lpwan Working Group
Internet-Draft
Intended status: Informational
Expires: May 1, 2018

D. Barthel
Orange SA
L. Toutain
Institut MINES TELECOM; IMT Atlantique
A. Kandasamy
Acklio

October 28, 2017

LPWAN Static Context Header Compression (SCHC) for ICMPv6
draft-barthel-icmpv6-schc-00

Abstract

ICMPv6 is a companion protocol to IPv6. It defines messages that inform the source of IPv6 packets of errors during packet delivery. It also defines the Echo Request/Reply messages that are used for basic network troubleshooting (ping command). ICMPv6 messages are transported on IPv6.

This document describes how to adapt ICMPv6 to Low Power Wide Area Networks (LPWANs) by compressing ICMPv6/IPv6 headers and by protecting the LPWAN network and the Device from undesirable ICMPv6 traffic.

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, 2018.

Copyright Notice

Table of Contents

1. Introduction

ICMPv6 [RFC4443] is a companion protocol to IPv6 [RFC8200].

[RFC4443] defines a generic message format. This format is used for messages to be sent back to the source of an IPv6 packet to inform it about errors during packet delivery.

More specifically, [RFC4443] defines 4 error messages: Destination Unreachable, Packet Too Big, Time Exceeded and Parameter Problem.

[RFC4443] also defines the Echo Request and Echo Reply messages, which provide support for the ping application.

Other ICMPv6 messages are defined in other RFCs, such as an extended format of the same messages [RFC4884] and other messages used by the Neighbor Discovery Protocol [RFC4861].

This document focuses on using Static Context Header Compression (SCHC) to compress [RFC4443] messages that need to be transmitted over the LPWAN network, and on having the LPWAN gateway proxying the Device to save it the unwanted traffic.

LPWANs' salient characteristics are described in [I-D.ietf-lpwan-overview]

2. Terminology

This draft re-uses the Terminology defined in [I-D.ietf-lpwan-ipv6-static-context-hc].

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. Use cases

In the LPWAN architecture, we can distinguish the following cases:

- the Device is the (purported) source of an ICMP error message, mainly in response to an incorrect incoming IPv6 message, or in response to a ping request. In this case, as much as possible, the core SCHC C/D should act as a proxy and originate the ICMP message, so that the Device and the LPWAN network are protected from this unwanted traffic.

- the Device is the destination of the ICMP message, mainly in response to a packet sent by the Device to the network that generates an error. In this case, we want the ICMP message to reach the Device, and this document describes in section Section 4.2.1 what SCHC compression should be applied.

- the Device is the originator of an Echo Request message, and therefore the destination of the Echo Reply message.

- the Device is the destination of an Echo Request message, and therefore the purported source of an Echo Reply message.

These cases are further described in Section 4.

4. Detailed behavior

4.1. Device is the source of an ICMPv6 error message

As stated in [RFC4443], a node should generate an ICMPv6 message in response to an IPv6 packet that is malformed or which cannot be processed due to some incorrect field value.

The general intent of this document is to spare both the Device and the LPWAN network this un-necessary traffic. The incorrect packets should be caught at the core SCHC C/D and the ICMPv6 notification should be sent back from there.

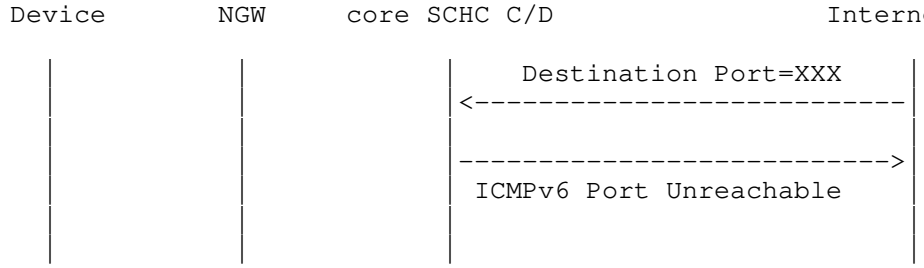


Figure 1: Example of ICMPv6 error message sent back to the Internet

Figure 1 shows an example of an IPv6 packet trying to reach a Device. Let's assume that the port number used as destination port is not "known" (needs better definition) from the core SCHC C/D. Instead of sending the packet over the LPWAN and having this packet rejected by the Device, the core SCHC C/D issues an ICMPv6 error message "Destination Unreachable" (Type 1) with Code 1 ("Port Unreachable") on behalf of the Device.

TODO: This assumes that all ports that the Device listens to will be matched by a SCHC rule. Is this the basic assumption of SCHC that all packets that do not match a rule are rejected? If yes, why do have fragmentation also for uncompressed packets?

TODO: discuss the various Type/Code that are expected to be generated in response to various errors.

4.2. Device is the destination of an ICMPv6 error message

In this situation, we assume that a Device has been configured to send information to a server on the Internet. If this server becomes no longer accessible, an ICMPv6 message will be generated back

towards the Device by an intermediate router. This information can be useful to the Device, for example for reducing the reporting rate in case of periodic reporting of data. Therefore, we compress the ICMPv6 message using SCHC and forward it to the Device over the LPWAN.

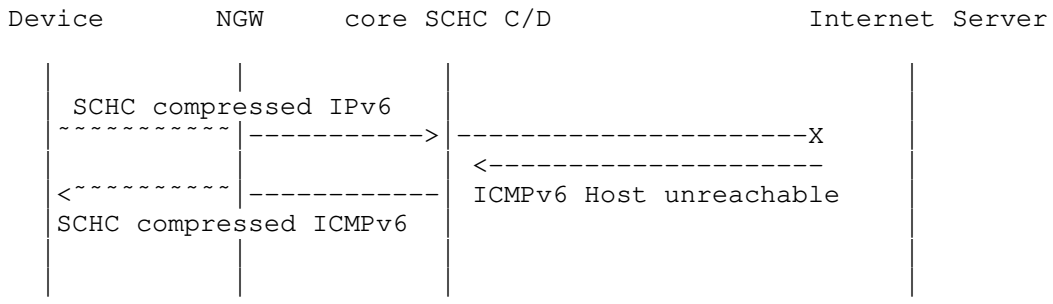


Figure 2: Example of ICMPv6 error message sent back to the Device

Figure 2 illustrates this behavior. The ICMPv6 error message is compressed as described in Section 4.2.1 and forwarded over the LPWAN to the Device.

4.2.1. ICMPv6 error message compression.

The ICMPv6 error messages defined in [RFC4443] contain the fields shown in Figure 3.

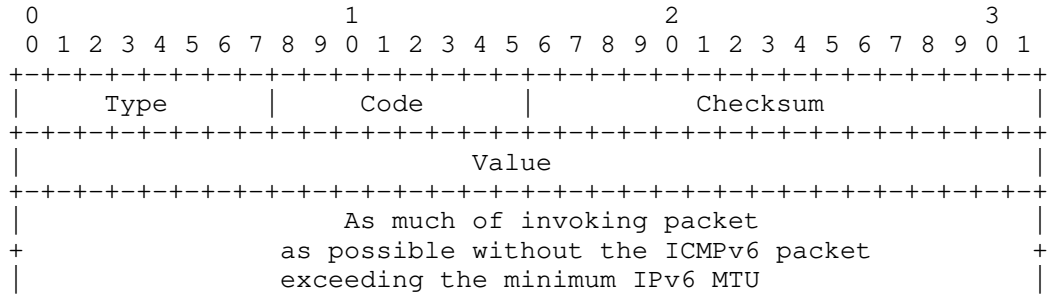


Figure 3: ICMPv6 Error Message format

[RFC4443] states that Type can take the values 1 to 4, and Code can be set to values between 0 and 6. Value is unused for the Destination Unreachable and Time Exceeded messages. It contains the MTU for the Packet Too Big message and a pointer to the byte causing

the error for the Parameter Error message. Therefore, Value is never expected to be greater than 1280 in LPWAN networks.

The following generic rule can therefore be used to compress all ICMPv6 error messages as defined today. More specific rules can also be defined to achieve better compression of some error messages.

The Type field can be associated to a matching list [1, 2, 3, 4] and is therefore compressed down to 2 bits. Code can be reduced to 3 bits using the LSB CDA. Value can be sent on 11 bits using the LSB CDA, but if the Device is known to send smaller packets, then the size of this field can be further reduced.

By [RFC4443], the rest of the ICMPv6 message must contain as much as possible of the IPv6 offending (invoking) packet that triggered this ICMPv6 error message. This information is used to try and identify the SCHC rule that was used to decompress the offending IPv6 packet. If the rule can be found then the Rule Id is added at the end of the compressed ICMPv6 message. Otherwise the compressed packet ends with the compressed Value field.

[RFC4443] states that the "ICMPv6 error message MUST include as much of the IPv6 offending (invoking) packet ... as possible". In order to comply with this requirement, if there is enough information in the incoming ICMPv6 message for the core SCHC C/D to identify the rule that has been used to decompress the erroneous IPv6 packet, this Rule Id must be sent in the compressed ICMPv6 message to the Device. TODO: the erroneous IPv6 packet header (not just the Rule Id) should be sent back. This includes the Rule Id and the compression residue. This means the SCHC C/D uses the context backwards (in the reverse direction). How does the Device know it must also use the context backwards?

TODO: how does one know that the "payload" of a compressed-header packet is in fact another compressed header?

4.3. Device does a ping

If a ping request is generated by a Device, then SCHC compression applies.

The format of an ICMPv6 Echo Request message is described in Figure 4, with Type=128 and Code=0.

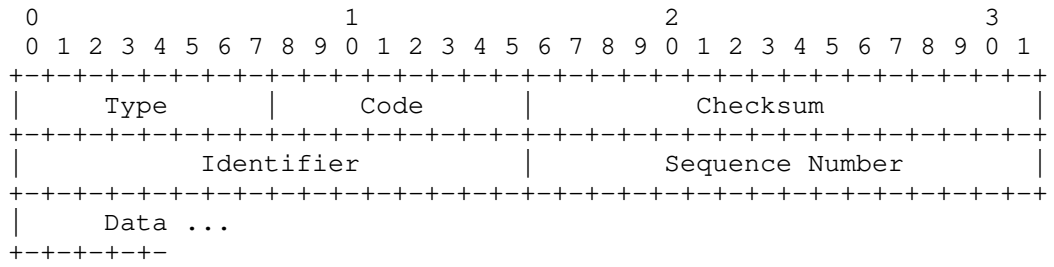


Figure 4: ICMPv6 Echo Request message format

If we assume that one rule will be devoted to compressing Echo Request messages, then Type and Code are known in the rule to be 128 and 0 and can therefore be elided with the not-sent CDA.

Checksum can be reconstructed with the compute-checksum CDA and therefore is not transmitted.

[RFC4443] states that Identifier and Sequence Number are meant to "aid in matching Echo Replies to this Echo Request" and that they "may be zero". Data is "zero or more bytes of arbitrary data".

We recommend that Identifier be zero, Sequence Number be a counter on 3 bits, and Data be zero bytes (absent). Therefore, Identifier is elided with the not-sent CDA, Sequence Number is transmitted on 3 bits with the LSB CDA and no Data is transmitted.

The transmission cost of the Echo Request message is therefore the size of the Rule Id + 3 bits.

When the destination receives the Echo Request message, it will respond back with a Echo Reply message. This message bears the same format as the Echo Request message but with Type = 129 (see Figure 4).

[RFC4443] states that the Identifier, Sequence Number and Data fields of the Echo Reply message shall contain the same values as the invoking Echo Request message. Therefore, a rule shall be used similar to that used for compressing the Echo Request message.

TODO: how about a shared rule for Echo Request and Echo Reply with an LSB(1) CDA on the Type field? Or exploiting the Up/Down direction field in the rule?

4.4. Device is ping'ed

If the Device is ping'ed (i.e., is the destination of an Echo Request message), the default behavior is to avoid propagating the Echo Request message over the LPWAN.

This is the recommended behavior with the Code 0 (default value) of the Echo Request message. In addition, this document defines two other Code values to achieve two other behaviors.

The resulting three behaviors are shown on Figure 5 and described below:

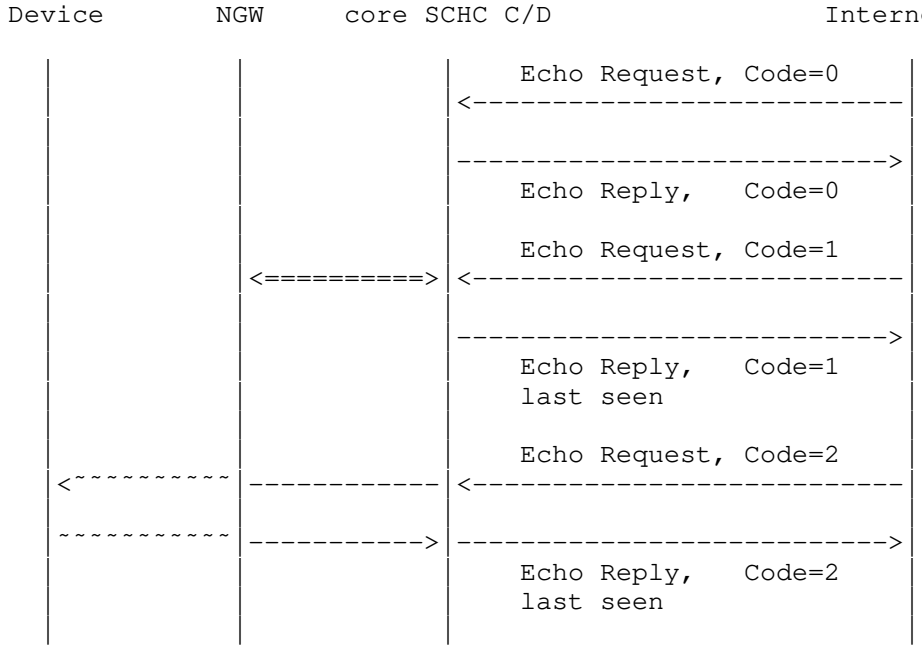


Figure 5: Examples of ICMPv6 Echo Request/Reply

- Code = 0: The Echo Request message is not propagated on the LPWAN to the Device. If the SCHC C/D finds a rule in the context with the IPv6 address of the Device, it responds with an Echo Reply on behalf of the Device. If no rule is found with that IPv6 address, the SCHC C/D does not respond.

TODO: again, we are assuming that no compression rule is equivalent to the device not providing the service.

- Code = 1: the SCHC C/D queries the NGW (or maintains a local database) and answers with the number of seconds since the Device last transmission.

TODO: what does it mean to answer "with the number of seconds ..."? There is no such field in an Echo Reply message. Do we overwrite one of the fields (Identifier, Sequencer Number, Data)? They are all supposed to be copied from the Echo Request. Do we change their definition with Code==1 or Code==2?

- Code = 2: the SCHC C/D compresses the ICMPv6 message and forwards it to the Device. The Echo Reply message sent by the Device is also compressed. Since the Echo Request message comes from the Internet, the values of the Identifier, Sequence Number and Data fields cannot be known in advance, and therefore must be transmitted. However, it is likely that the Echo Request with Code 2 will be firewalled from the Internet and restricted to authorized users. Therefore, the Echo Request message can be assumed to have the same content as recommended in Section 4.3, and the same compression rules apply.

5. Traceroute

The traceroute6 program sends successive probe packets destined to a chosen target but with the Hop Limit value successively incremented from the initial value 1.

It expects to receive a "Time Exceeded" (Type = 3) "Hop Limit" (Code = 0) ICMPv6 error message back from the successive routers along the path to the destination.

The probe packet is usually a UDP datagram, but can also be a TCP datagram or even an ICMPv6 message. The destination port is chosen in the unassigned range in hope that the destination, when eventually reached, will respond with a "Destination Unreachable" (Type = 1) "Port Unreachable" (Code = 4) ICMPv6 error message.

It is not anticipated that a Device will want to traceroute a destination on the Internet.

By contrast, a host on the Internet may attempt to traceroute an IPv6 address that is assigned to an LPWAN device. This is described in Figure 6.

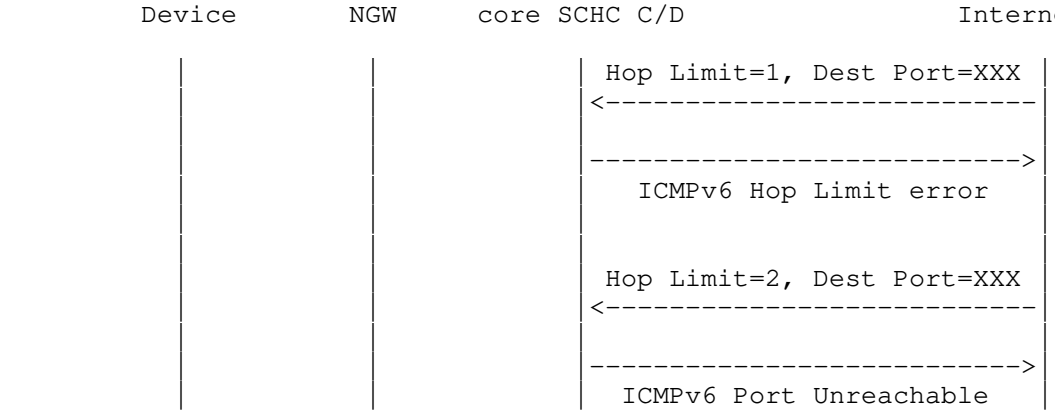


Figure 6: Example of traceroute to the LPWAN Device

When the probe packet first reaches the core SCHC C/D, its remaining Hop Limit is 1. The core SCHC C/D will respond back with a "Time Exceeded" (Type = 3) "Hop Limit" (Code = 0) ICMPv6 error message. Later on, when the probe packet reaches the code SCHC C/D with a Hop Limit value of 2, the core SCHC C/D will, as explained in Section 4.1, answer back with a "Destination Unreachable" (Type = 1) "Port Unreachable" (Code = 4) ICMPv6 error message. This is what the traceroute6 command expects. Therefore, the traceroute6 command will work with LPWAN IPv6 destinations, except for the time displayed for the destination, which is actually the time to its proxy.

However, if the probe packet happens to hit a port that matches a SCHC rule for that Device, the packet will be compressed with this rule and sent over the LPWAN, which is unfortunate. Forwarding of packets to the Device over the LPWAN should only be done from authenticated/trusted sources anyway. Rate-limitation on top of authentication will mitigate this nuisance.

6. Security considerations

TODO

7. IANA Considerations

TODO

8. References

8.1. Normative References

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

[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>.

[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>.

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <https://www.rfc-editor.org/info/rfc4884>.

[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>.

8.2. Informative References

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

Authors' Addresses

Dominique Barthel
Orange SA
28 chemin du Vieux Chene
BP 98
38243 Meylan Cedex
France

Email: dominique.barthel@orange.com

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

Arunprabhu Kandasamy
Acklio
2bis rue de la Chataigneraie
35510 Cesson-Sevigne Cedex
France

Email: arun@ackl.io

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

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

October 22, 2018

LPWAN Static Context Header Compression (SCHC) for CoAP
draft-ietf-lpwan-coap-static-context-hc-05

Abstract

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

Status of This Memo

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

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

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

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

Table of Contents

1. Introduction

CoAP [rfc7252] is an implementation of the REST architecture for constrained devices. Nevertheless, if limited, the size of a CoAP

header may be too large for LPWAN constraints and some compression may be needed to reduce the header size.

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

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

2. SCHC Compression Process

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

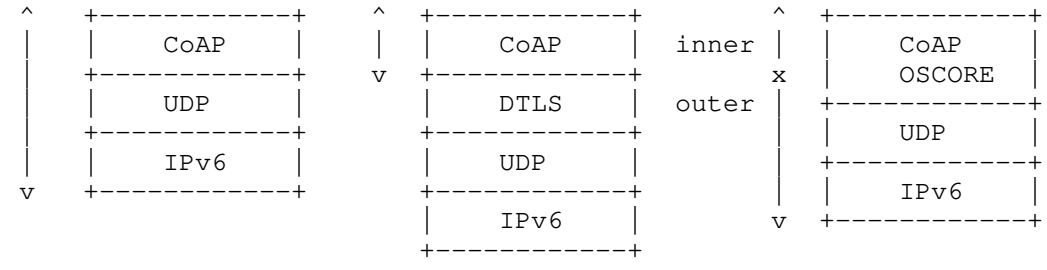


Figure 1: rule scope for CoAP

Figure 1 shows some examples for CoAP architecture and the SCHC rule's scope. A rule can covers all headers from IPv6 to CoAP, SCHC C/D is done in the device and at the LPWAN boundary. If an end-to-end encryption mechanisms is used between the device and the application. CoAP must be compressed independently of the other layers. The rule ID and the compression residue are encrypted using a mechanism such as DTLS. Only the other end can decipher the information.
Layers below may also be compressed using other SCHC rules (this is out of the scope of this document). OSCORE [I-D.ietf-core-object-security] can also define 2 rules to compress the CoAP message. A first rule focuses on the inner header and is end to end, a second rule may compress the outer header and the layer above. SCHC C/D for inner header is done by both ends, SCHC C/D for outer header and other headers is done between the device and the LPWAN boundary.

3. CoAP Compression with SCHC

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

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

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

- Even when a field is "symmetric" (i.e. found in both directions) the values carried in each direction are different. Combined with a matching list in the TV, this will allow to reduce the range of expected values in a particular direction and therefore reduce the size of a compression residue. For instance, if a client sends only CON request, the type can be elided by compression and the answer may use one bit to carry either the ACK or RST type. Same behavior can be applied to the CoAP Code field (0.0X code are present in the request and Y.ZZ in the answer). The direction allows to split in two parts the possible values for each direction.

- In IPv6 and UDP header fields have a fixed size. In CoAP, Token size may vary from 0 to 8 bytes, length is given by a field in the

header. More systematically, the CoAP options are described using the Type-Length-Value.

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

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

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

- Field size defined in the CoAP protocol can be too large regarding LPWAN traffic constraints. This is particularly true for the message ID field or Token field. The use of MSB MO can be used to reduce the information carried on LPWANs.

- CoAP also obeys to the client/server paradigm and the compression rate can be different if the request is issued from an LPWAN node or from an non LPWAN device. For instance a Device (Dev) aware of LPWAN constraints can generate a 1 byte token, but a regular CoAP client will certainly send a larger token to the Thing. SCHC compression will not modify the values to offer a better compression rate. Nevertheless a proxy placed before the compressor may change some field values to offer a better compression rate and maintain the necessary context for interoperability with existing CoAP implementations.

4. Compression of CoAP header fields

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

4.1. CoAP version field

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

4.2. CoAP type field

[rfc7252] defines 4 types of messages: CON, NON, ACK and RST. The latter two ones are a response of the two first ones. If the device plays a specific role, a rule can exploit these property with the

mapping list: [CON, NON] for one direction and [ACK, RST] for the other direction. Compression residue is reduced to 1 bit.

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

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

4.3. CoAP code field

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

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

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

4.4. CoAP Message ID field

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

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

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

4.5. CoAP Token fields

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

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

Token Value size should not be defined directly in the rule in the Field Length (FL). Instead a specific function designed as "TKL" must be used and length do not have to the sent with the residue. During the decompression, this function returns the value contained in the Token Length field.

5. CoAP options

5.1. CoAP Content and Accept options.

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

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

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

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

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

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

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

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

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

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

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

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

Figure 2: complex path example

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

5.3.1. Variable length Uri-Path and Uri-Query

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

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

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

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

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

Figure 3: CoMi URI compression

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

5.3.2. Variable number of path or query elements

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

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

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

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

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

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

These fields are unidirectional.

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

6. Other RFCs

6.1. Block

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

6.2. Observe

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

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

6.3. No-Response

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

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

6.4. Time Scale

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

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

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

6.5. OSCORE

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

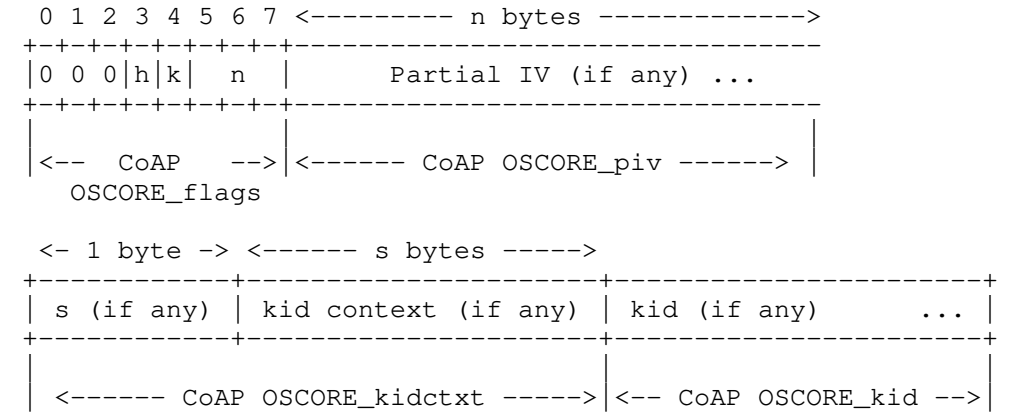


Figure 4: OSCORE Option

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

The first byte is used for flags that specify the contents of the OSCORE option. The 3 most significant bits are reserved and always set to 0. Bit h, when set, indicates the presence of the kid context field in the option. Bit k, when set, indicates the presence of a

kid field. The 3 least significant bits n indicate the length of the piv field in bytes. When n = 0, no piv is present.

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

This draft recommends to implement a parser that is able to identify the OSCORE Option and the fields it contains.

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. It is thus recommended that the parser split the OSCORE option into the 4 subsequent fields:

- CoAP OSCORE_flags,
- CoAP OSCORE_piv,
- CoAP OSCORE_kidctxt,
- CoAP OSCORE_kid.

These fields are superposed on the OSCORE Option format in Figure 4, the CoAP OSCORE_kidctxt field including the size bits s. Their size may be reduced using the MSB matching operator.

7. Examples of CoAP header compression

7.1. Mandatory header with CON message

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

Rule ID 1

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

Figure 5: CoAP Context to compress header without token

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

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

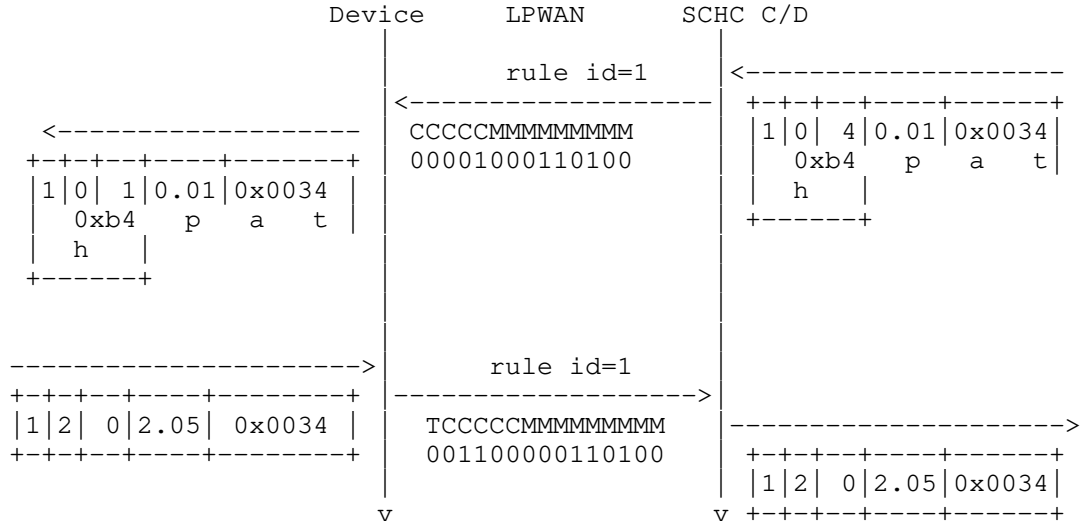


Figure 6: Compression with global addresses

7.2. OSCORE Compression

OSCORE aims to solve the problem of end-to-end encryption for CoAP messages. The goal, therefore, is to hide as much of the message as possible while still enabling proxy operation.

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

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

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

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

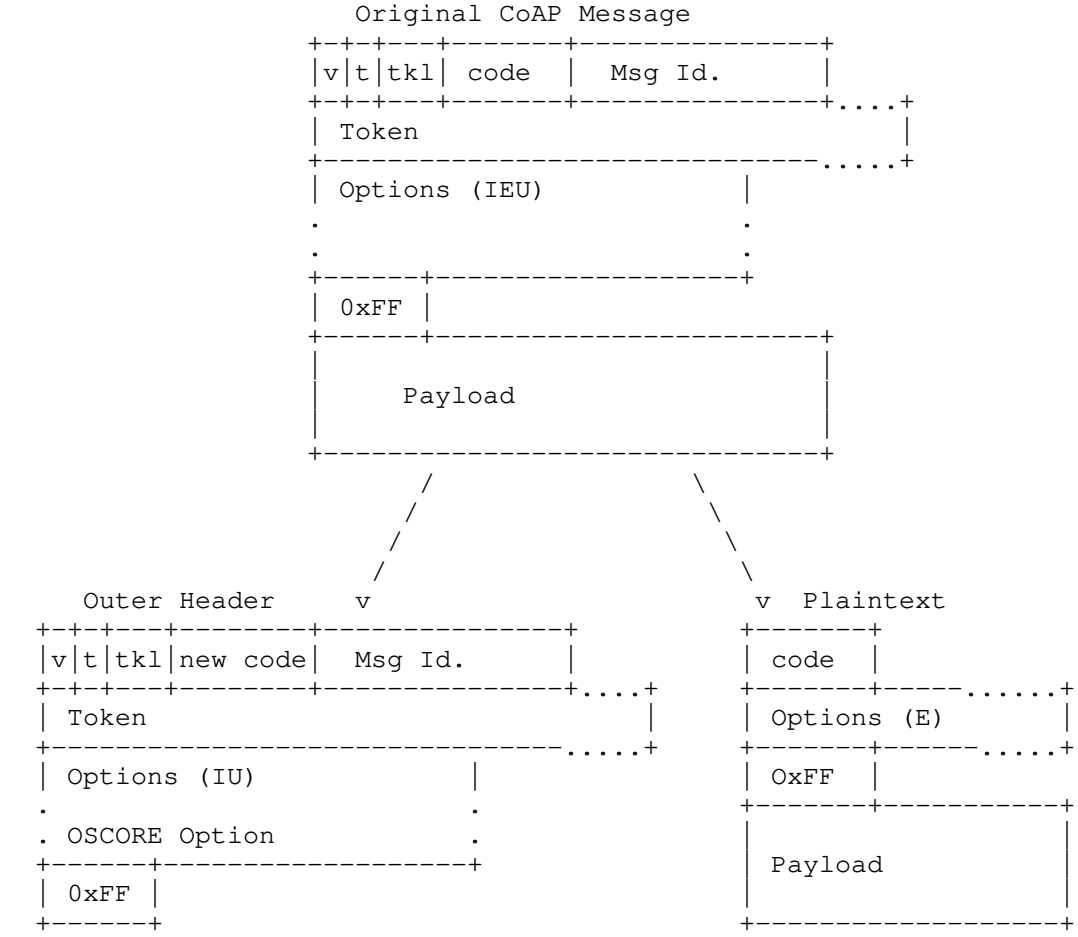


Figure 7: OSCORE inner and outer header form a CoAP message

Figure 7 shows the message format for the OSCORE Message and Plaintext.

In the Outer Header, the original message code is hidden and replaced by a default dummy value. As seen in sections 4.1.3.5 and 4.2 of

[I-D.ietf-core-object-security], the message code is replaced by POST for requests and Changed for responses when Observe is not used. If Observe is used, the message code is replaced by FETCH for requests and Content for responses.

The original message code is put into the first byte of the Plaintext. Following the message code, the class E options comes and if present the original message Payload is preceded by its payload marker.

The Plaintext is now encrypted by an AEAD algorithm which integrity protects Security Context parameters and eventually any class I options from the Outer Header. Currently no CoAP options are marked class I. The resulting Ciphertext becomes the new Payload of the OSCORE message, as illustrated in Figure 8.

This Ciphertext is, as defined in RFC 5116, the concatenation of the encrypted Plaintext and its authentication tag. Note that Inner Compression only affects the Plaintext before encryption, thus we can only aim to reduce this first, variable length component of the Ciphertext. The authentication tag is fixed in length and considered part of the cost of protection.

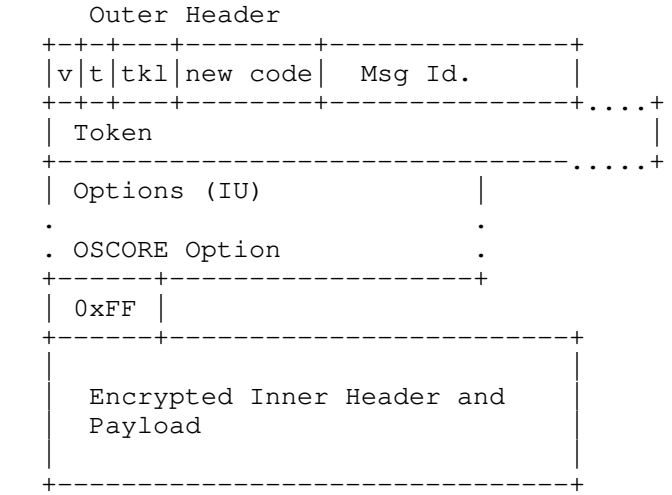


Figure 8: OSCORE message

The SCHC Compression scheme consists of compressing both the Plaintext before encryption and the resulting OSCORE message after encryption, see Figure 9.

This 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 Inner part of the message can only be decrypted by the corresponding end-point, this end-point will also have to implement Inner SCHC Compression/Decompression.

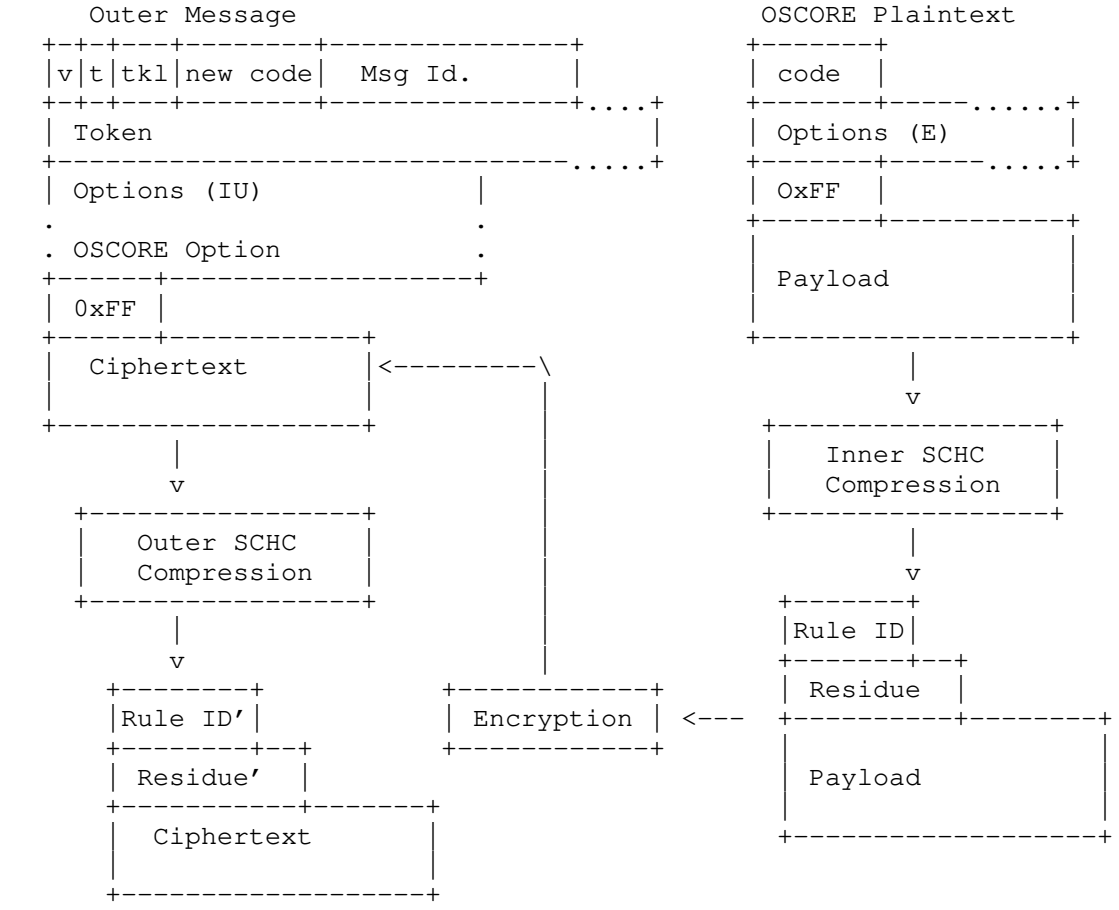


Figure 9: OSCORE Compression Diagram

7.3. Example OSCORE Compression

An example is given with a GET Request and its consequent CONTENT Response. A possible set of rules for the Inner and Outer SCHC Compression is shown. A dump of the results and a contrast between SCHC + OSCORE performance with SCHC + COAP performance is also listed. This gives an approximation to the cost of security with SCHC-OSCORE.

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

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

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

0x0001 = mid
0x82 = token

Options:
0xbb74656d7065726174757265
Option 11: URI_PATH
Value = temperature

Original msg length:   17 bytes.
```

Figure 10: CoAP GET Request

Its corresponding response is the CONTENT Response in Figure 11.

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

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

0x0001 = mid
0x82 = token

0xFF  Payload marker
Payload:
0x32332043

Original msg length:   10
```

Figure 11: CoAP CONTENT Response

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

Rule ID 0

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

Figure 12: Inner SCHC Rules

Figure 13 shows the Plaintext obtained for our example GET Request and follows the process of Inner Compression and Encryption until we end up with the Payload to be added in the outer OSCORE Message.

In this case the original message has no payload and its resulting Plaintext can be compressed up to only 1 byte (size of the Rule ID). The AEAD algorithm preserves this length in its first output, but also yields a fixed-size tag which cannot be compressed and must be

included in the OSCORE message. This translates into an overhead in total message length, which limits the amount of compression that can be achieved and plays into the cost of adding security to the exchange.

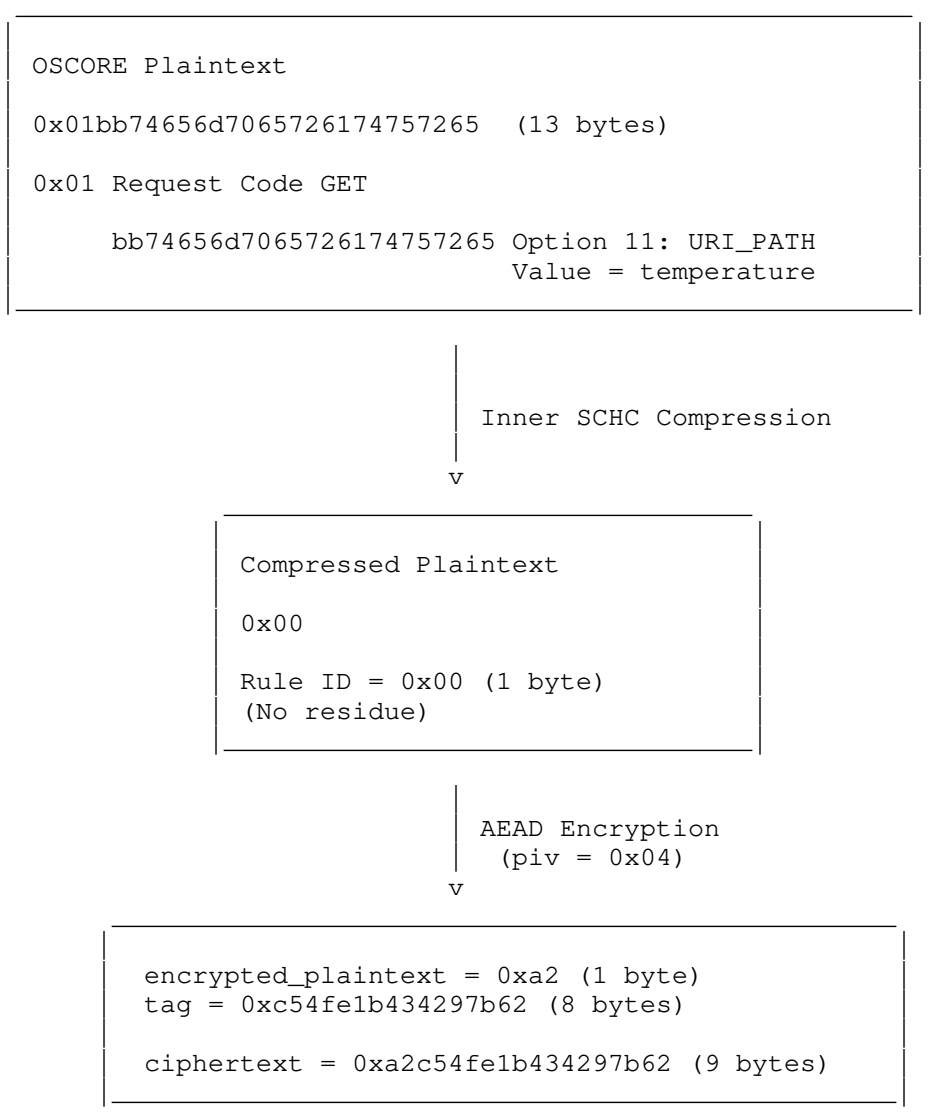


Figure 13: Plaintext compression and encryption for GET Request

In Figure 14 we repeat the process for the example CONTENT Response. In this case the misalignment produced by the compression residue (1 bit) makes it so that 7 bits of padding must be applied after the payload, resulting in a compressed Plaintext that is the same size as before compression. This misalignment also causes the hexcode from the payload to differ from the original, even though it has not been compressed. On top of this, the overhead from the tag bytes is incurred as before.

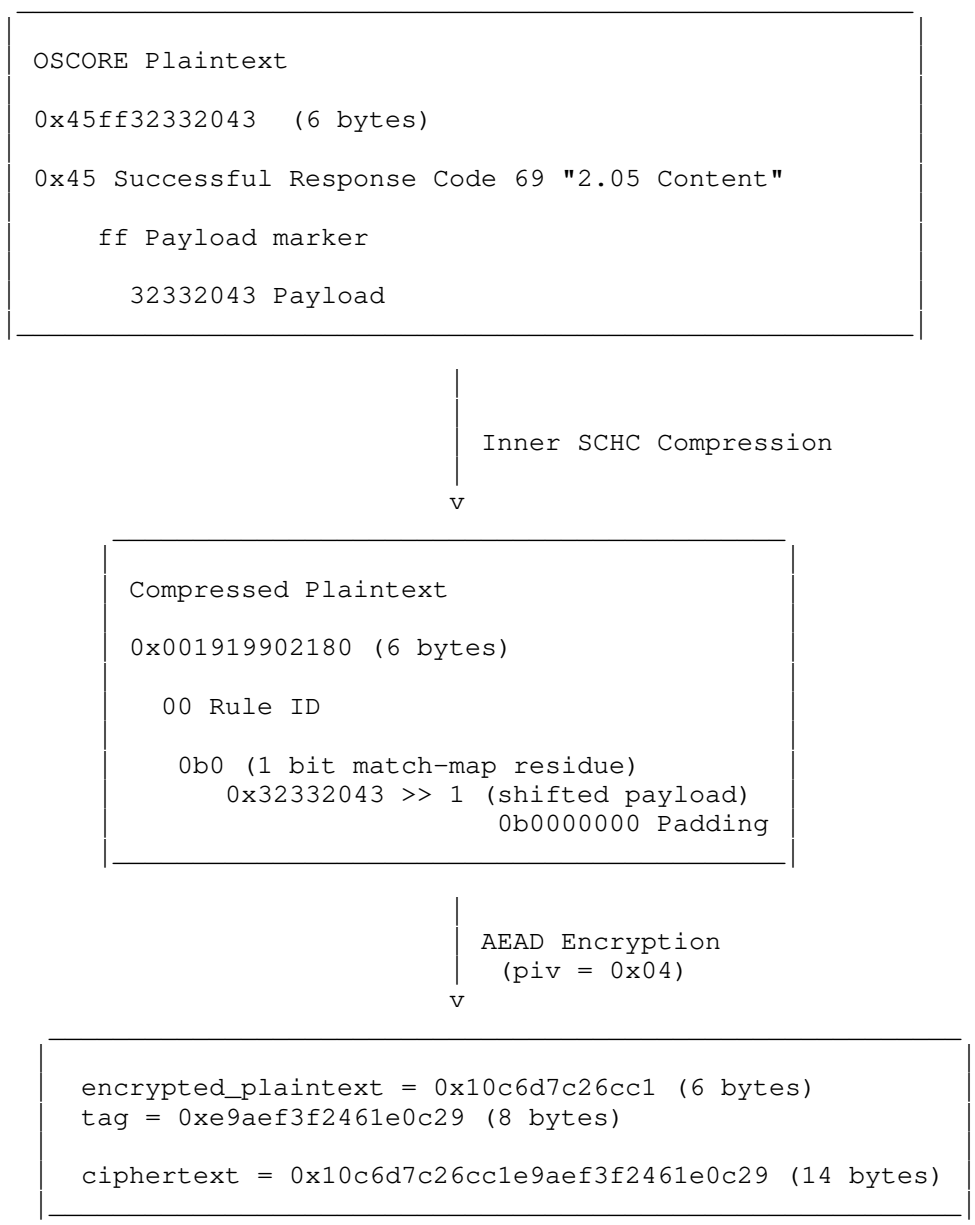


Figure 14: Plaintext compression and encryption for CONTENT Response

The Outer SCHC Rules (Figure 17) must process the OSCORE Options fields. In Figure 15 and Figure 16 we show a dump of the OSCORE

Messages generated from our example messages once they have been provided with the Inner Compressed Ciphertext in the payload. These are the messages that are to go through Outer SCHC Compression.

```
Protected message:
==================
0x4102000182d7080904636c69656e74ffa2c54fe1b434297b62
(25 bytes)

Header:
0x4102
01   Ver
  00   CON
    0001   tkl
        00000010   Request Code 2 "POST"

0x0001 = mid
0x82 = token

Options:
0xd7080904636c69656e74 (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 15: 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 16: Protected and Inner SCHC Compressed CONTENT Response

For the flag bits, a number of compression methods could prove to be useful depending on the application. The simplest alternative is to provide a fixed value for the flags, combining MO equal and CDA not-sent. This saves most bits but could hinder flexibility. Otherwise, match-mapping could allow to choose from a number of configurations of interest to the exchange. If neither of these alternatives is desirable, MSB could be used to mask off the 3 hard-coded most significant bits.

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

The piv field lends itself to having a number of bits masked off with MO MSB and CDA LSB. This could prove useful in applications where the message frequency is low such as that found in LPWAN technologies. Note that compressing the sequence numbers effectively reduces the maximum amount of sequence numbers that can be used in an exchange. Once this amount is exceeded, the SCHC Context would need to be re-established.

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

Figure 17 shows a possible set of Outer Rules to compress the Outer Header.

Rule ID 0

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

Figure 17: Outer SCHC Rules

These Outer Rules are applied to the example GET Request and CONTENT Response. The resulting messages are shown in Figure 18 and Figure 19.

```
Compressed message:
==================
0x001489458a9fc3686852f6c4 (12 bytes)
0x00 Rule ID
    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 18: SCHC-OSCORE Compressed GET Request

```
Compressed message:
==================
0x0014218daf84d983d35de7e48c3c1852 (16 bytes)
0x00 Rule ID
    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 19: SCHC-OSCORE Compressed CONTENT Response

For contrast, we compare these results with what would be obtained by SCHC compressing the original CoAP messages without protecting them with OSCORE. To do this, we compress the CoAP messages according to the SCHC rules in Figure 20.

Rule ID 1

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

Figure 20: SCHC-CoAP Rules (No OSCORE)

This yields the results in Figure 21 for the Request, and Figure 22 for the Response.

```
Compressed message:
==================
0x0114
0x01 = Rule ID

Compression residue:
0b00010100 (1 byte)

Compressed msg length: 2
```

Figure 21: CoAP GET Compressed without OSCORE

```
Compressed message:
==================
0x010a32332043
0x01 = Rule ID

Compression residue:
0b00001010 (1 byte)

Payload
0x32332043

Compressed msg length: 6
```

Figure 22: CoAP CONTENT Compressed without OSCORE

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

8. Normative References

[I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", draft-ietf-core-object-security-15 (work in progress), August 2018.

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

[I-D.toutain-core-time-scale]
Minaburo, A. and L. Toutain, "CoAP Time Scale Option", draft-toutain-core-time-scale-00 (work in progress), October 2017.

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

[rfc7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[rfc7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

[rfc7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <https://www.rfc-editor.org/info/rfc7967>.

Authors' Addresses

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

Email: ana@ackl.io

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

Email: Laurent.Toutain@imt-atlantique.fr

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

Email: randreasen@fi.uba.ar

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

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

October 22, 2018

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

Abstract

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

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

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

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

Status of This Memo

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

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

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

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

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

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

2. Requirements Notation

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

3. LPWAN Architecture

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

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

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

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

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

o Application Server (App)

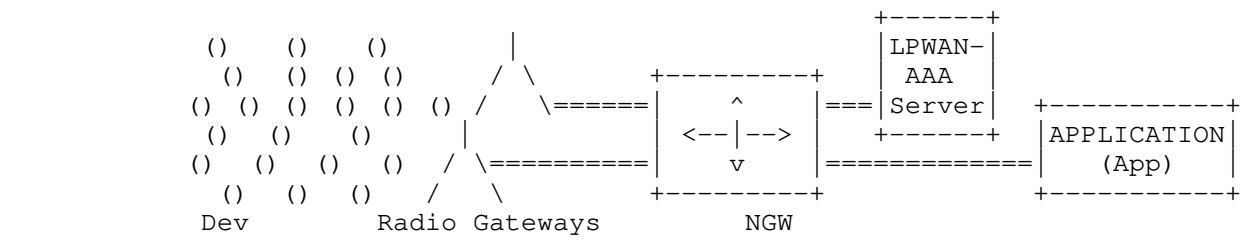


Figure 1: LPWAN Architecture

4. Terminology

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

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

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

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

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

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

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

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

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

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

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

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

- Field Description. A line in the Rule table.

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

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

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

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

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

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

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

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

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

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

- Rule: A set of header field values.

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

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

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

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

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

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

5. SCHC overview

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

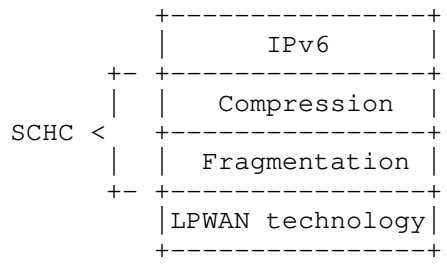
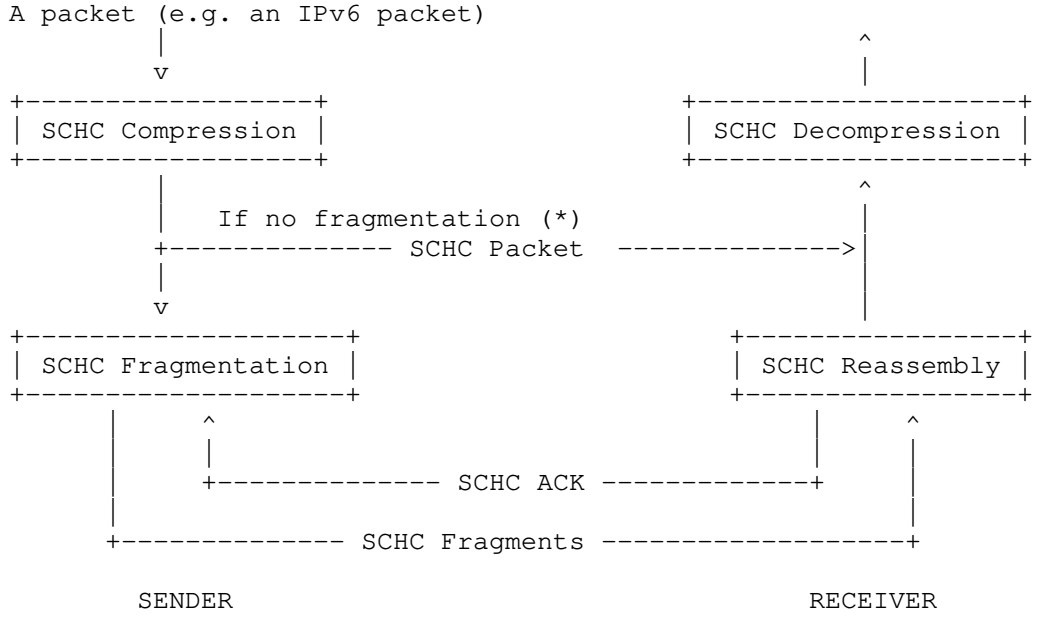


Figure 2: Protocol stack comprising IPv6, SCHC and an LPWAN technology

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



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

Figure 3: SCHC operations at the SENDER and the RECEIVER

5.1. SCHC Packet format

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

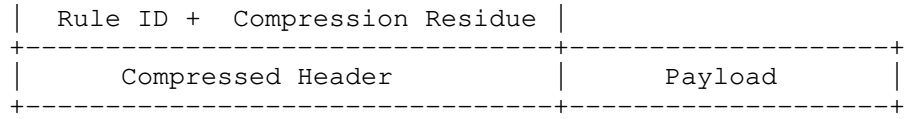


Figure 4: SCHC Packet

5.2. Functional mapping

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

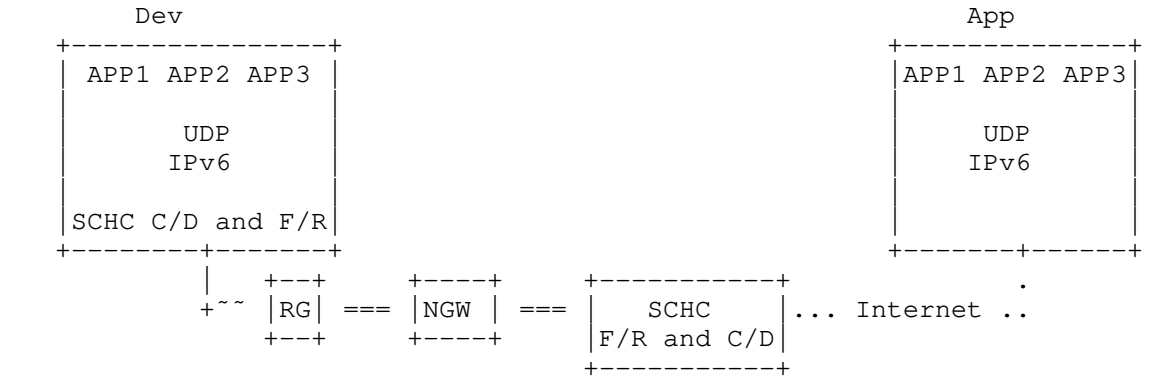


Figure 5: Architecture

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

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

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

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

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

6. Rule ID

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

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

The Rule IDs are used:

- In the SCHC C/D context, to identify the Rule (i.e., the set of Field Descriptions) that is used to compress a packet header.
- At least one Rule ID MAY be allocated to tagging packets for which SCHC compression was not possible (no matching Rule was found).
- In SCHC F/R, to identify the specific modes and settings of SCHC Fragments being transmitted, and to identify the SCHC ACKs, including their modes and settings. Note that when F/R is used for both communication directions, at least two Rule ID values are therefore needed for F/R.

7. Compression/Decompression

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

7.1. SCHC C/D Rules

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

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

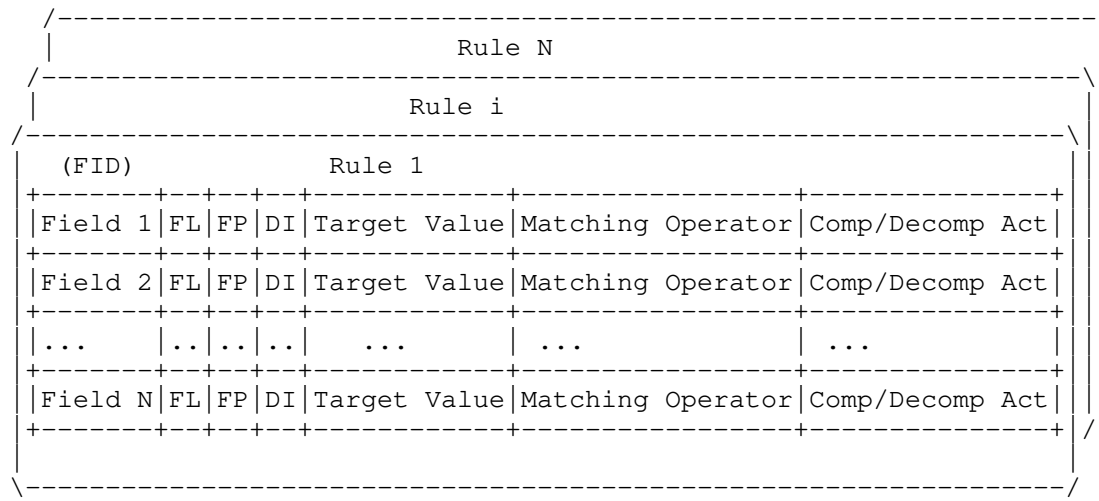


Figure 6: A Compression/Decompression Context

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

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

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

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

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

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

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

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

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

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

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

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

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

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

7.2. Rule ID for SCHC C/D

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

7.3. Packet processing

The compression/decompression process follows several steps:

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

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

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

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

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

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

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

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

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

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

7.4. Matching operators

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

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

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

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

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

7.5. Compression Decompression Actions (CDA)

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

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

Figure 7: Compression and Decompression Actions

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

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

7.5.1. processing variable-length fields

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

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

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

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

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

7.5.2. not-sent CDA

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

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

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

7.5.3. value-sent CDA

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

7.5.4. mapping-sent CDA

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

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

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

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

7.5.5. LSB CDA

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

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

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

7.5.6. DevIID, AppIID CDA

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

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

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

7.5.7. Compute-*

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

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

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

8. Fragmentation/Reassembly

8.1. Overview

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

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

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

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

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

8.2. SCHC F/R Tools

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

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

8.2.1. Messages

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

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

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

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

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

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

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

8.2.2. Tiles, Windows, Bitmaps, Timers, Counters

8.2.2.1. Tiles

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

See Figure 8 for an example.

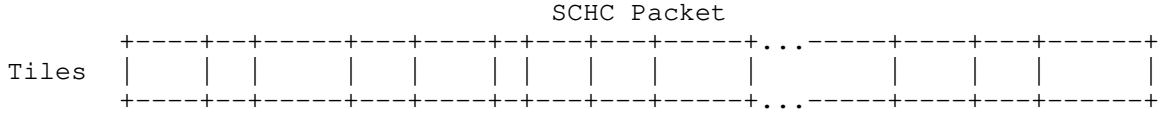


Figure 8: a SCHC Packet fragmented in tiles

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

8.2.2.2. Windows

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

If windows are used

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

- WINDOW_SIZE MUST be specified in a Profile.

- the windows are numbered.

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

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

See Figure 9 for an example.

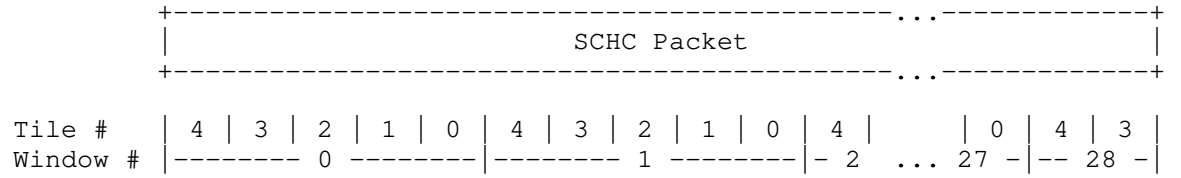


Figure 9: a SCHC Packet fragmented in tiles grouped in 28 windows, with WINDOW_SIZE = 5

When windows are used

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

8.2.2.3. Bitmaps

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

At the receiver

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

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

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

8.2.2.4. Timers and counters

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

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

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

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

8.2.3. Integrity Checking

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

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

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

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

8.2.4. Header Fields

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8.3. SCHC F/R Message Formats

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

8.3.1. SCHC Fragment format

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

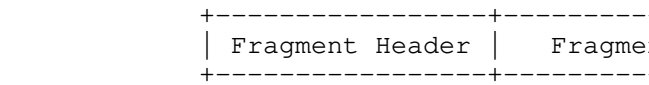


Figure 10: SCHC Fragment general format. Presence of a padding field is optional

8.3.1.1. Regular SCHC Fragment

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

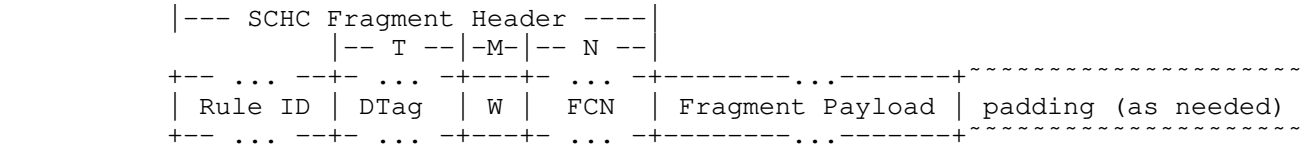


Figure 11: Detailed Header Format for Regular SCHC Fragments

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

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

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

8.3.1.2. All-1 SCHC Fragment

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

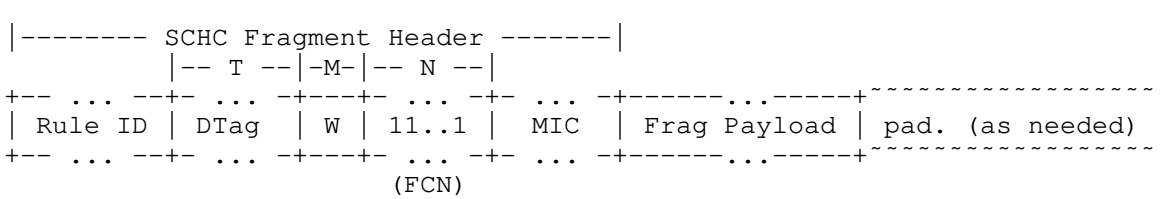


Figure 12: Detailed format for the All-1 SCHC Fragment

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

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

8.3.2. SCHC ACK format

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

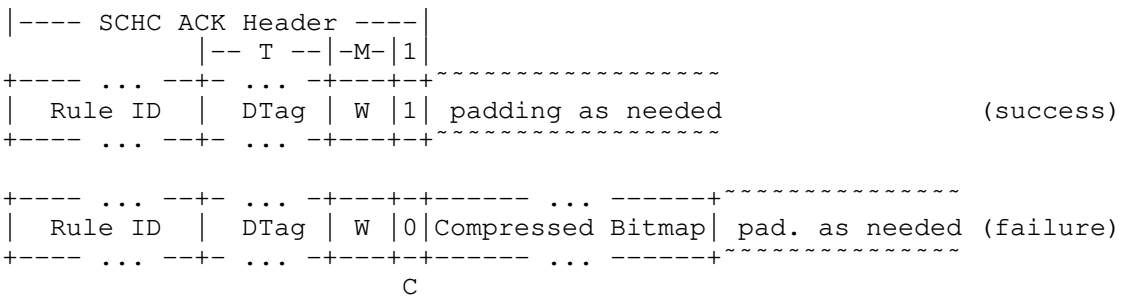


Figure 13: Format of the SCHC ACK message

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

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

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

- a representation of the Bitmap for the window referred to by the W field MUST follow the C bit
- padding bits MUST be appended as needed to fill up the last L2 Word

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

See Section 8.2.2.3 for a description of the Bitmap.

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

8.3.2.1. Bitmap Compression

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

- Build a temporary SCHC ACK message that contains the Header followed by the original Bitmap.
- Positioning scissors at the end of the Bitmap, after its last bit.

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

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

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

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

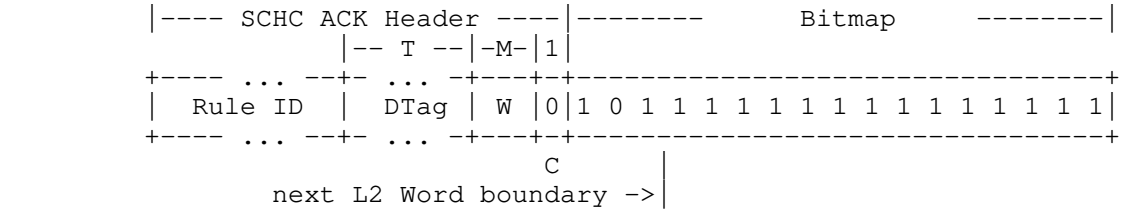


Figure 14: Tentative SCHC ACK message with Bitmap before compression

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

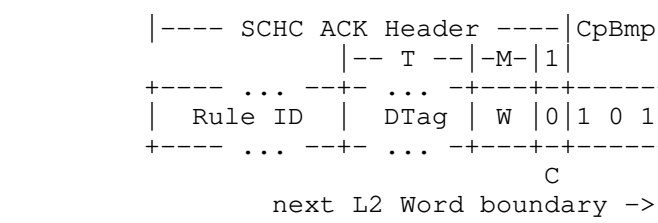


Figure 15: Actual SCHC ACK message with Compressed Bitmap, no padding

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

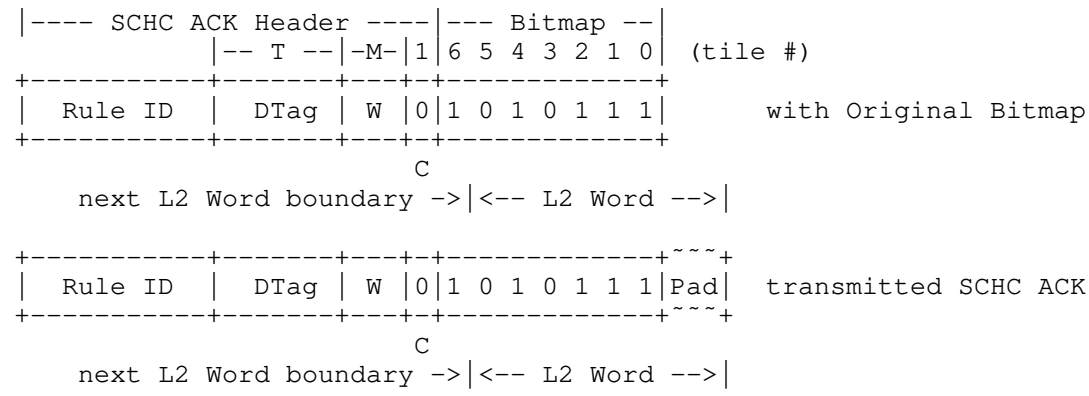


Figure 16: Example of a SCHC ACK message, missing tiles, with padding

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

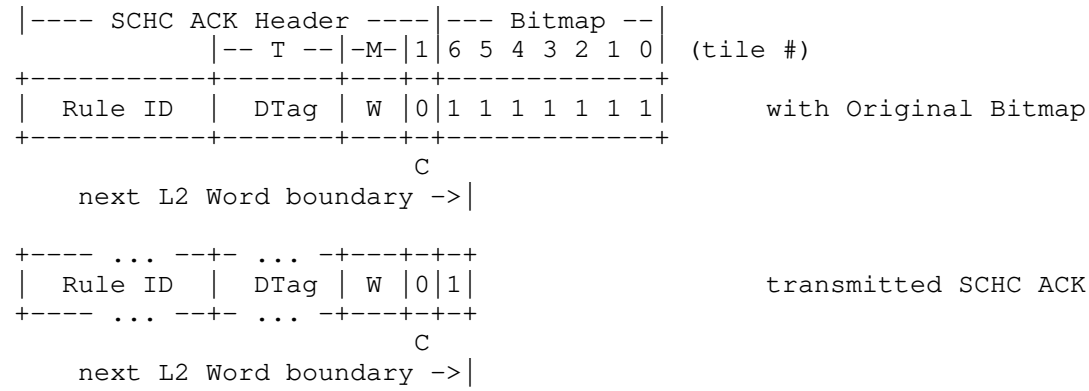


Figure 17: Example of a SCHC ACK message, no missing tile, no padding

8.3.3. SCHC ACK REQ format

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

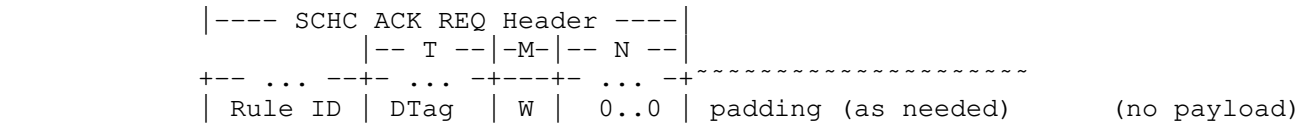


Figure 18: SCHC ACK REQ detailed format

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

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

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

8.3.4. SCHC Abort formats

8.3.4.1. SCHC Sender-Abort

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

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

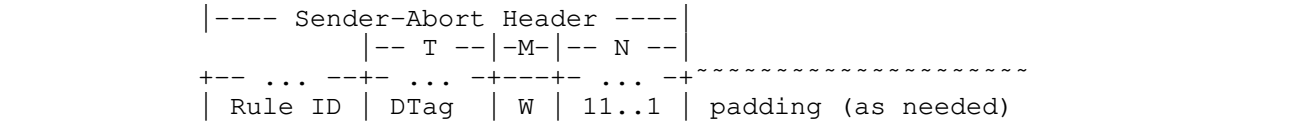


Figure 19: SCHC Sender-Abort format

If the W field is present,

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

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

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

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

The SCHC Sender-Abort MUST NOT be acknowledged.

8.3.4.2. SCHC Receiver-Abort

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

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

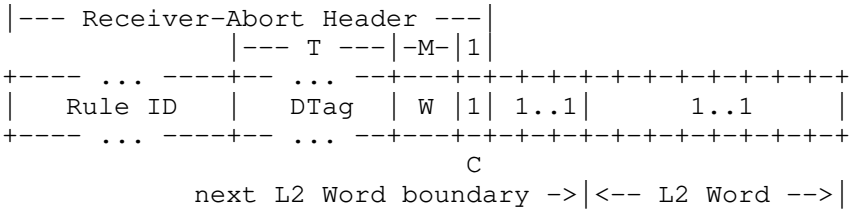


Figure 20: SCHC Receiver-Abort format

If the W field is present,

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

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

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

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

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

The SCHC Receiver-Abort MUST NOT be acknowledged.

8.4. SCHC F/R modes

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

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

More modes may be defined in the future.

8.4.1. No-ACK mode

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

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

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

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

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

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

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

Each Profile, for each Rule ID value, MUST define

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

Each Profile, for each Rule ID value, MAY define

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

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

- one Inactivity Timer

8.4.1.1. Sender behaviour

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

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

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

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

The sender MAY transmit a SCHC Sender-Abort.

Figure 35 shows an example of a corresponding state machine.

8.4.1.2. Receiver behaviour

On receiving Regular SCHC Fragments,

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

On receiving an All-1 SCHC Fragment,

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

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

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

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

Figure 36 shows an example of a corresponding state machine.

8.4.2. ACK-Always

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

- Data unit out-of-sequence delivery does not occur between the entity performing fragmentation and the entity performing reassembly
- The L2 MTU value does not change while a fragmented SCHC Packet is being transmitted.

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

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

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

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

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

Each Profile, for each Rule ID value, MUST define

- the value of N (size of the FCN field),
- the value of MAX_WIND_FCN
- the size and algorithm for the MIC field in the SCHC F/R messages, if different from the default,

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

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

- one Attempts counter
- one Retransmission Timer

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

- one Inactivity Timer

8.4.2.1. Sender behaviour

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

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

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

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

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

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

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

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

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

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

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

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

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

At any time,

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

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

Figure 37 shows an example of a corresponding state machine.

8.4.2.2. Receiver behaviour

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

In the "equalization phase":

- if the window is a not-last window

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

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

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

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

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

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

o if the window is the last window

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

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

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

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

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

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

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

- if the SCHC Fragment received was an All-1 SCHC Fragment, the receiver MUST send a SCHC ACK for this window
- it keeps accepting incoming messages.

In the "clean-up phase":

- Any received SCHC F/R message with a W bit different from the local W bit MUST be silently ignored and discarded.
- Any received SCHC F/R message different from an All-1 SCHC Fragment or a SCHC ACK REQ MUST be silently ignored and discarded.
- On receiving an All-1 SCHC Fragment or a SCHC ACK REQ, the receiver MUST send a SCHC ACK.
- On expiration of the Inactivity Timer, the receive process for that SCHC Packet MAY exit

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

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

Figure 38 shows an example of a corresponding state machine.

8.4.3. ACK-on-Error

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

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

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

See Figure 21 for an example.

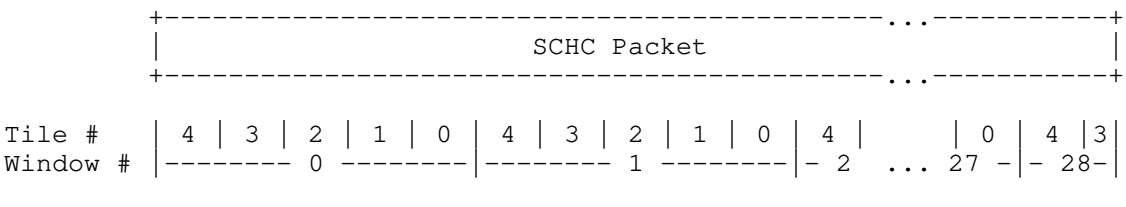


Figure 21: a SCHC Packet fragmented in tiles, Ack-on-Error mode

The W field is wide enough that it unambiguously represents an absolute window number. The fragment receiver feeds SCHC ACKs back to the fragment sender about windows that it misses tiles of. No SCHC ACK is fed back by the fragment receiver for windows that it knows have been fully received.

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

- integrity checking shows that the fragmented SCHC Packet has been correctly reassembled at the receive end, and this information has been conveyed back to the sender,

- or too many retransmission attempts were made,

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

Each Profile MUST specify which Rule ID value(s) is (are) allocated to this ACK-on-Error mode. For brevity, the rest of Section 8.4.3 only refers to SCHC F/R messages with Rule ID values that are allocated to this mode.

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

Each Profile, for each Rule ID value, MUST define

- the tile size (a tile does not need to be multiple of an L2 Word, but it MUST be at least the size of an L2 Word)

- the value of M (size of the W field),

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

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

- one Attempts counter
- one Retransmission Timer

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

- one Inactivity Timer

8.4.3.1. Sender behaviour

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- an All-1 SCHC Fragment

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

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

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

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

On Retransmission Timer expiration

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

On receiving a SCHC ACK,

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

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

o otherwise, the fragment sender

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

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

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

8.4.3.2. Receiver behaviour

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

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

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

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

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

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

o if the FCN is All-1, if a Payload is present, the full SCHC Fragment Payload MUST be assembled including the padding bits. This is because the size of the last tile is not known by the receiver, therefore padding bits are indistinguishable from the

tile data bits, at this stage. They will be removed by the SCHC C/D sublayer. If the size of the SCHC Fragment Payload exceeds or equals the size of one regular tile plus the size of an L2 Word, this SHOULD raise an error flag.

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

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

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

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

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

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

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

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

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

Reassembly of the SCHC Packet concludes when

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

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

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

9. Padding management

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

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

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

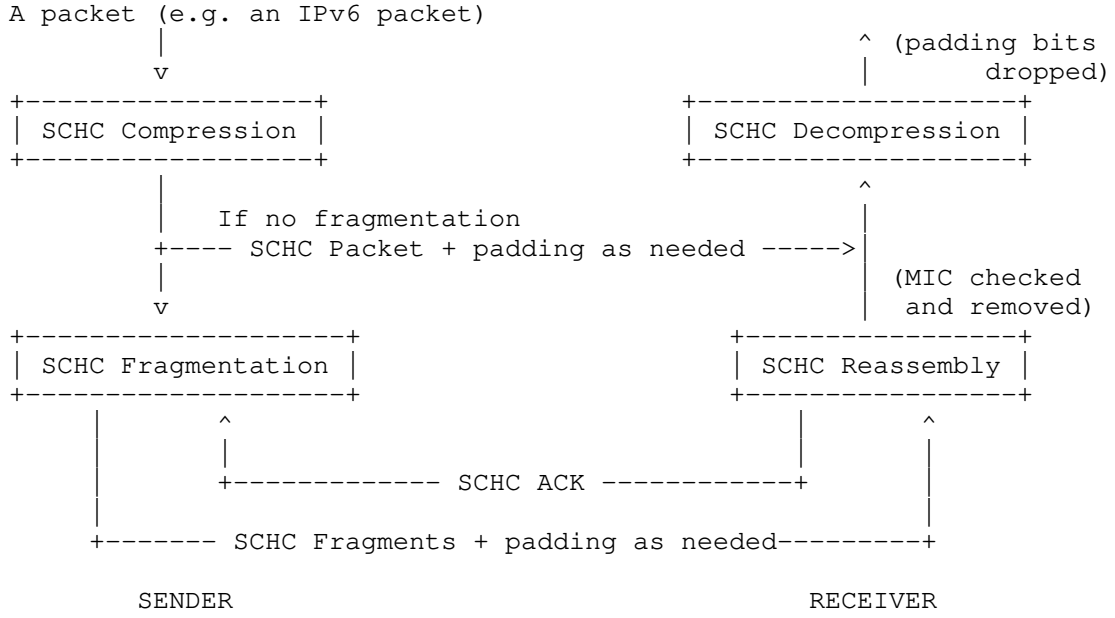


Figure 22: SCHC operations, including padding as needed

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

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

10. SCHC Compression for IPv6 and UDP headers

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

10.1. IPv6 version field

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

10.2. IPv6 Traffic class field

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

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

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

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

10.3. Flow label field

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

Otherwise, two possibilities can be considered:

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

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

10.4. Payload Length field

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

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

10.5. Next Header field

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

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

10.6. Hop Limit field

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

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

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

10.7. IPv6 addresses fields

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

10.7.1. IPv6 source and destination prefixes

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

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

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

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

10.7.2. IPv6 source and destination IID

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

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

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

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

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

10.8. IPv6 extensions

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

10.9. UDP source and destination port

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

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

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

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

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

10.10. UDP length field

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

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

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

10.11. UDP Checksum field

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

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

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

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

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

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

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

11. IANA Considerations

This document has no request to IANA.

12. Security considerations

12.1. Security considerations for SCHC Compression/Decompression

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

12.2. Security considerations for SCHC Fragmentation/Reassembly

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

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

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

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

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

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

13. Acknowledgements

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

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

14. References

14.1. Normative References

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

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

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

14.2. Informative References

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

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

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

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

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

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

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

[RFC8376] Farrell, S., Ed., "Low-Power Wide Area Network (LPWAN) Overview", RFC 8376, DOI 10.17487/RFC8376, May 2018, <https://www.rfc-editor.org/info/rfc8376>.

Appendix A. SCHC Compression Examples

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

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

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

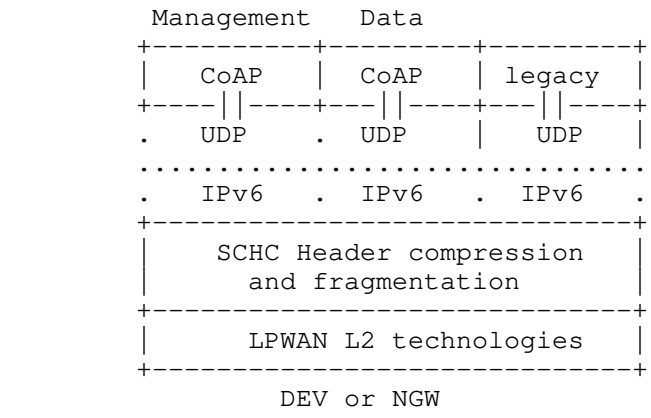


Figure 23: Simplified Protocol Stack for LP-WAN

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

Rule 0

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

Rule 1

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

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

Rule 2

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

Figure 24: Context Rules

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

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

The third Rule compresses port numbers to 4 bits.

Appendix B. Fragmentation Examples

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

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

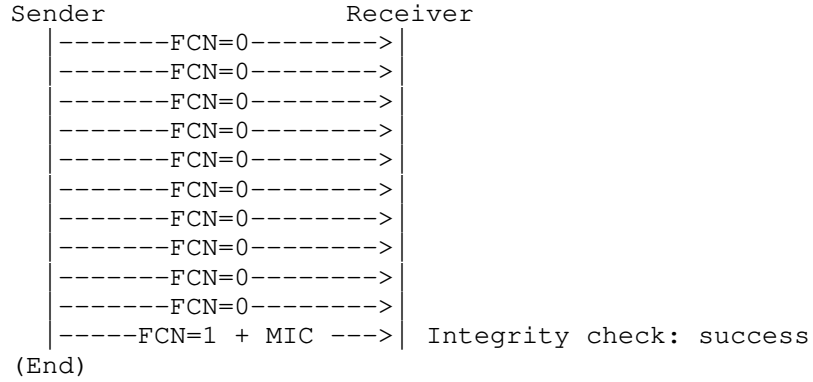


Figure 25: Transmission in No-ACK mode of a SCHC Packet carried by 11 SCHC Fragments

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

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

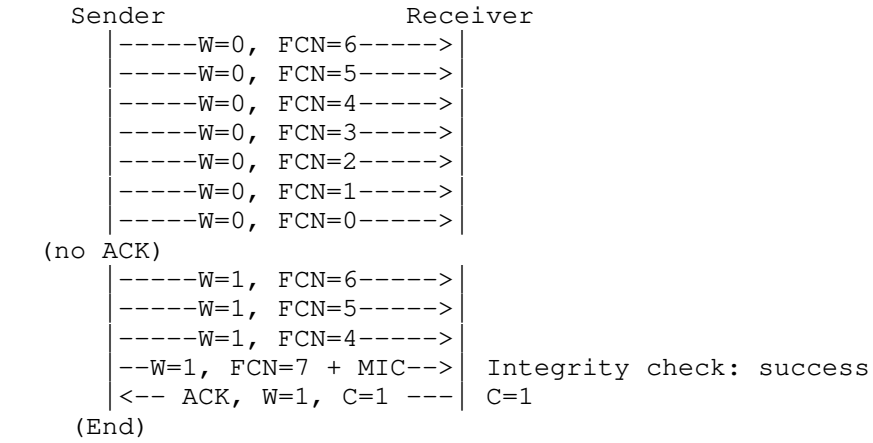


Figure 26: Transmission in ACK-on-Error mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, MAX_WIND_FCN=6 and no lost SCHC Fragment.

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

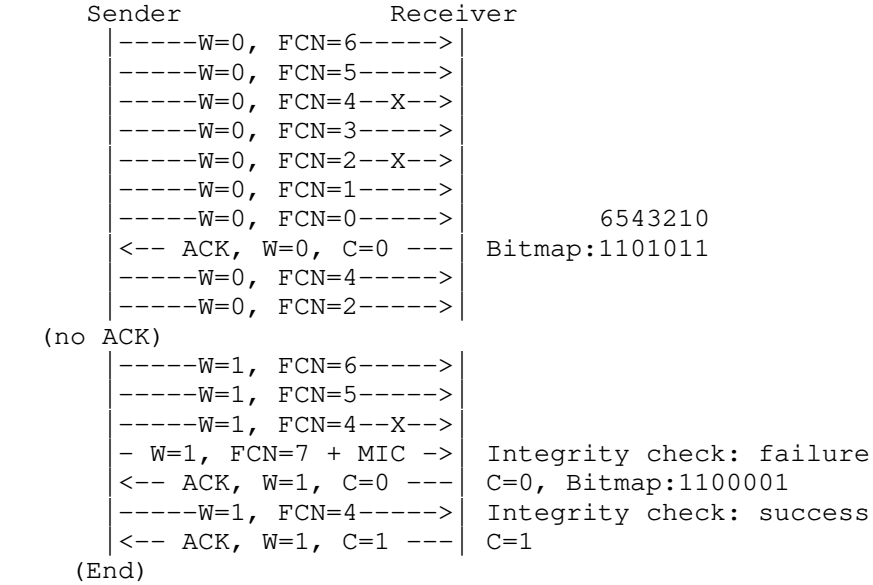


Figure 27: Transmission in ACK-on-Error mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, MAX_WIND_FCN=6 and three lost SCHC Fragments.

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

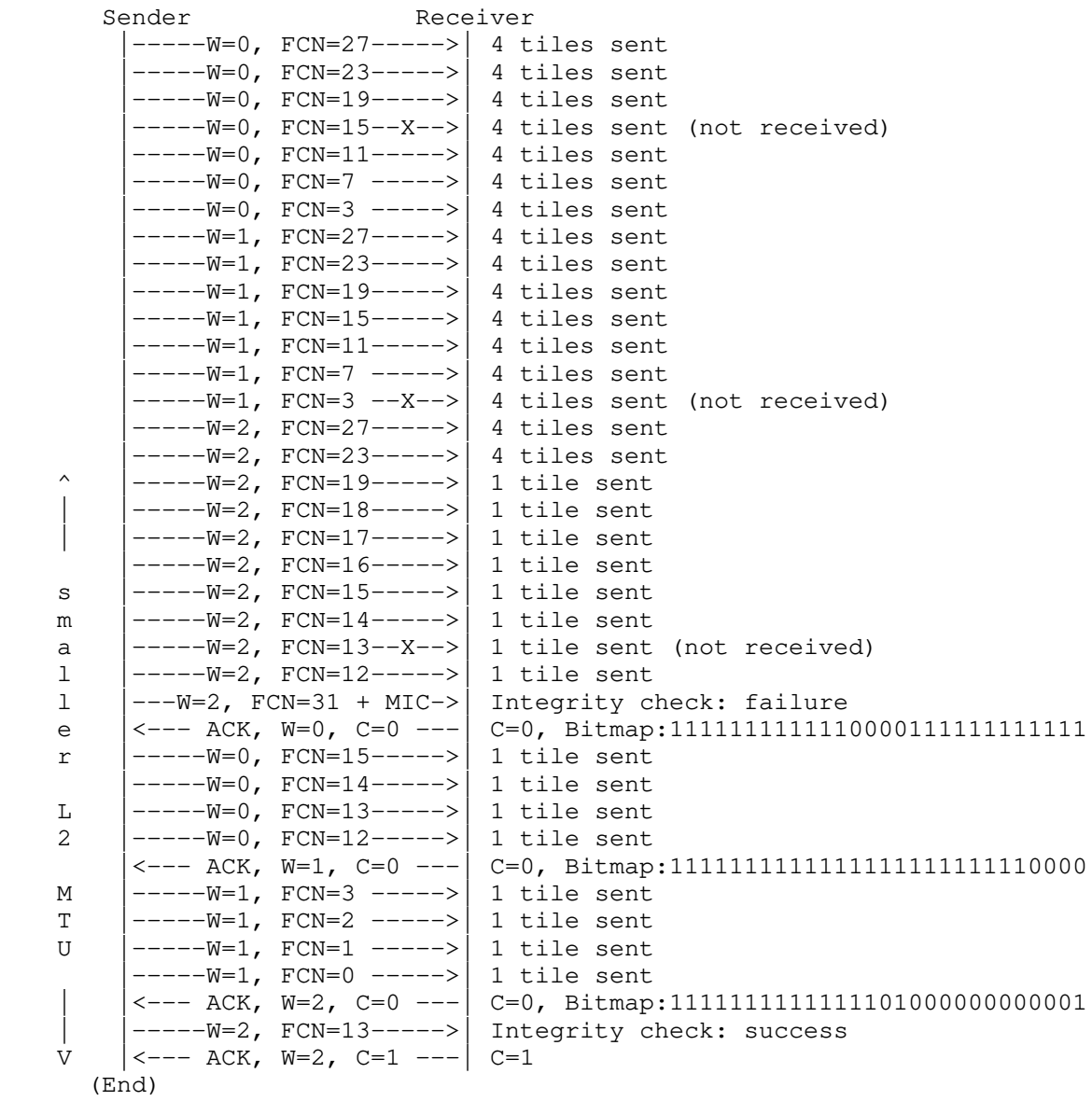


Figure 28: ACK-on-Error mode with variable MTU.

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

Note 1: Bitmaps are shown prior to compression for transmission

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

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

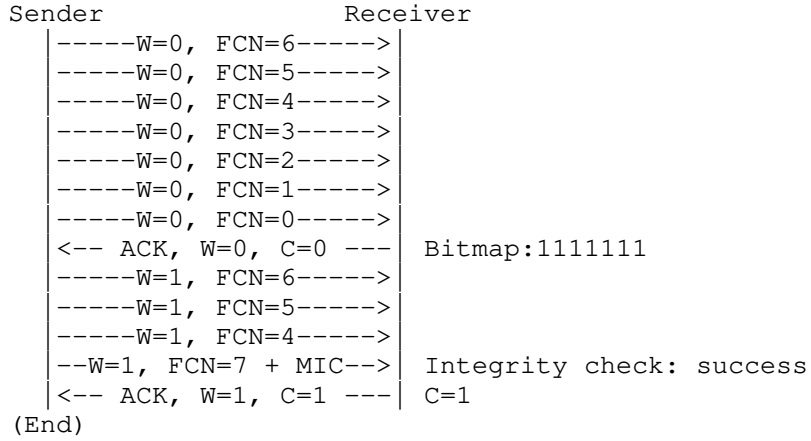


Figure 29: Transmission in ACK-Always mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, with N=3, MAX_WIND_FCN=6 and no loss.

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

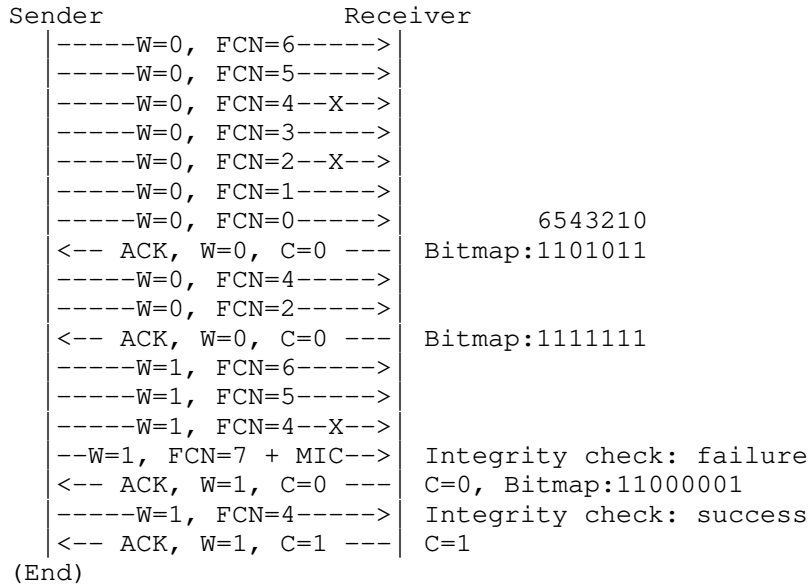


Figure 30: Transmission in ACK-Always mode of a SCHC Packet fragmented in 11 tiles, with one tile per SCHC Fragment, N=3, MAX_WIND_FCN=6 and three lost SCHC Fragments.

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

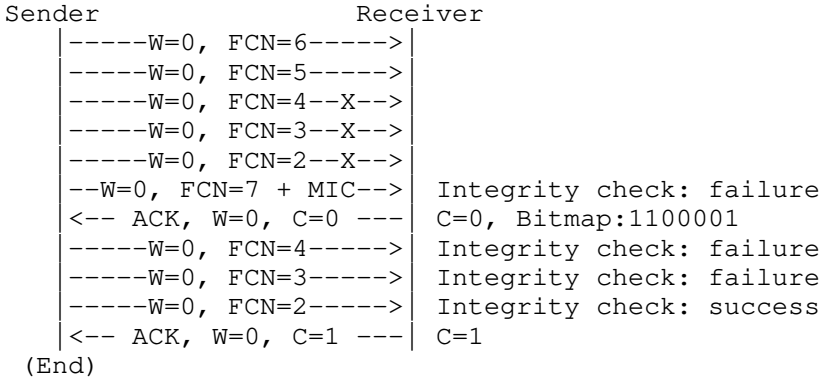


Figure 31: Transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with one tile per SCHC Fragment, N=3, MAX_WIND_FCN=6, three lost SCHC Fragments.

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

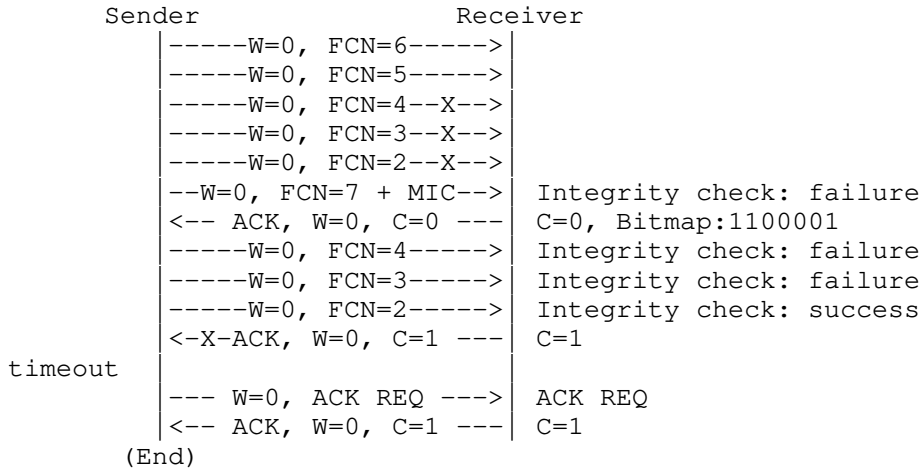


Figure 32: Transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with one tile per SCHC Fragment, N=3, MAX_WIND_FCN=6, three lost SCHC Fragments, and the second SCHC ACK lost.

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

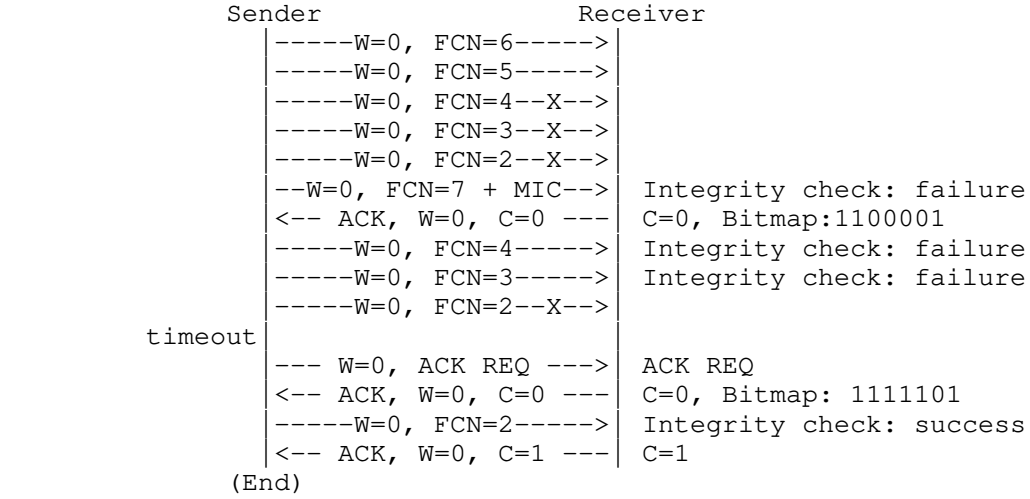


Figure 33: Transmission in ACK-Always mode of a SCHC Packet fragmented in 6 tiles, with N=3, MAX_WIND_FCN=6, with three lost SCHC Fragments, and one retransmitted SCHC Fragment lost again.

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

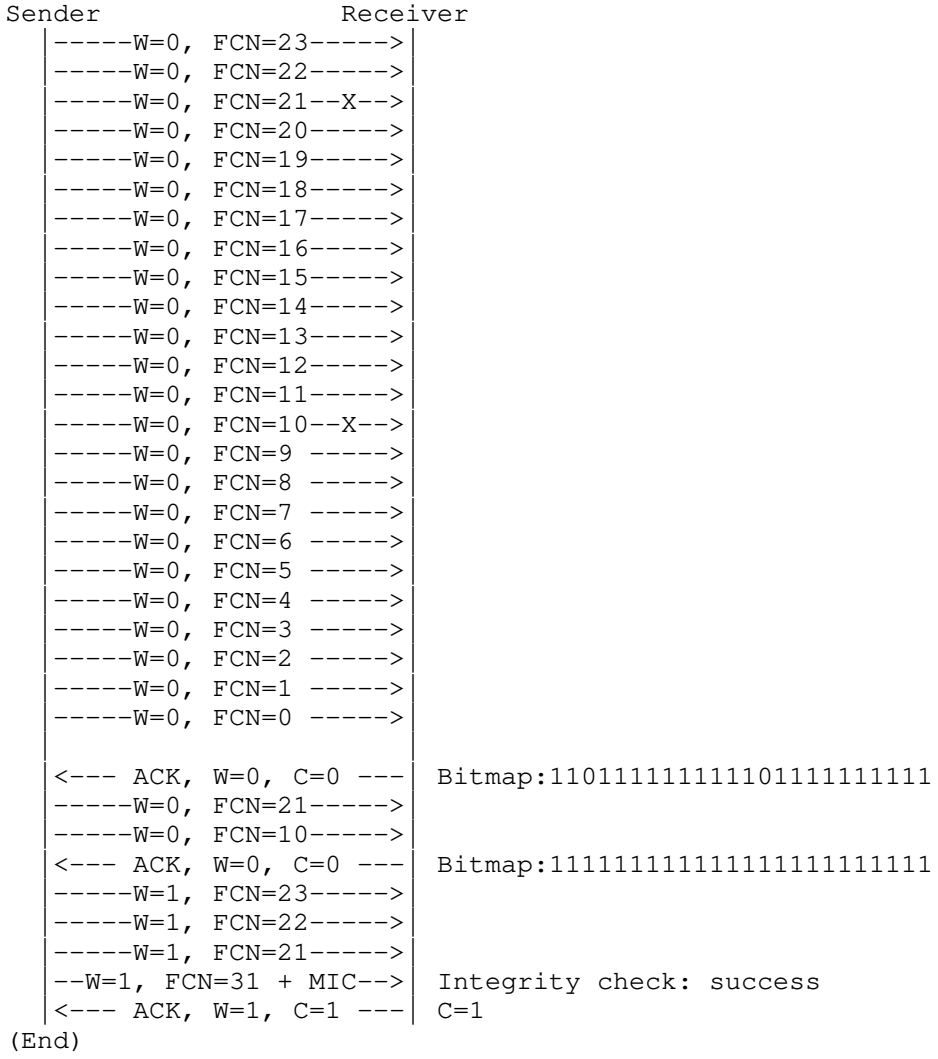


Figure 34: Transmission in ACK-Always mode of a SCHC Packet fragmented in 28 tiles, with one tile per SCHC Fragment, N=5, MAX_WIND_FCN=23 and two lost SCHC Fragments.

Appendix C. Fragmentation State Machines

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

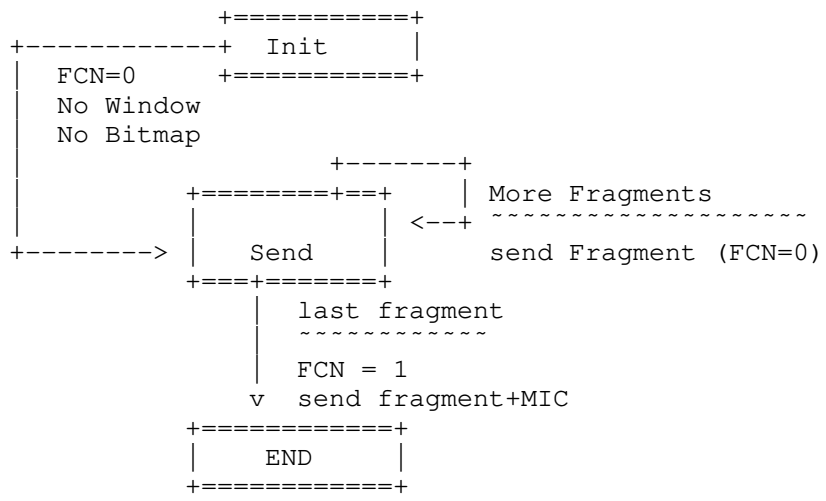


Figure 35: Sender State Machine for the No-ACK Mode

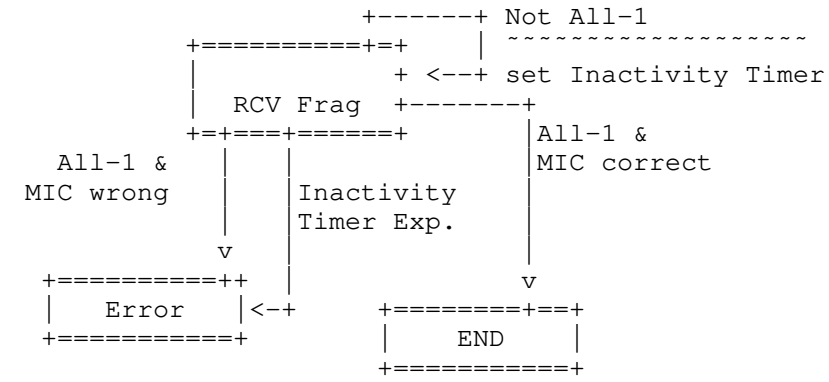


Figure 36: Receiver State Machine for the No-ACK Mode

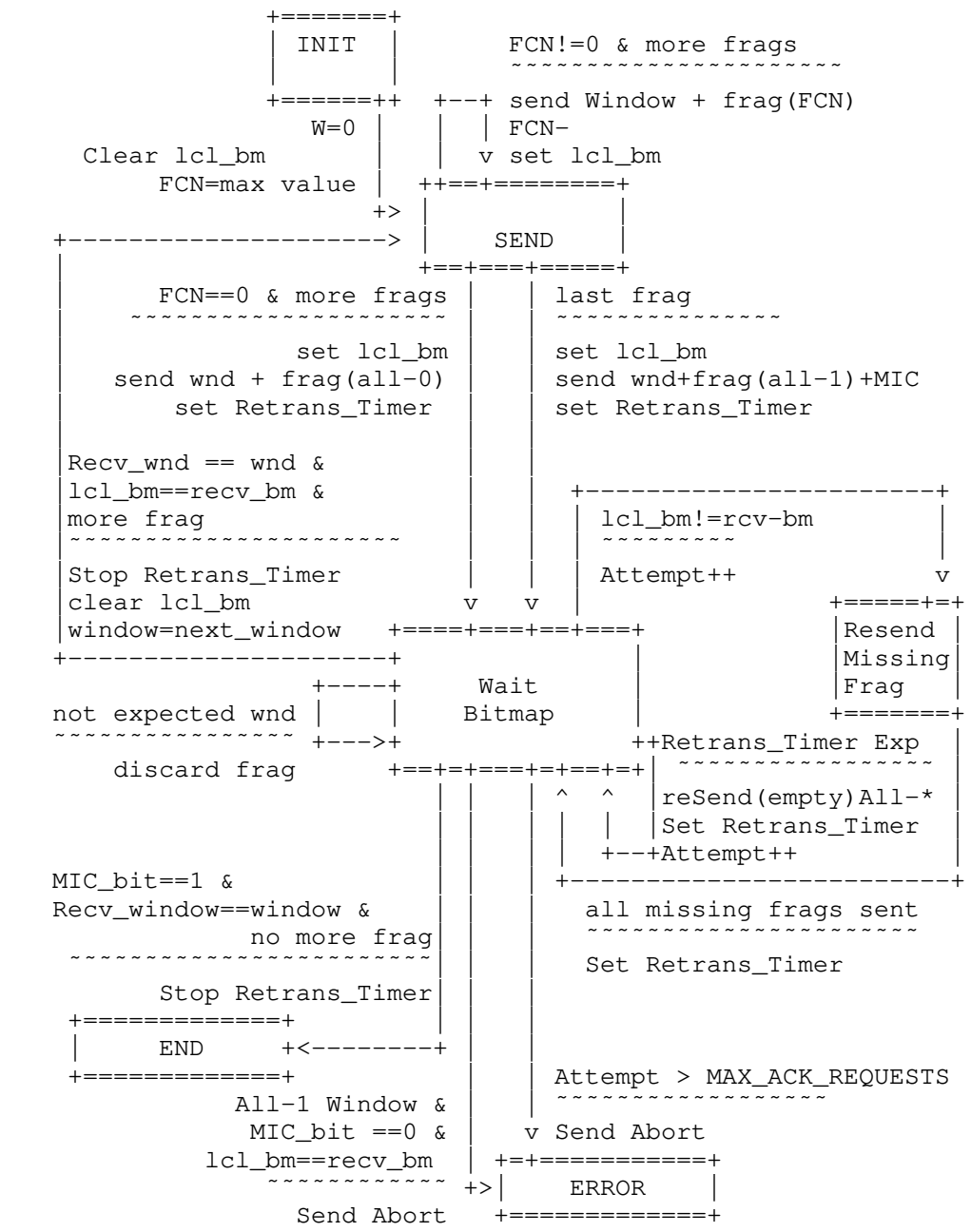
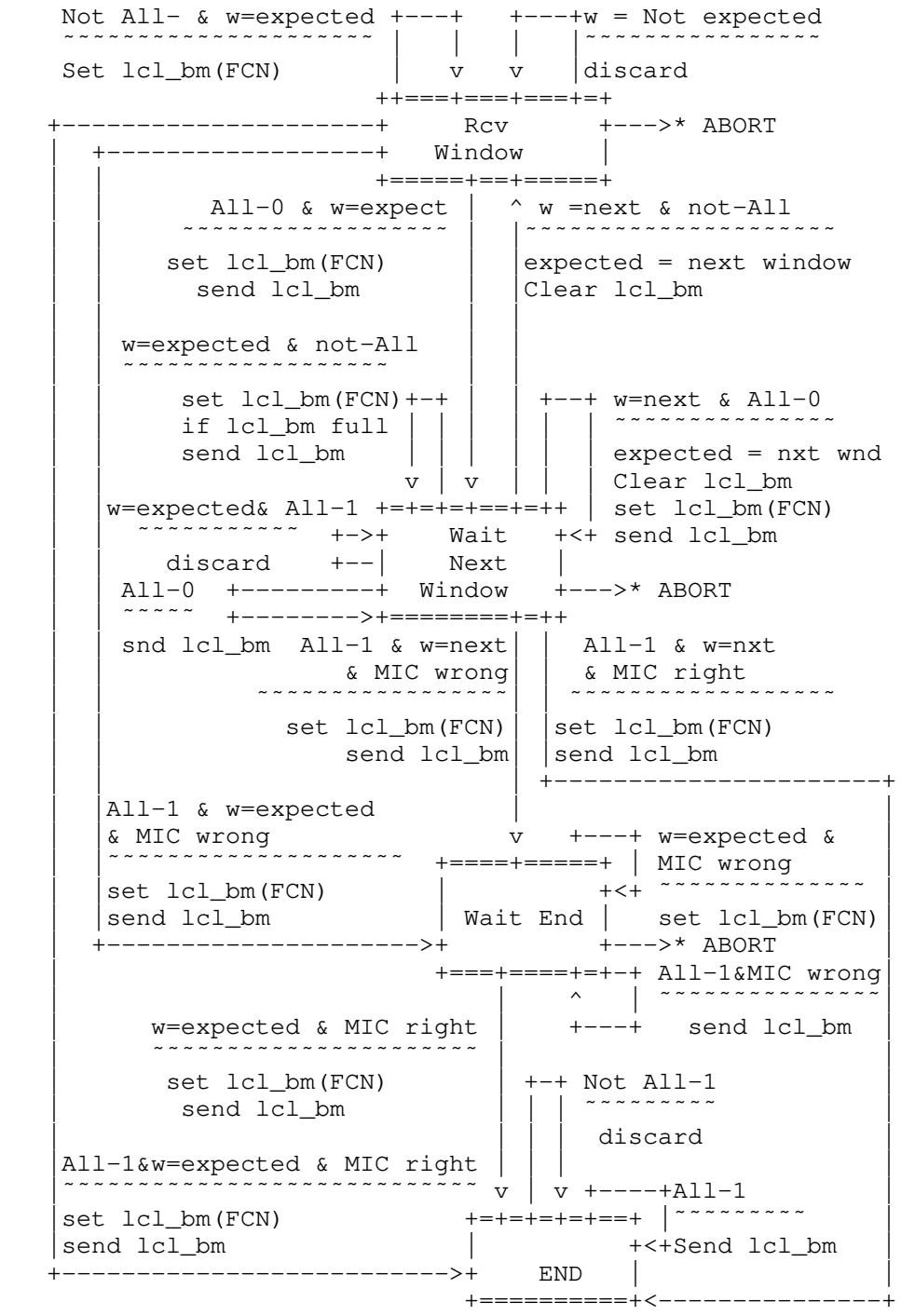
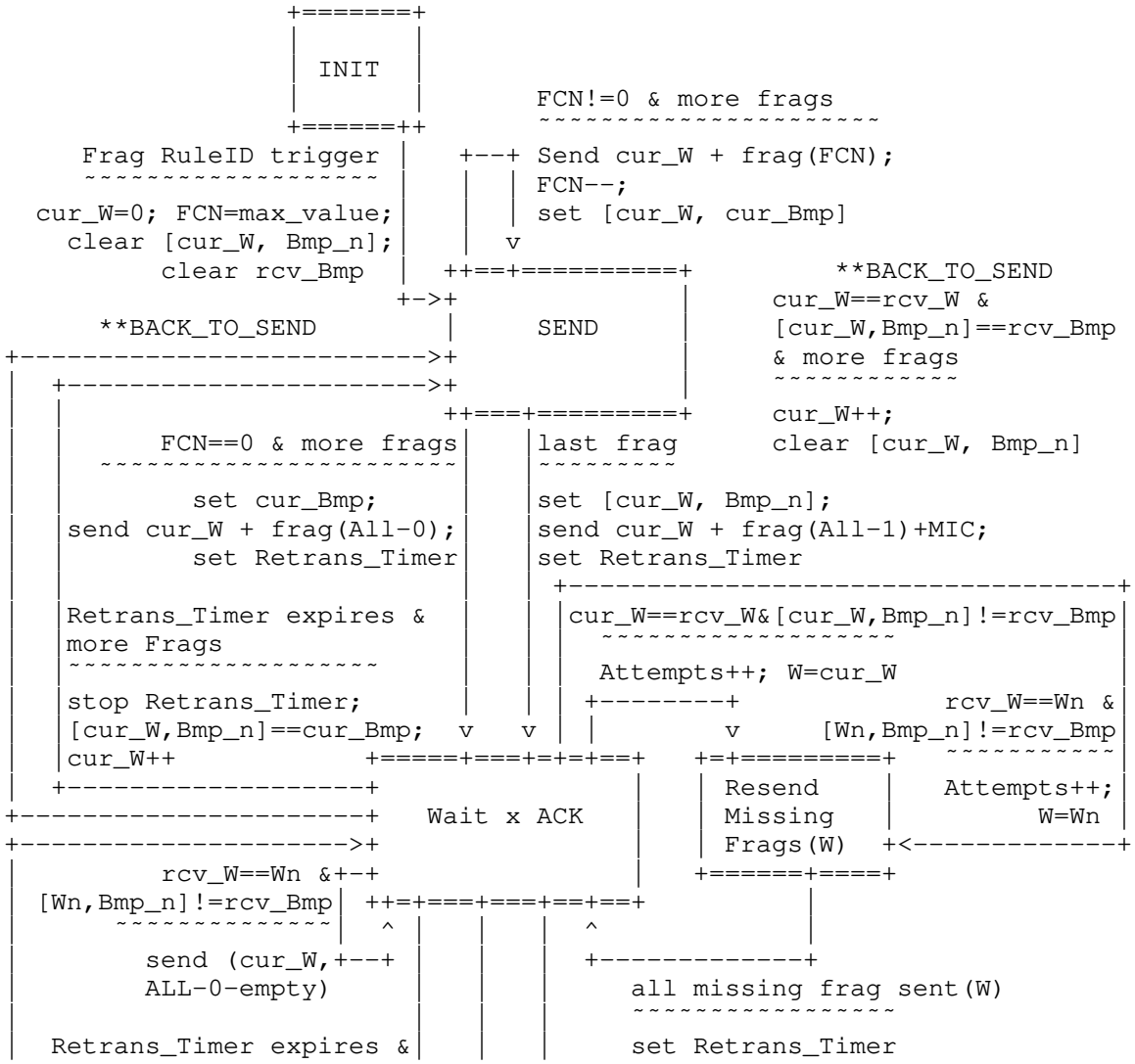


Figure 37: Sender State Machine for the ACK-Always Mode


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

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

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



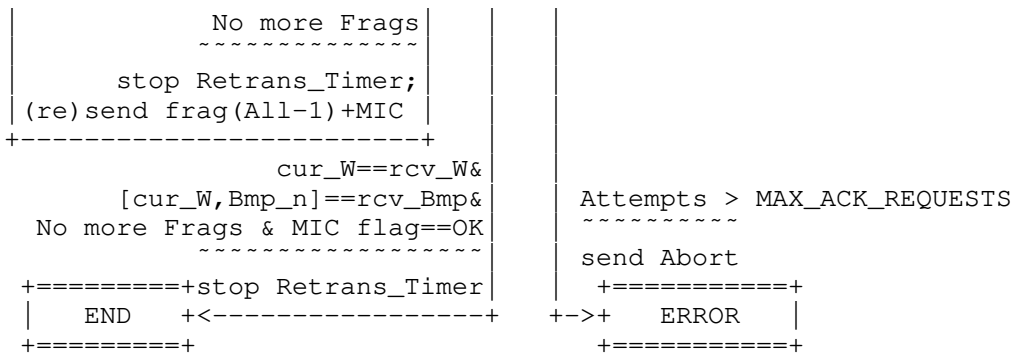
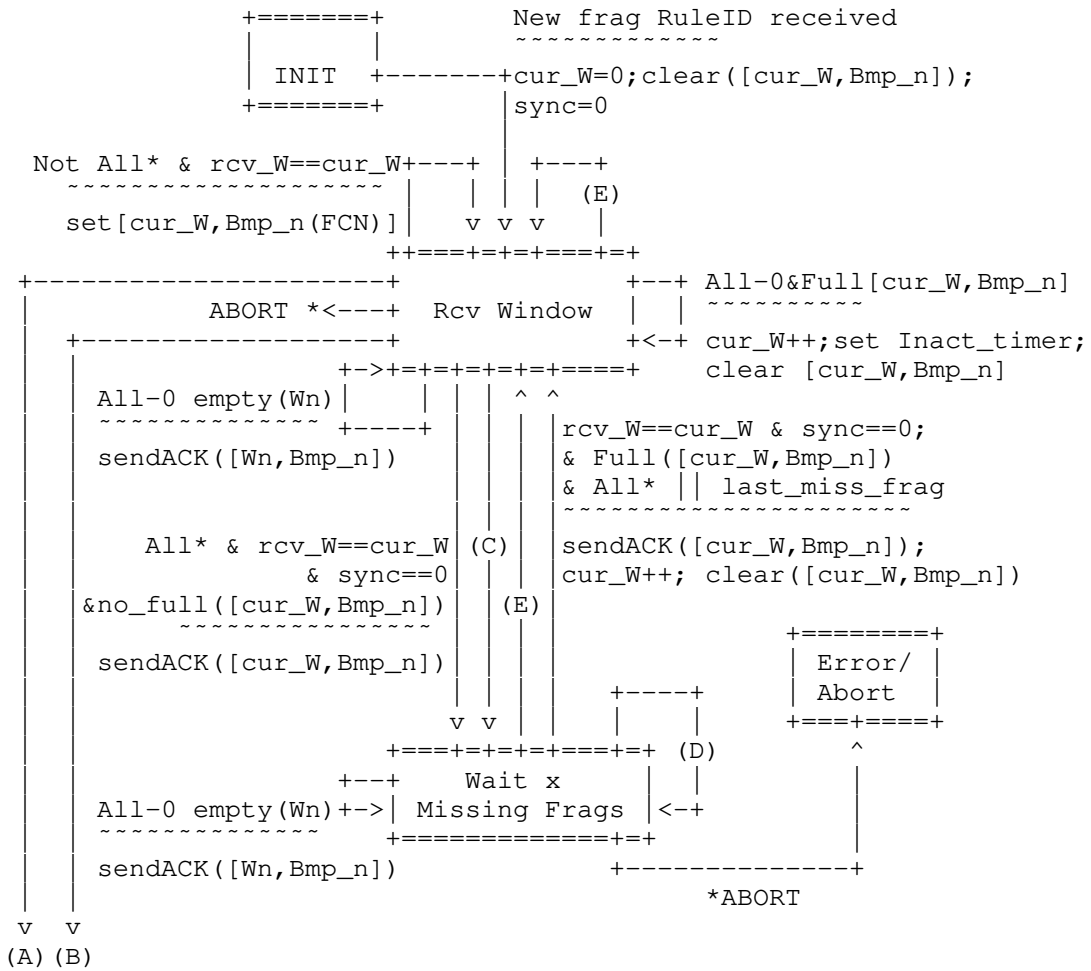


Figure 39: Sender State Machine for the ACK-on-Error Mode

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



(C) All* & sync>0

Wn=oldest[not full(W)];
sendACK([Wn,Bmp_n])

(D) All* || last_miss_frag
& rcv_W!=cur_W & sync>0
& Full([rcv_W,Bmp_n])

Wn=oldest[not full(W)];
sendACK([Wn,Bmp_n]);sync--

ABORT-->* Uplink Only &
Inact_Timer expires
|| Attempts > MAX_ACK_REQUESTS

send Abort

(E) Not All* & rcv_W!=cur_W

sync++; cur_W=rcv_W;
set[cur_W,Bmp_n(FCN)]

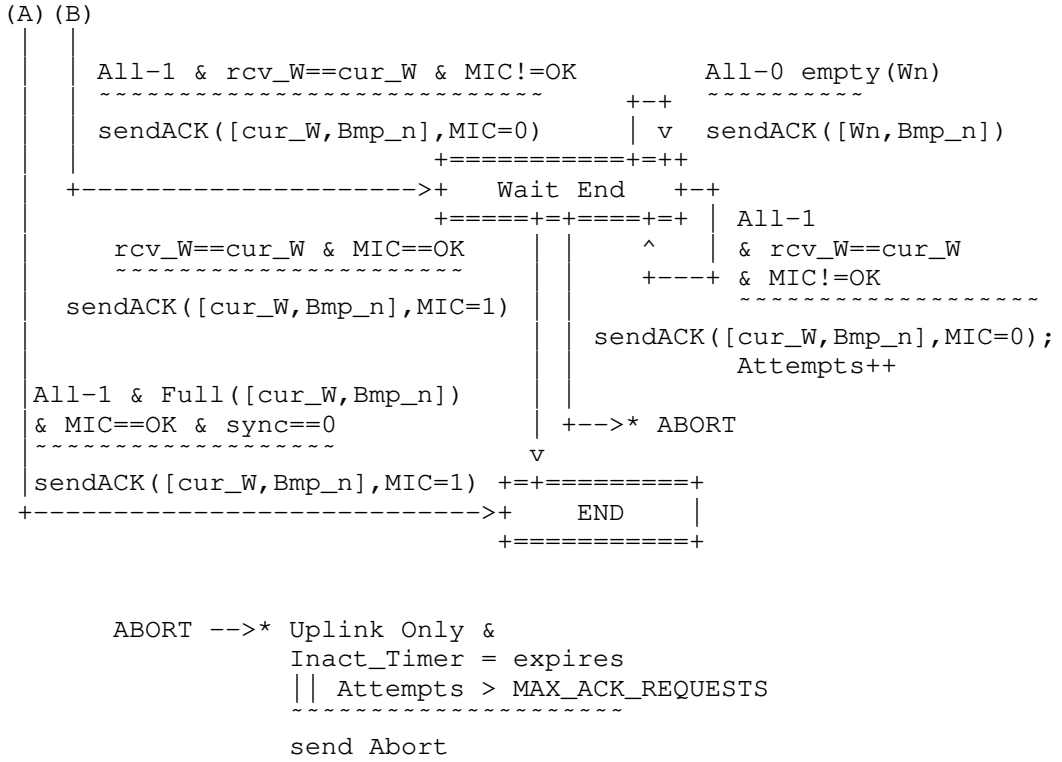


Figure 40: Receiver State Machine for the ACK-on-Error Mode

Appendix D. SCHC Parameters

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

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

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

- Rule ID numbering scheme, fixed-sized or variable-sized Rule IDs, number of Rules, the way the Rule ID is transmitted
- Padding: size of the L2 Word (for most LPWAN technologies, this would be a byte; for some technologies, a bit)
- Decision to use SCHC fragmentation mechanism or not. If yes:
  - reliability mode(s) used, in which cases (e.g. based on link channel condition)
  - Rule ID values assigned to each mode in use
  - presence and number of bits for DTag (T) for each Rule ID value
  - support for interleaved packet transmission, to what extent
  - WINDOW_SIZE, for modes that use windows
  - number of bits for W (M) for each Rule ID value, for modes that use windows
  - number of bits for FCN (N) for each Rule ID value
  - value of MAX_WIND_FCN and use of FCN values, if applicable to the SCHC F/R mode.
  - size of MIC and algorithm for its computation, for each Rule ID, if different from the default CRC32. Byte fill-up with zeroes or other mechanism, to be specified.
  - Retransmission Timer duration for each Rule ID value, if applicable to the SCHC F/R mode
  - Inactivity Timer duration for each Rule ID value, if applicable to the SCHC F/R mode
  - MAX_ACK_REQUEST value for each Rule ID value, if applicable to the SCHC F/R mode
- if L2 Word is wider than a bit and SCHC fragmentation is used, value of the padding bits (0 or 1). This is needed because the padding bits of the last fragment are included in the MIC computation.

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

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

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

  * The way the contexts are provisioned

  * The way the Rules as generated

Appendix E. Supporting multiple window sizes for fragmentation

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

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

Appendix F. Downlink SCHC Fragment transmission

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

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

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

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

Appendix G. Note

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

Authors' Addresses

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

Email: ana@ackl.io

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

Email: Laurent.Toutain@imt-atlantique.fr

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

Email: carlesgo@entel.upc.edu

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

Email: dominique.barthel@orange.com

Juan Carlos Zuniga
SIGFOX
425 rue Jean Rostand
Labege 31670
France

Email: JuanCarlos.Zuniga@sigfox.com

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

Table of Contents

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.

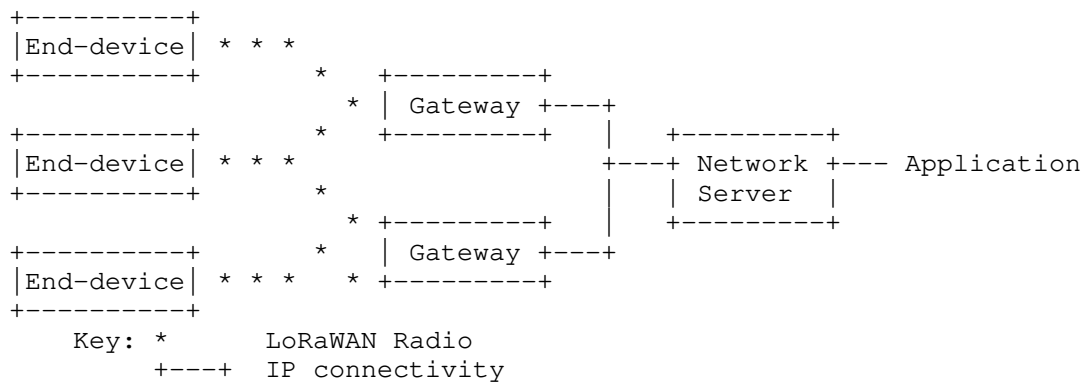


Figure 1: LoRaWAN architecture

- End-device: a LoRa client device, sometimes called a mote. Communicates with gateways.

- 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.

- 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.

- Join Server: The Join Server (JS) is a server on the Internet side of an NS that processes join requests from an end-devices.

- Uplink message: refers to communications from an end-device to a network server or application via one or more gateways.

- 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).

- 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.

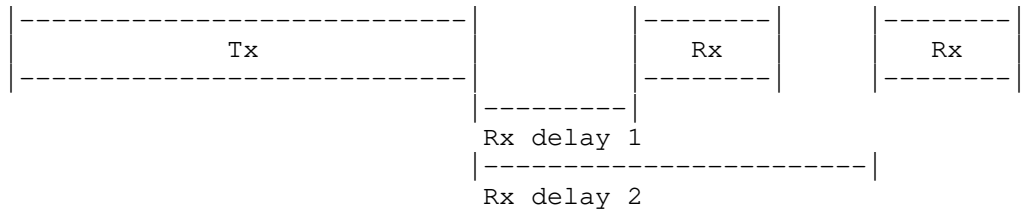


Figure 2: LoRaWAN Class A transmission and reception window

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, of 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].

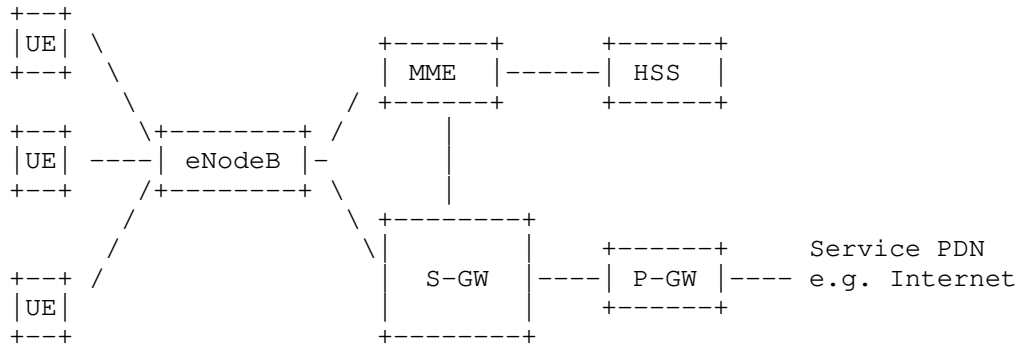


Figure 3: 3GPP network architecture

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.

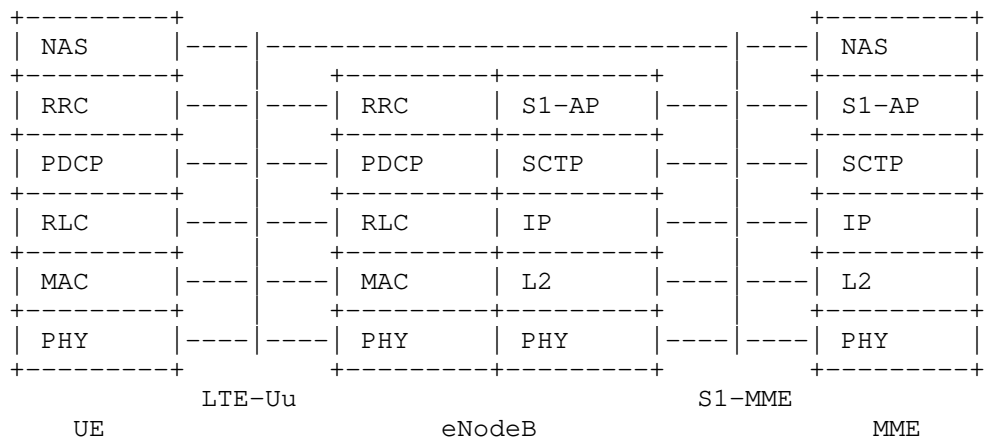


Figure 4: 3GPP radio protocol architecture for control plane

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:

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

- Modulation scheme: DBPSK
- Uplink transmission power: compliant with local regulation
- Link budget: 155 dB (or better)
- 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:

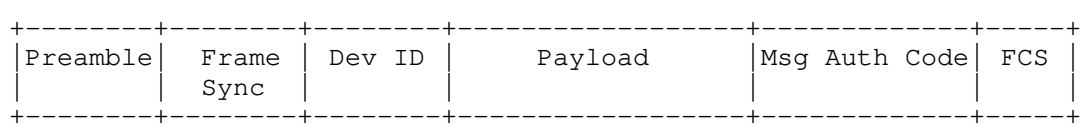


Figure 5: Uplink Frame Format

The uplink frame is composed of the following fields:

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

The main radio characteristics of the UNB downlink transmission are:

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

- 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:

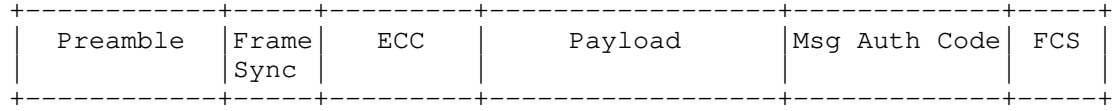


Figure 6: Downlink Frame Format

The downlink frame is composed of the following fields:

- Preamble: 91 bits
- Frame sync and header: 13 bits
- Error Correcting Code (ECC): 32 bits
- Payload: 0-64 bits
- Authentication: 16 bits
- 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.

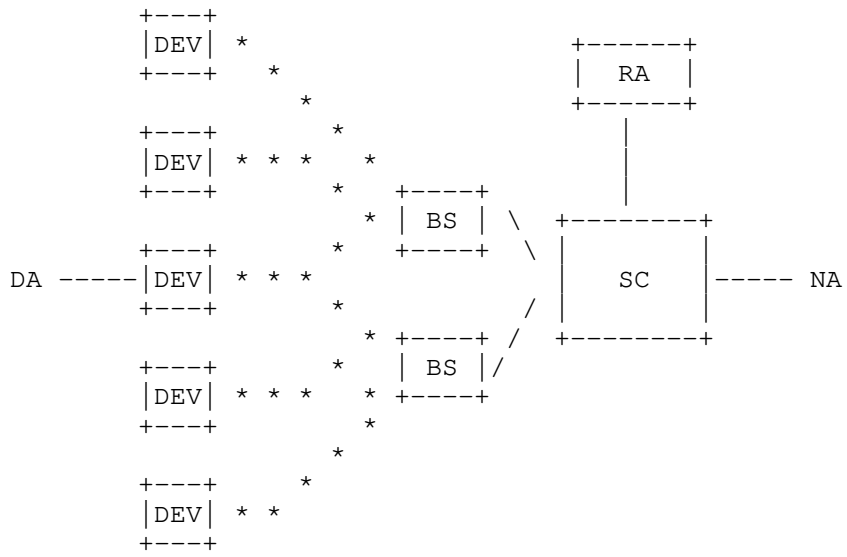


Figure 7: SIGFOX network architecture

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.

- 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.

- 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.

- 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.

- 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:

- 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.

- The Radio Gateway, which is the end point of the constrained link. It is known as: Gateway, Evolved Node B or Base station.

- 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.

- 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.

- 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

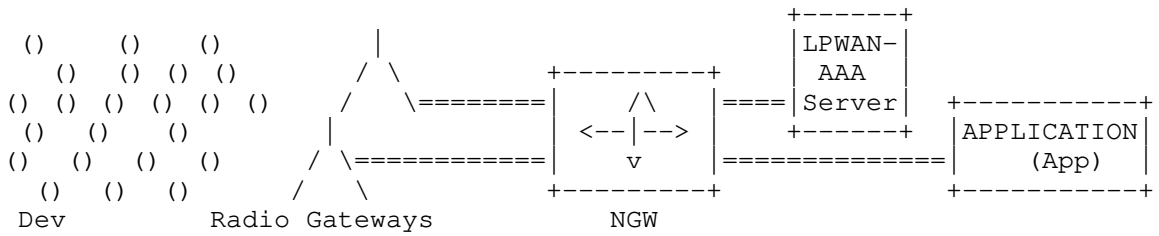


Figure 9: LPWAN Architecture

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:

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

- very low bit rate (from ~10 bit/s to ~100 kbit/s), and
- 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) 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:

- Text for Section 2.1 was provided by Alper Yegin and Stephen Farrell in [I-D.farrell-lpwan-lora-overview].

- Text for Section 2.2 was provided by Antti Ratilainen in [I-D.ratilainen-lpwan-nb-iot].

- Text for Section 2.3 was provided by Juan Carlos Zuniga and Benoit Ponsard in [I-D.zuniga-lpwan-sigfox-system-description].

- 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.

- 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
Labege 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
Labege 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] Tschofenig, 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

- WG have stated they want this to be an RFC.
- WG clearly want to keep the RF details.
- Various changes made to remove/resolve a number of editorial notes from -00 (in some cases as per suggestions from Ana Minaburo)
- Merged PR's: #1...
- Rejected PR's: #2 (change was made to .txt not .xml but was replicated manually by editor)
- Github repo is at: https://github.com/sftcd/lpwan-ov

A.2. From -01 to -02

- 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)
- Got new text wrt Wi-SUN via email from Paul Duffy and merged that in
- Reflected list discussion wrt terminology and "end-device"
- Merged PR's: #3...

A.3. From -02 to -03

- Editorial changes and typo fixes thanks to Fred Baker running something called Grammerly and sending me it's report.
- Merged PR's: #4, #6, #7...
- Editor did an editing pass on the lot.

A.4. From -03 to -04

- Picked up a PR that had been wrongly applied that expands UE
- Editorial changes wrt LoRa suggested by Alper
- Editorial changes wrt SIGFOX provided by Juan-Carlos

A.5. From -04 to -05

- Handled Russ Housley's WGLC review.
- Handled Alper Yegin's WGLC review.

A.6. From -05 to -06

- More Alper comments:-)
- Added some more detail about sigfox security.
- 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

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

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

July 02, 2018

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

Abstract

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

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

Status of This Memo

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

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

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

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

Table of Contents

1. Introduction

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

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

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

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

2. Terminology

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

o DevEUI: an IEEE EUI-64 identifier used to identify the device during the procedure while joining the network (Join Procedure)

o DevAddr: a 32-bit non-unique identifier assigned to a device statically or dynamically after a Join Procedure (depending on the activation mode)

o TBD: all significant LoRaWAN-related terms.

3. Static Context Header Compression Overview

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

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

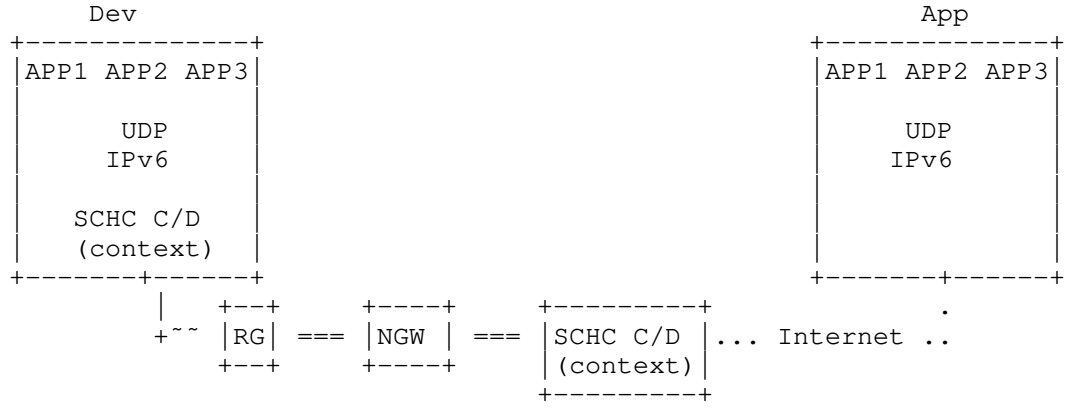


Figure 1: Architecture

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

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

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

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

4. LoRaWAN Architecture

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

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

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

o The Radio Gateway (RGW), which is the end point of the constrained link. This entity maps to the LoRaWAN Gateway.

o The Network Gateway (NGW) is the interconnection node between the Radio Gateway and the Internet. This entity maps to the LoRaWAN Network Server.

o LPWAN-AAA Server, which controls the user authentication and the applications. This entity maps to the LoRaWAN Join Server.

o Application Server (App). The same terminology is used in LoRaWAN.

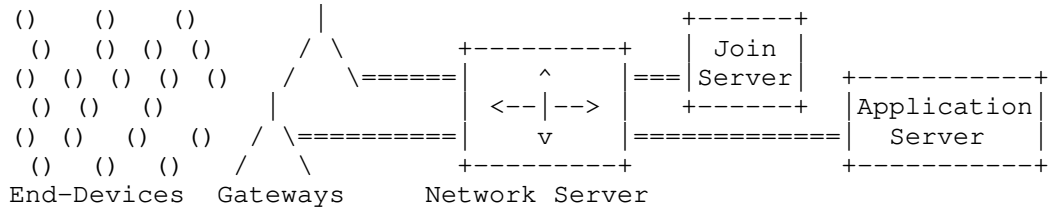


Figure 1: LPWAN/LoRaWAN Architecture

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

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

TBD

4.2. Device addressing

TBD

4.3. General Message Types

TBD

4.4. LoRaWAN MAC Frames

TBD

5. SCHC over LoRaWAN

5.1. Rule ID management

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

TBD

5.2. IID computation

TBD

5.3. Fragmentation

TBD

5.3.1. Reliability options

5.3.1.1. Uplinks: From device to gateway

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

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

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

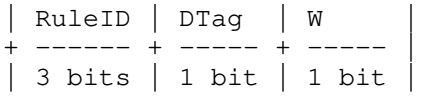


Figure 2: All fragment except the last one. Header size is 8 bits.

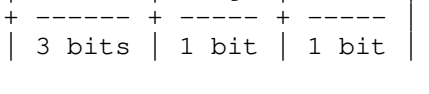


Figure 3: All-1 fragment detailed format for the last fragment. Header size is 8 bits.

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

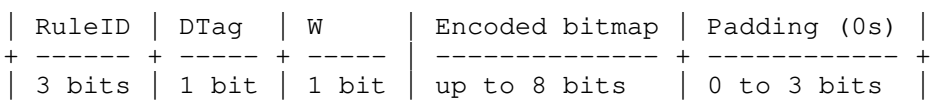


Figure 4: ACK format for All-0 windows. Header size is 1 or 2 bytes.

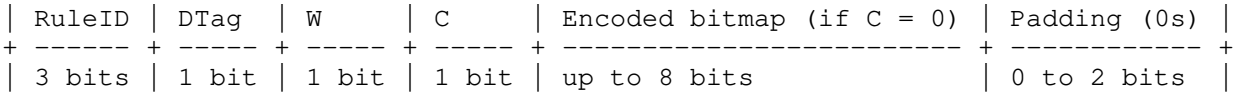


Figure 5: ACK format for All-1 windows. Header size is 1 or 2 bytes.

5.3.1.2. Downlinks: From gateway to device

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

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

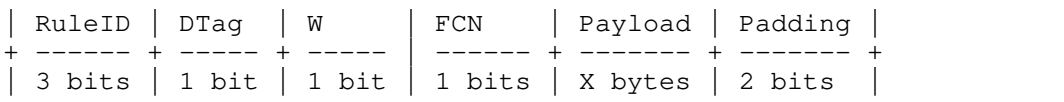


Figure 6: All fragments but the last one. Header size is 6 bits.

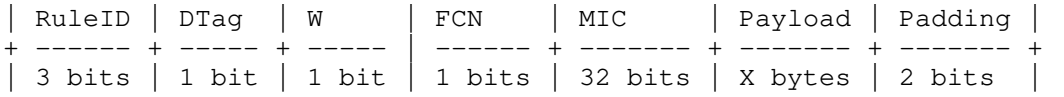


Figure 7: All-1 Fragment Detailed Format for the Last Fragment. Header size is 6 bits.

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

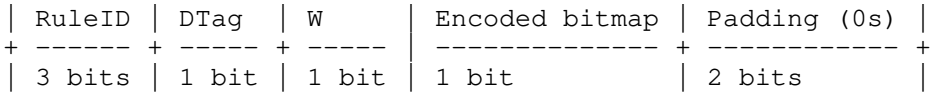


Figure 8: ACK format for All-0 windows. Header size is 8 bits.

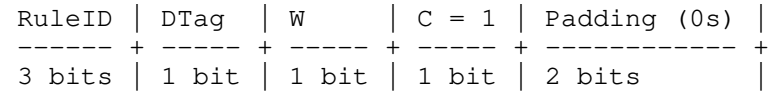


Figure 9: ACK format for All-1 windows, MIC is correct. Header size is 8 bits.

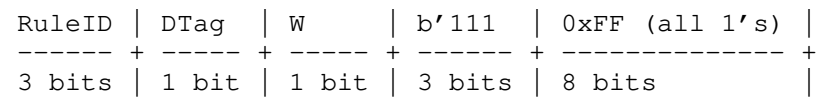


Figure 10: Receiver ABORT packet (following an all-1 packet with incorrect MIC). Header size is 16 bits.

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

5.3.1.2.1. Class A devices

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

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

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

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

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

5.3.1.3. Class B or C devices

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

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

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

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

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

5.3.2. Supporting multiple window sizes

TBD

5.3.3. Downlink fragment transmission

TBD

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

TBD

6. Security considerations

TBD

7. Acknowledgements

TBD

8. References

8.1. Normative References

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

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

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

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

8.2. Informative References

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

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

[lora-alliance-spec] Alliance, L., "LoRaWAN Specification Version V1.0.2", <http://portal.lora-alliance.org/DesktopModules/Inventures_Document/FileDownload.aspx?ContentID=1398>.

Appendix A. Examples

Appendix B. Note

Authors' Addresses

Nicolas Sornin (editor)
Semtech
14 Chemin des Clos
Meylan
France

Email: nsornin@semtech.com

Michael Coracin
Semtech
14 Chemin des Clos
Meylan
France

Email: mcoracin@semtech.com

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

Email: ivaylo@ackl.io

Alper Yegin
Actility
.
Paris, Paris
France

Email: alper.yegin@actility.com

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

Email: j.catalano@kerlink.fr

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

Email: vincent.audebert@edf.fr

lpwan Working Group
Internet-Draft
Intended status: Informational
Expires: May 9, 2019

JC. Zuniga
SIGFOX
C. Gomez
Universitat Politecnica de Catalunya
L. Toutain
IMT-Atlantique

November 05, 2018

SCHC over Sigfox LPWAN
draft-zuniga-lpwan-schc-over-sigfox-05

Abstract

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

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

Table of Contents

1. Introduction

The Static Context Header Compression (SCHC) specification [I-D.ietf-lpwan-ipv6-static-context-hc] defines a header compression scheme and a fragmentation functionality. Both can be used on top of all the LWPAN systems defined in [RFC8376] . These LPWAN systems have similar characteristics such as star-oriented topologies, network architecture, connected devices with built-in applications, etc.

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

This document describes the recommended parameters and modes of operation to be used when SCHC is implemented over a Sigfox LPWAN.

2. Terminology

It is assumed that the reader is familiar with the terms and mechanisms defined in [RFC8376] and in [I-D.ietf-lpwan-ipv6-static-context-hc].

3. Static Context Header Compression

The Static Context Header Compression (SCHC) described in [I-D.ietf-lpwan-ipv6-static-context-hc] takes advantage of the predictability of data flows existing in LPWAN networks to avoid context synchronization. Nonetheless, these contexts must be stored and configured on both ends. This can be done either by using a provisioning protocol, by out of band means, or by pre-provisioning them (for instance at manufacturing time). The way the contexts are configured and stored on both ends is out of the scope of this document.

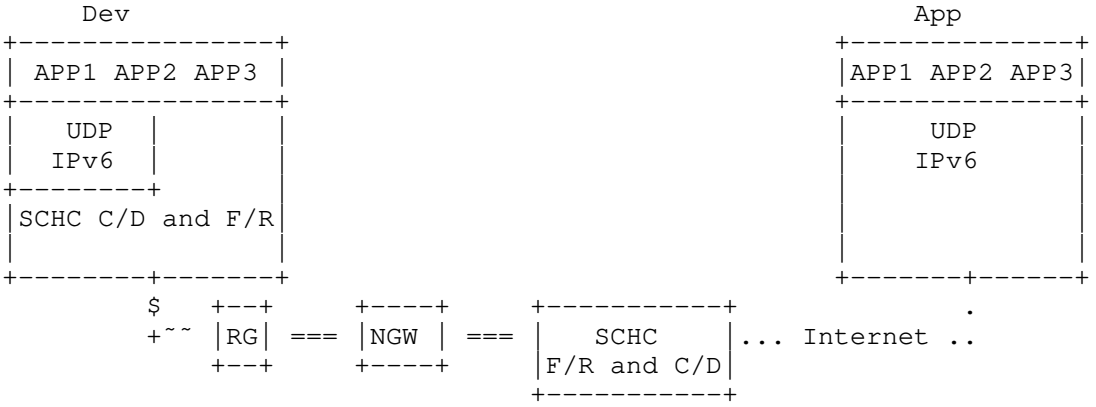


Figure 1: Architecture

Figure 1 represents the architecture for compression/decompression and fragmentation/reassembly, which is based on [RFC8376] terminology.

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

4. SCHC over Sigfox

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

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

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

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

4.1. SCHC Rules

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

4.2. Packet processing

TBD

5. Fragmentation

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

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

5.1. Fragmentation headers

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

5.2. Uplink fragment transmissions

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

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

5.2.1. Uplink No-ACK mode

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

The recommended Fragmentation Header size is 8 bits, and it is composed as follows:

The recommended Rule ID size is: 2 bits

The recommended DTag size (T) is: 2 bits

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

As per [I-D.ietf-lpwan-ipv6-static-context-hc], in the No-ACK mode the W (window) field is not present.

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

The algorithm for computing the MIC field MUST be TBD.

5.2.2. Uplink ACK-Always mode

TBD

5.2.3. Uplink ACK-on-Error mode

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

In the most generic case, the Fragmentation Header size is 8 bits and it is composed as follows:

The recommended Rule ID size is: 2 bits.

The recommended DTag size (T) is: 1 bit.

The recommended Window (W) size is: 2 bits.

Fragment Compressed Number (FCN) size (N): 3 bits.

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

The value of MAX_ACK_REQUESTS SHOULD be 2, and the value of MAX_WIND_FCN SHOULD be 6 (or 0b110, which allows a maximum window size of 7 fragments).

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

The algorithm for computing the MIC field MUST be TBD.

5.3. Downlink fragment transmissions

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

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

fragment that is not the last one. Such uplink transmission MAY be triggered by sending a SCHC message, such as a SCHC ACK.

For reliable downlink fragment transmission, the ACK-Always mode is RECOMMENDED.

The recommended Fragmentation Header size is: 8 bits

The recommended Rule ID size is: 2 bits.

The recommended DTag size (T) is: 2 bits.

Fragment Compressed Number (FCN) size (N): 3 bits.

As per [I-D.ietf-lpwan-ipv6-static-context-hc], in the ACK-Always mode a Window (W) 1-bit field must be present.

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

The value of MAX_ACK_REQUESTS SHOULD be 2, and the value of MAX_WIND_FCN SHOULD be 6 (or 0b110, which allows a maximum window size of 7 fragments).

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

The algorithm for computing the MIC field MUST be TBD.

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

6. Padding

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

Uplink frames can contain a payload from 0 to 96 bits (i.e. 12 bytes). The radio protocol allows sending zero bits or one single bit of information for binary applications (e.g. status). However, for 2 or more bits of payload it is required to add padding to the next integer number of bytes. The reason for this flexibility is to optimize transmission time and hence save battery consumption at the device.

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

7. Security considerations

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

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

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

8. Acknowledgements

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

9. Informative References

[I-D.ietf-lpwan-ipv6-static-context-hc]
Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "LPWAN Static Context Header Compression (SCHC) and fragmentation for IPv6 and UDP", draft-ietf-lpwan-ipv6-static-context-hc-17 (work in progress), October 2018.

[RFC8376] Farrell, S., Ed., "Low-Power Wide Area Network (LPWAN) Overview", RFC 8376, DOI 10.17487/RFC8376, May 2018, <https://www.rfc-editor.org/info/rfc8376>.

Authors' Addresses

Juan Carlos Zuniga
SIGFOX
425 rue Jean Rostand
Labege 31670
France

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

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

Email: carlesgo@entel.upc.edu

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

Email: Laurent.Toutain@imt-atlantique.fr