FAN devices have been deployed in urban, suburban and rural environments, including deployments with more than 1 million devices.

A FAN contains one or more networks. Within a network, nodes assume one of three operational roles. First, each network contains a Border Router providing Wide Area Network (WAN) connectivity to the network. The Border Router maintains source routing tables for all nodes within its network, provides node authentication and key management services, and disseminates network-wide information such as broadcast schedules. Secondly, Router nodes, which provide upward and downward packet forwarding (within a network). A Router also provides services for relaying security and address management

protocols. Lastly, Leaf nodes provide minimum capabilities: discovering and joining a network, send/receive IPv6 packets, etc. A low power network may contain a mesh topology with Routers at the edges that construct a star topology with Leaf nodes.

The FAN profile is based on various open standards developed by the IETF (including [RFC0768], [RFC2460], [RFC4443] and [RFC6282]), IEEE802 (including [IEEE-802-15-4] and [IEEE-802-15-9]) and ANSI/TIA [ANSI-4957-210] for low power and lossy networks.

The FAN profile specification provides an application-independent IPv6-based transport service. There are two possible methods for establishing the IPv6 packet routing: Routing Protocol for Low-Power and Lossy Networks (RPL) at the Network layer is mandatory, and Multi-Hop Delivery Service (MHDS) is optional at the Data Link layer. Table 5 provides an overview of the FAN network stack.

The Transport service is based on User Datagram Protocol (UDP) defined in RFC768 or Transmission Control Protocol (TCP) defined in RFC793.

The Network service is provided by IPv6 as defined in RFC2460 with 6LoWPAN adaptation as defined in RFC4944 and RFC6282. ICMPv6, as defined in RFC4443, is used for the control plane during information exchange.

The Data Link service provides both control/management of the Physical layer and data transfer/management services to the Network layer. These services are divided into Media Access Control (MAC) and Logical Link Control (LLC) sub-layers. The LLC sub-layer provides a protocol dispatch service which supports 6LoWPAN and an optional MAC sub-layer mesh service. The MAC sub-layer is constructed using data structures defined in IEEE802.15.4-2015. Multiple modes of frequency hopping are defined. The entire MAC payload is encapsulated in an IEEE802.15.9 Information Element to enable LLC protocol dispatch between upper layer 6LoWPAN processing, MAC sublayer mesh processing, etc. These areas will be expanded once IEEE802.15.12 is completed.

The PHY service is derived from a sub-set of the SUN FSK specification in IEEE802.15.4-2015. The 2-FSK modulation schemes, with channel spacing range from 200 to 600 kHz, are defined to provide data rates from 50 to 300 kbps, with Forward Error Coding (FEC) as an optional feature. Towards enabling ultra-low-power applications, the PHY layer design is also extendable to low energy and critical infrastructure monitoring networks.

Layer	Description
IPv6 protocol suite	TCP/UDP 6LoWPAN Adaptation + Header Compression DHCPv6 for IP address management. Routing using RPL. ICMPv6. Unicast and Multicast forwarding.
MAC based on IEEE 802.15.4e + IE extensions	Frequency hopping Discovery and Join Protocol Dispatch (IEEE 802.15.9) Several Frame Exchange patterns Optional Mesh Under routing (ANSI 4957.210).
PHY based on 802.15.4g	Various data rates and regions
Security	802.1X/EAP-TLS/PKI Authentication. TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 required for EAP-TLS. 802.11i Group Key Management Frame security is implemented as AES-CCM* as specified in IEEE 802.15.4 Optional ETSI-TS-102-887-2 Node 2 Node Key Management

Table 5: Wi-SUN Stack Overview

The FAN security supports Data Link layer network access control, mutual authentication, and establishment of a secure pairwise link

between a FAN node and its Border Router, which is implemented with an adaptation of IEEE802.1X and EAP-TLS as described in [RFC5216] using secure device identity as described in IEEE802.1AR. Certificate formats are based upon [RFC5280]. A secure group link between a Border Router and a set of FAN nodes is established using an adaptation of the IEEE802.11 Four-Way Handshake. A set of 4 group keys are maintained within the network, one of which is the current transmit key. Secure node to node links are supported between one-hop FAN neighbors using an adaptation of ETSI-TS-102-887-2. FAN nodes implement Frame Security as specified in IEEE802.15.4-2015.

3. Generic Terminology

LPWAN technologies, such as those discussed above, have similar architectures but different terminology. We can identify different types of entities in a typical LPWAN network:

- o End-Devices are the devices or the "things" (e.g. sensors, actuators, etc.); they are named differently in each technology (End Device, User Equipment or End Point). There can be a high density of end devices per radio gateway.
- o The Radio Gateway, which is the end point of the constrained link. It is known as: Gateway, Evolved Node B or Base station.
- o The Network Gateway or Router is the interconnection node between the Radio Gateway and the Internet. It is known as: Network Server, Serving GW or Service Center.
- o LPWAN-AAA Server, which controls the user authentication, the applications. It is known as: Join-Server, Home Subscriber Server or Registration Authority. (We use the term LPWAN-AAA server because we're not assuming that this entity speaks RADIUS or Diameter as many/most AAA servers do, but equally we don't want to rule that out, as the functionality will be similar.
- o At last we have the Application Server, known also as Packet Data Node Gateway or Network Application.

Function/ Technology	LORAWAN	NB-IOT	SIGFOX	Wi-SUN	IETF
Sensor, Actuator, device, object	End Device	User Equipment	End Point	Leaf Node	Device (Dev)
Transceiver Antenna	Gateway	Evolved Node B	Base Station	Router Node	RADIO Gateway
Server	Network Server	PDN GW/ SCEF	Service Center	Border Router	Network Gateway (NGW)
Security Server	Join Server	Home Subscriber Server	Registration Authority	Authent. Server	LPWAN- AAA SERVER
Application	Application Server	Application Server	Network Application	Appli- cation	Application (App)

Figure 8: LPWAN Architecture Terminology

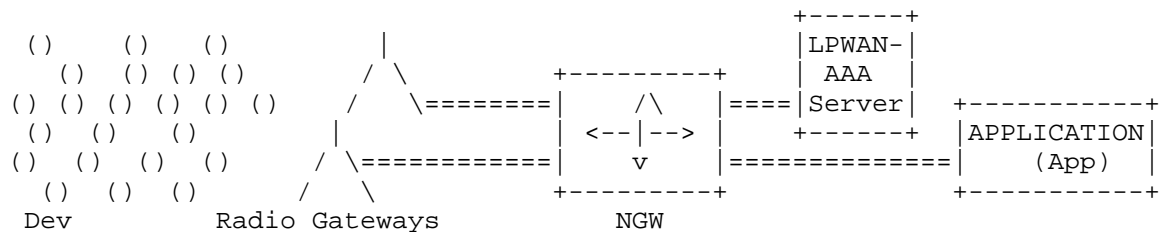


Figure 9: LPWAN Architecture

In addition to the names of entities, LPWANs are also subject to possibly regional frequency band regulations. Those may include restrictions on the duty-cycle, for example requiring that hosts only transmit for a certain percentage of each hour.

4. Gap Analysis

This section considers some of the gaps between current LPWAN technologies and the goals of the LPWAN working group. Many of the generic considerations described in [RFC7452] will also apply in LPWANs, as end-devices can also be considered as a subclass of (so-

called) "smart objects." In addition, LPWAN device implementers will also need to consider the issues relating to firmware updates described in [RFC8240].

4.1. Naive application of IPv6

IPv6 [RFC8200] has been designed to allocate addresses to all the nodes connected to the Internet. Nevertheless, the header overhead of at least 40 bytes introduced by the protocol is incompatible with LPWAN constraints. If IPv6 with no further optimization were used, several LPWAN frames could be needed just to carry the IP header. Another problem arises from IPv6 MTU requirements, which require the layer below to support at least 1280 byte packets [RFC2460].

IPv6 has a configuration protocol - neighbor discovery protocol, (NDP) [RFC4861]). For a node to learn network parameters NDP generates regular traffic with a relatively large message size that does not fit LPWAN constraints.

In some LPWAN technologies, layer two multicast is not supported. In that case, if the network topology is a star, the solution and considerations of section 3.2.5 of [RFC7668] may be applied.

Other key protocols such as DHCPv6 [RFC3315], IPsec [RFC4301] and TLS [RFC5246] have similarly problematic properties in this context. Each of those require relatively frequent round-trips between the host and some other host on the network. In the case of cryptographic protocols such as IPsec and TLS, in addition to the round-trips required for secure session establishment, cryptographic operations can require padding and addition of authenticators that are problematic when considering LPWAN lower layers. Note that mains powered Wi-SUN mesh router nodes will typically be more resource capable than the other LPWAN techs discussed. This can enable use of more "chatty" protocols for some aspects of Wi-SUN.

4.2. 6LoWPAN

Several technologies that exhibit significant constraints in various dimensions have exploited the 6LoWPAN suite of specifications [RFC4944], [RFC6282], [RFC6775] to support IPv6 [I-D.hong-6lo-use-cases]. However, the constraints of LPWANs, often more extreme than those typical of technologies that have (re)used 6LoWPAN, constitute a challenge for the 6LoWPAN suite in order to enable IPv6 over LPWAN. LPWANs are characterized by device constraints (in terms of processing capacity, memory, and energy availability), and specially, link constraints, such as:

- o tiny layer two payload size (from ~10 to ~100 bytes),

- o very low bit rate (from ~10 bit/s to ~100 kbit/s), and
- o in some specific technologies, further message rate constraints (e.g. between ~0.1 message/minute and ~1 message/minute) due to regional regulations that limit the duty cycle.

4.2.1. Header Compression

6LoWPAN header compression reduces IPv6 (and UDP) header overhead by eliding header fields when they can be derived from the link layer, and by assuming that some of the header fields will frequently carry expected values. 6LoWPAN provides both stateless and stateful header compression. In the latter, all nodes of a 6LoWPAN are assumed to share compression context. In the best case, the IPv6 header for link-local communication can be reduced to only 2 bytes. For global communication, the IPv6 header may be compressed down to 3 bytes in the most extreme case. However, in more practical situations, the smallest IPv6 header size may be 11 bytes (one address prefix compressed) or 19 bytes (both source and destination prefixes compressed). These headers are large considering the link layer payload size of LPWAN technologies, and in some cases are even bigger than the LPWAN PDUs. 6LoWPAN has been initially designed for IEEE 802.15.4 networks with a frame size up to 127 bytes and a throughput of up to 250 kb/s, which may or may not be duty-cycled.

4.2.2. Address Autoconfiguration

Traditionally, Interface Identifiers (IIDs) have been derived from link layer identifiers [RFC4944]. This allows optimizations such as header compression. Nevertheless, recent guidance has given advice on the fact that, due to privacy concerns, 6LoWPAN devices should not be configured to embed their link layer addresses in the IID by default. [RFC8065] provides guidance on better methods for generating IIDs.

4.2.3. Fragmentation

As stated above, IPv6 requires the layer below to support an MTU of 1280 bytes [RFC2460]. Therefore, given the low maximum payload size of LPWAN technologies, fragmentation is needed.

If a layer of an LPWAN technology supports fragmentation, proper analysis has to be carried out to decide whether the fragmentation functionality provided by the lower layer or fragmentation at the adaptation layer should be used. Otherwise, fragmentation functionality shall be used at the adaptation layer.

6LoWPAN defined a fragmentation mechanism and a fragmentation header to support the transmission of IPv6 packets over IEEE 802.15.4 networks [RFC4944]. While the 6LoWPAN fragmentation header is appropriate for IEEE 802.15.4-2003 (which has a frame payload size of 81-102 bytes), it is not suitable for several LPWAN technologies, many of which have a maximum payload size that is one order of magnitude below that of IEEE 802.15.4-2003. The overhead of the 6LoWPAN fragmentation header is high, considering the reduced payload size of LPWAN technologies and the limited energy availability of the devices using such technologies. Furthermore, its datagram offset field is expressed in increments of eight octets. In some LPWAN technologies, the 6LoWPAN fragmentation header plus eight octets from the original datagram exceeds the available space in the layer two payload. In addition, the MTU in the LPWAN networks could be variable which implies a variable fragmentation solution.

4.2.4. Neighbor Discovery

6LoWPAN Neighbor Discovery [RFC6775] defined optimizations to IPv6 Neighbor Discovery [RFC4861], in order to adapt functionality of the latter for networks of devices using IEEE 802.15.4 or similar technologies. The optimizations comprise host-initiated interactions to allow for sleeping hosts, replacement of multicast-based address resolution for hosts by an address registration mechanism, multihop extensions for prefix distribution and duplicate address detection (note that these are not needed in a star topology network), and support for 6LoWPAN header compression.

6LoWPAN Neighbor Discovery may be used in not so severely constrained LPWAN networks. The relative overhead incurred will depend on the LPWAN technology used (and on its configuration, if appropriate). In certain LPWAN setups (with a maximum payload size above ~60 bytes, and duty-cycle-free or equivalent operation), an RS/RA/NS/NA exchange may be completed in a few seconds, without incurring packet fragmentation.

In other LPWANs (with a maximum payload size of ~10 bytes, and a message rate of ~0.1 message/minute), the same exchange may take hours or even days, leading to severe fragmentation and consuming a significant amount of the available network resources. 6LoWPAN Neighbor Discovery behavior may be tuned through the use of appropriate values for the default Router Lifetime, the Valid Lifetime in the PIOs, and the Valid Lifetime in the 6LoWPAN Context Option (6CO), as well as the address Registration Lifetime. However, for the latter LPWANs mentioned above, 6LoWPAN Neighbor Discovery is not suitable.

4.3. 6lo

The 6lo WG has been reusing and adapting 6LoWPAN to enable IPv6 support over link layer technologies such as Bluetooth Low Energy (BTLE), ITU-T G.9959, DECT-ULE, MS/TP-RS485, NFC IEEE 802.11ah. (See <<https://tools.ietf.org/wg/6lo>> for details.) These technologies are similar in several aspects to IEEE 802.15.4, which was the original 6LoWPAN target technology.

6lo has mostly used the subset of 6LoWPAN techniques best suited for each lower layer technology, and has provided additional optimizations for technologies where the star topology is used, such as BTLE or DECT-ULE.

The main constraint in these networks comes from the nature of the devices (constrained devices), whereas in LPWANs it is the network itself that imposes the most stringent constraints.

4.4. 6tisch

The 6tisch solution is dedicated to mesh networks that operate using 802.15.4e MAC with a deterministic slotted channel. The time slot channel (TSCH) can help to reduce collisions and to enable a better balance over the channels. It improves the battery life by avoiding the idle listening time for the return channel.

A key element of 6tisch is the use of synchronization to enable determinism. TSCH and 6TiSCH may provide a standard scheduling function. The LPWAN networks probably will not support synchronization like the one used in 6tisch.

4.5. RoHC

Robust header compression (RoHC) is a header compression mechanism [RFC3095] developed for multimedia flows in a point to point channel. RoHC uses 3 levels of compression, each level having its own header format. In the first level, RoHC sends 52 bytes of header, in the second level the header could be from 34 to 15 bytes and in the third level header size could be from 7 to 2 bytes. The level of compression is managed by a sequence number, which varies in size from 2 bytes to 4 bits in the minimal compression. SN compression is done with an algorithm called W-LSB (Window- Least Significant Bits). This window has a 4-bit size representing 15 packets, so every 15 packets RoHC needs to slide the window in order to receive the correct sequence number, and sliding the window implies a reduction of the level of compression. When packets are lost or errored, the decompressor loses context and drops packets until a bigger header is sent with more complete information. To estimate the performance of

RoHC, an average header size is used. This average depends on the transmission conditions, but most of the time is between 3 and 4 bytes.

RoHC has not been adapted specifically to the constrained hosts and networks of LPWANs: it does not take into account energy limitations nor the transmission rate, and RoHC context is synchronised during transmission, which does not allow better compression.

4.6. ROLL

Most technologies considered by the lpwan WG are based on a star topology, which eliminates the need for routing at that layer. Future work may address additional use-cases that may require adaptation of existing routing protocols or the definition of new ones. As of the time of writing, work similar to that done in the ROLL WG and other routing protocols are out of scope of the LPWAN WG.

4.7. CoAP

CoAP [RFC7252] provides a RESTful framework for applications intended to run on constrained IP networks. It may be necessary to adapt CoAP or related protocols to take into account for the extreme duty cycles and the potentially extremely limited throughput of LPWANs.

For example, some of the timers in CoAP may need to be redefined. Taking into account CoAP acknowledgments may allow the reduction of L2 acknowledgments. On the other hand, the current work in progress in the CoRE WG where the COMI/CoOL network management interface which, uses Structured Identifiers (SID) to reduce payload size over CoAP may prove to be a good solution for the LPWAN technologies. The overhead is reduced by adding a dictionary which matches a URI to a small identifier and a compact mapping of the YANG model into the CBOR binary representation.

4.8. Mobility

LPWAN nodes can be mobile. However, LPWAN mobility is different from the one specified for Mobile IP. LPWAN implies sporadic traffic and will rarely be used for high-frequency, real-time communications. The applications do not generate a flow, they need to save energy and most of the time the node will be down.

In addition, LPWAN mobility may mostly apply to groups of devices, that represent a network in which case mobility is more a concern for the gateway than the devices. NEMO [RFC3963] Mobility or other mobile gateway solutions (such as a gateway with an LTE uplink) may be used in the case where some end-devices belonging to the same

network gateway move from one point to another such that they are not aware of being mobile.

4.9. DNS and LPWAN

The Domain Name System (DNS) [RFC1035], enables applications to name things with a globally resolvable name. Many protocols use the DNS to identify hosts, for example applications using CoAP.

The DNS query/answer protocol as a pre-cursor to other communication within the time-to-live (TTL) of a DNS answer is clearly problematic in an LPWAN, say where only one round-trip per hour can be used, and with a TTL that is less than 3600. It is currently unclear whether and how DNS-like functionality might be provided in LPWANs.

5. Security Considerations

Most LPWAN technologies integrate some authentication or encryption mechanisms that were defined outside the IETF. The working group may need to do work to integrate these mechanisms to unify management. A standardized Authentication, Accounting, and Authorization (AAA) infrastructure [RFC2904] may offer a scalable solution for some of the security and management issues for LPWANs. AAA offers centralized management that may be of use in LPWANs, for example [I-D.garcia-dime-diameter-lorawan] and [I-D.garcia-radext-radius-lorawan] suggest possible security processes for a LoRaWAN network. Similar mechanisms may be useful to explore for other LPWAN technologies.

Some applications using LPWANs may raise few or no privacy considerations. For example, temperature sensors in a large office building may not raise privacy issues. However, the same sensors, if deployed in a home environment and especially if triggered due to human presence, can raise significant privacy issues - if an end-device emits (an encrypted) packet every time someone enters a room in a home, then that traffic is privacy sensitive. And the more that the existence of that traffic is visible to network entities, the more privacy sensitivities arise. At this point, it is not clear whether there are workable mitigations for problems like this - in a more typical network, one would consider defining padding mechanisms and allowing for cover traffic. In some LPWANs, those mechanisms may not be feasible. Nonetheless, the privacy challenges do exist and can be real and so some solutions will be needed. Note that many aspects of solutions in this space may not be visible in IETF specifications, but can be e.g. implementation or deployment specific.

Another challenge for LPWANs will be how to handle key management and associated protocols. In a more traditional network (e.g. the web), servers can "staple" Online Certificate Status Protocol (OCSP) responses in order to allow browsers to check revocation status for presented certificates. [RFC6961] While the stapling approach is likely something that would help in an LPWAN, as it avoids an RTT, certificates and OCSP responses are bulky items and will prove challenging to handle in LPWANs with bounded bandwidth.

6. IANA Considerations

There are no IANA considerations related to this memo.

7. Contributors

[[RFC editor: Please fix names below for I18N.]]

As stated above this document is mainly a collection of content developed by the full set of contributors listed below. The main input documents and their authors were:

- o Text for Section 2.1 was provided by Alper Yegin and Stephen Farrell in [I-D.farrell-lpwan-lora-overview].
- o Text for Section 2.2 was provided by Antti Ratilainen in [I-D.ratilainen-lpwan-nb-iot].
- o Text for Section 2.3 was provided by Juan Carlos Zuniga and Benoit Ponsard in [I-D.zuniga-lpwan-sigfox-system-description].
- o Text for Section 2.4 was provided via personal communication from Bob Heile (bheile@ieee.org) and was authored by Bob and Sum Chin Sean. There is no Internet draft for that at present.
- o Text for Section 4 was provided by Ana Minabiru, Carles Gomez, Laurent Toutain, Josep Paradells and Jon Crowcroft in [I-D.minaburo-lpwan-gap-analysis]. Additional text from that draft is also used elsewhere above.

The full list of contributors are:

Jon Crowcroft
University of Cambridge
JJ Thomson Avenue
Cambridge, CB3 0FD
United Kingdom

Email: jon.crowcroft@cl.cam.ac.uk

Carles Gomez
UPC/i2CAT
C/Esteve Terradas, 7
Castelldefels 08860
Spain

Email: carlesgo@entel.upc.edu

Bob Heile
Wi-Sun Alliance
11 Robert Toner Blvd, Suite 5-301
North Attleboro, MA 02763
USA

Phone: +1-781-929-4832
Email: bheile@ieee.org

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

Email: ana@ackl.io

Josep PARadells
UPC/i2CAT
C/Jordi Girona, 1-3
Barcelona 08034
Spain

Email: josep.paradells@entel.upc.edu

Charles E. Perkins
Futurewei
2330 Central Expressway
Santa Clara 95050
Unites States

Email: charliep@computer.org

Benoit Ponsard
SIGFOX
425 rue Jean Rostand
Labège 31670
France

Email: Benoit.Ponsard@sigfox.com
URI: <http://www.sigfox.com/>

Antti Ratilainen
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: antti.ratilainen@ericsson.com

Chin-Sean SUM
Wi-Sun Alliance
20, Science Park Rd
Singapore 117674

Phone: +65 6771 1011
Email: sum@wi-sun.org

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

Email: Laurent.Toutain@telecom-bretagne.eu

Alper Yegin
Actility
Paris, Paris
FR

Email: alper.yegin@actility.com

Juan Carlos Zuniga
SIGFOX

425 rue Jean Rostand
Labège 31670
France

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

8. Acknowledgments

Thanks to all those listed in Section 7 for the excellent text.
Errors in the handling of that are solely the editor's fault.

[[RFC editor: Please fix names below for I18N, at least Mirja's does
need fixing.]]

In addition to the contributors above, thanks are due to (in
alphabetical order): Abdussalam Baryun, Andy Malis, Arun
(arun@acklio.com), Behcet SariKaya, Dan Garcia Carrillo, Jiazi Yi,
Mirja Kuehlewind, Paul Duffy, Russ Housley, Samita Chakrabarti, Thad
Guidry, Warren Kumari, for comments.

Alexander Pelov and Pascal Thubert were the LPWAN WG chairs while
this document was developed.

Stephen Farrell's work on this memo was supported by Pervasive
Nation, the Science Foundation Ireland's CONNECT centre national IoT
network. <<https://connectcentre.ie/pervasive-nation/>>

9. Informative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and
specification", STD 13, RFC 1035, DOI 10.17487/RFC1035,
November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6
(IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460,
December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L.,
Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and
D. Spence, "AAA Authorization Framework", RFC 2904,
DOI 10.17487/RFC2904, August 2000, <<https://www.rfc-editor.org/info/rfc2904>>.

- [RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, DOI 10.17487/RFC3095, July 2001, <<https://www.rfc-editor.org/info/rfc3095>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<https://www.rfc-editor.org/info/rfc3315>>.
- [RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, DOI 10.17487/RFC3963, January 2005, <<https://www.rfc-editor.org/info/rfc3963>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [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>>.
- [RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216, March 2008, <<https://www.rfc-editor.org/info/rfc5216>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.

- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [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>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<https://www.rfc-editor.org/info/rfc6775>>.
- [RFC6961] Pettersen, Y., "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension", RFC 6961, DOI 10.17487/RFC6961, June 2013, <<https://www.rfc-editor.org/info/rfc6961>>.
- [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>>.
- [RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <<https://www.rfc-editor.org/info/rfc7452>>.
- [RFC7668] Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "IPv6 over BLUETOOTH(R) Low Energy", RFC 7668, DOI 10.17487/RFC7668, October 2015, <<https://www.rfc-editor.org/info/rfc7668>>.
- [RFC8065] Thaler, D., "Privacy Considerations for IPv6 Adaptation-Layer Mechanisms", RFC 8065, DOI 10.17487/RFC8065, February 2017, <<https://www.rfc-editor.org/info/rfc8065>>.
- [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>>.

- [RFC8240] Tschafenig, H. and S. Farrell, "Report from the Internet of Things Software Update (IoTSU) Workshop 2016", RFC 8240, DOI 10.17487/RFC8240, September 2017, <<https://www.rfc-editor.org/info/rfc8240>>.
- [I-D.farrell-lpwan-lora-overview]
Farrell, S. and A. Yegin, "LoRaWAN Overview", draft-farrell-lpwan-lora-overview-01 (work in progress), October 2016.
- [I-D.minaburo-lpwan-gap-analysis]
Minaburo, A., Gomez, C., Toutain, L., Paradells, J., and J. Crowcroft, "LPWAN Survey and GAP Analysis", draft-minaburo-lpwan-gap-analysis-02 (work in progress), October 2016.
- [I-D.zuniga-lpwan-sigfox-system-description]
Zuniga, J. and B. PONSARD, "SIGFOX System Description", draft-zuniga-lpwan-sigfox-system-description-04 (work in progress), December 2017.
- [I-D.ratilainen-lpwan-nb-iot]
Ratilainen, A., "NB-IoT characteristics", draft-ratilainen-lpwan-nb-iot-00 (work in progress), July 2016.
- [I-D.garcia-dime-diameter-lorawan]
Garcia, D., Lopez, R., Kandasamy, A., and A. Pelov, "LoRaWAN Authentication in Diameter", draft-garcia-dime-diameter-lorawan-00 (work in progress), May 2016.
- [I-D.garcia-radext-radius-lorawan]
Garcia, D., Lopez, R., Kandasamy, A., and A. Pelov, "LoRaWAN Authentication in RADIUS", draft-garcia-radext-radius-lorawan-03 (work in progress), May 2017.
- [I-D.hong-6lo-use-cases]
Hong, Y. and C. Gomez, "IPv6 over Constrained Node Networks(6lo) Applicability & Use cases", draft-hong-6lo-use-cases-03 (work in progress), October 2016.
- [TGPP36300]
3GPP, "TS 36.300 v13.4.0 Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2", 2016, <http://www.3gpp.org/ftp/Specs/2016-09/Rel-14/36_series/>.

- [TGPP36321]
3GPP, "TS 36.321 v13.2.0 Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification", 2016.
- [TGPP36322]
3GPP, "TS 36.322 v13.2.0 Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification", 2016.
- [TGPP36323]
3GPP, "TS 36.323 v13.2.0 Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) specification (Not yet available)", 2016.
- [TGPP36331]
3GPP, "TS 36.331 v13.2.0 Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification", 2016.
- [TGPP36201]
3GPP, "TS 36.201 v13.2.0 - Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description", 2016.
- [TGPP23720]
3GPP, "TR 23.720 v13.0.0 - Study on architecture enhancements for Cellular Internet of Things", 2016.
- [TGPP33203]
3GPP, "TS 33.203 v13.1.0 - 3G security; Access security for IP-based services", 2016.
- [fcc_ref] "FCC CFR 47 Part 15.247 Telecommunication Radio Frequency Devices - Operation within the bands 902-928 MHz, 2400-2483.5 MHz, and 5725-5850 MHz.", June 2016.
- [etsi_ref]
"ETSI EN 300-220 (Parts 1 and 2): Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mW", May 2016.
- [arib_ref]
"ARIB STD-T108 (Version 1.0): 920MHz-Band Telemeter, Telecontrol and data transmission radio equipment.", February 2012.

[LoRaSpec]

LoRa Alliance, "LoRaWAN Specification Version V1.0.2", July 2016, <http://portal.lora-alliance.org/DesktopModules/Inventures_Document/FileDownload.aspx?ContentID=1398>.

[ANSI-4957-000]

ANSI, TIA-4957.000, "Architecture Overview for the Smart Utility Network", May 2013, <https://global.ihs.com/doc_detail.cfm?%26rid=TIA%26item_s_key=00606368>.

[ANSI-4957-210]

ANSI, TIA-4957.210, "Multi-Hop Delivery Specification of a Data Link Sub-Layer", May 2013, <https://global.ihs.com/doc_detail.cfm?%26csf=TIA%26item_s_key=00601800>.

[wisun-pressie1]

Phil Beecher, Chair, Wi-SUN Alliance, "Wi-SUN Alliance Overview", March 2017, <<http://indiasmartgrid.org/event2017/10-03-2017/4.%20Roundtable%20on%20Communication%20and%20Cyber%20Security/1.%20Phil%20Beecher.pdf>>.

[wisun-pressie2]

Bob Heile, Director of Standards, Wi-SUN Alliance, "IETF97 Wi-SUN Alliance Field Area Network (FAN) Overview", November 2016, <<https://www.ietf.org/proceedings/97/slides/slides-97-lpwan-35-wi-sun-presentation-00.pdf>>.

[IEEE-802-15-4]

"IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs)", IEEE Standard 802.15.4, 2015, <<https://standards.ieee.org/findstds/standard/802.15.4-2015.html>>.

[IEEE-802-15-9]

"IEEE Recommended Practice for Transport of Key Management Protocol (KMP) Datagrams", IEEE Standard 802.15.9, 2016, <<https://standards.ieee.org/findstds/standard/802.15.9-2016.html>>.

[etsi_unb]

"ETSI TR 103 435 System Reference document (SRdoc); Short Range Devices (SRD); Technical characteristics for Ultra Narrow Band (UNB) SRDs operating in the UHF spectrum below 1 GHz", February 2017.

[nbiot-ov]

Beyene, Yihenew Dagne, et al., "NB-IoT technology overview and experience from cloud-RAN implementation", IEEE Wireless Communications 24,3 (2017): 26-32, June 2017.

Appendix A. Changes

[[RFC editor: Please remove this before publication]]

A.1. From -00 to -01

- o WG have stated they want this to be an RFC.
- o WG clearly want to keep the RF details.
- o Various changes made to remove/resolve a number of editorial notes from -00 (in some cases as per suggestions from Ana Minaburo)
- o Merged PR's: #1...
- o Rejected PR's: #2 (change was made to .txt not .xml but was replicated manually by editor)
- o Github repo is at: <https://github.com/sftcd/lpwan-ov>

A.2. From -01 to -02

- o WG seem to agree with editor suggestions in slides 13-24 of the presentation on this topic given at IETF98 (See: <https://www.ietf.org/proceedings/98/slides/slides-98-lpwan-aggregated-slides-07.pdf>)
- o Got new text wrt Wi-SUN via email from Paul Duffy and merged that in
- o Reflected list discussion wrt terminology and "end-device"
- o Merged PR's: #3...

A.3. From -02 to -03

- o Editorial changes and typo fixes thanks to Fred Baker running something called Grammarly and sending me it's report.
- o Merged PR's: #4, #6, #7...
- o Editor did an editing pass on the lot.

A.4. From -03 to -04

- o Picked up a PR that had been wrongly applied that expands UE
- o Editorial changes wrt LoRa suggested by Alper
- o Editorial changes wrt SIGFOX provided by Juan-Carlos

A.5. From -04 to -05

- o Handled Russ Housley's WGLC review.
- o Handled Alper Yegin's WGLC review.

A.6. From -05 to -06

- o More Alper comments:-)
- o Added some more detail about sigfox security.
- o Added Wi-SUN changes from Charlie Perkins

A.7. From -06 to -07

Yet more Alper comments:-)

Comments from Behcet Sarikaya

A.8. From -07 to -08

various typos

Last call and directorate comments from Abdussalam Baryun (AB) and Andy Malis

20180118 IESG ballot comments from Warren: nits handled, two possible bits of text still needed.

Some more AB comments handled. Still need to check over 7452 and 8240 to see if issues from those need to be discussed here.

Corrected "no IP capabilities - Wi-SUN devices do v6 (thanks Paul Duffy:-)

Mirja's AD ballot comments handled.

Added a sentence in intro trying to say what's "special" about LPWAN compared to other constrained networks. (As suggested by Warren.)

Added text @ start of gap analysis referring to RFCs 7252 and 8240, as suggested by a few folks (AB, Warren, Mirja)

Added nbiot-ov reference for those who'd like a more polished presentation of NB-IoT

A.9. From -08 to -09

Changes due to IoT-DIR review from Samita Chakrabarti: fixed error on max rate between tables 1 and 2; s/eNb/eNodeB/; fixed references to hong-6lo-use-cases; added RFC8065 reference

A.10. From -09 to -10

Added Charlie Perkins as contributor - was supposed to have been done ages ago - editor forgot;-)

Author's Address

Stephen Farrell (editor)
Trinity College Dublin
Dublin 2
Ireland

Phone: +353-1-896-2354
Email: stephen.farrell@cs.tcd.ie