lpwan Working Group
Internet-Draft
Intended status: Informational
Expires: March 16, 2018

A. Minaburo
Acklio
L. Toutain
IMT-Atlantique
C. Gomez
Universitat Politecnica de Catalunya

September 12, 2017

LPWAN Static Context Header Compression (SCHC) and fragmentation for IPv6 and UDP
draft-ietf-lpwan-ipv6-static-context-hc-06

Abstract

This document describes a header compression scheme and fragmentation functionality for very low bandwidth networks. These techniques are especially tailored for LPWAN (Low Power Wide Area Network) networks.

The Static Context Header Compression (SCHC) offers a great level of flexibility when processing the header fields and must be used for these kind of networks. A common context stored in a LPWAN device and in the network is used. This context keeps information that will not be transmitted in the constrained network. Static context means that information stored in the context, which describes field values, does not change during packet transmission. This avoids complex resynchronization mechanisms, which are incompatible with LPWAN characteristics. In most cases, IPv6/UDP headers are reduced to a small identifier called Rule ID. But sometimes, a packet will not be compressed enough by SCHC to fit in one L2 PDU, and the SCHC fragmentation protocol will be used.

This document describes the SCHC compression/decompression framework and applies it to IPv6/UDP headers. Similar solutions for other protocols such as CoAP will be described in separate documents. Moreover, this document specifies a fragmentation and reassembly mechanism that is used in two situations: for SCHC-compressed packets that still exceed the L2 PDU size; and for the case where the SCHC compression cannot be performed.

Status of This Memo

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

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

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

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

This Internet-Draft will expire on March 16, 2018.

Table of Contents

1. Introduction

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

- Topology is star-oriented, therefore all the packets follow the same path. For the needs of this draft, the architecture can be summarized to Devices (Dev) exchanging information with LPWAN Application Server (App) through a Network Gateway (NGW).

- Traffic flows are mostly known in advance, since devices embed built-in applications. Contrary to computers or smartphones, new applications cannot be easily installed.

The Static Context Header Compression (SCHC) is defined for this environment. SCHC uses a context where header information is kept in the header format order. This context is static (the values on the header fields do not change over time) avoiding complex resynchronization mechanisms, incompatible with LPWAN characteristics. In most of the cases, IPv6/UDP headers are reduced to a small context identifier.

The SCHC header compression mechanism is independent from the specific LPWAN technology over which it will be used.

LPWAN technologies are also characterized, among others, by a very reduced data unit and/or payload size [I-D.ietf-lpwan-overview]. However, some of these technologies do not support layer two fragmentation, therefore the only option for them to support the IPv6 MTU requirement of 1280 bytes [RFC2460] is the use of a fragmentation protocol at the adaptation layer below IPv6. This draft defines also a fragmentation functionality to support the IPv6 MTU requirements over LPWAN technologies. Such functionality has been designed under the assumption that data unit reordering will not happen between the entity performing fragmentation and the entity performing reassembly.

2. LPWAN Architecture

LPWAN technologies have similar architectures but different terminology. 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 high density of devices per radio gateway.

o The Radio Gateway (RG), 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. We use the term LPWAN-AAA server because we are not assuming that this entity speaks RADIUS or Diameter as many/most AAA servers do, but equally we don't want to rule that out, as the functionality will be similar.

o Application Server (App)

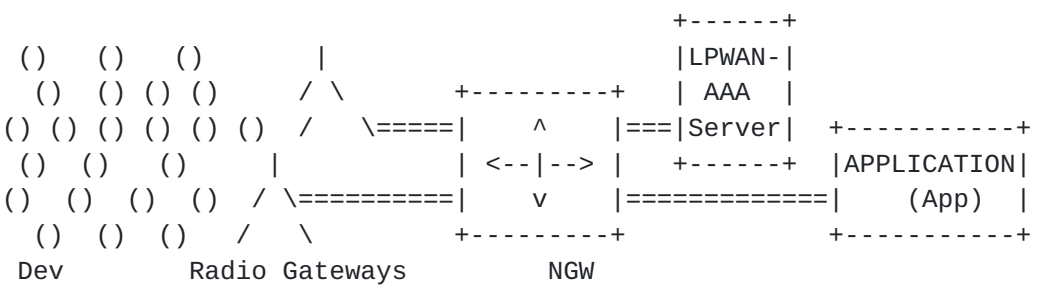


Figure 1: LPWAN Architecture

3. Terminology

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

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

- APP-IID: Application Interface Identifier. Second part of the IPv6 address to identify the application interface

- Bi: Bidirectional, it can be used in both senses

- CDA: Compression/Decompression Action. An action that is perfomed for both functionnalities to compress a header field or to recover its original value in the decompression phase.

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

- Dev: Device. Node connected to the LPWAN. A Dev may implement SCHC.

- Dev-IID: Device Interface Identifier. Second part of the IPv6 address to identify the device interface

- DI: Direction Indicator is a differentiator for matching in order to be able to have different values for both sides.

- DTag: Datagram Tag is a fragmentation header field that is set to the same value for all fragments carrying the same IPv6 datagram.

- Dw: Down Link direction for compression, from SCHC C/D to Dev

- FCN: Fragment Compressed Number is a fragmentation header field that carries an efficient representation of a larger-sized fragment number.

- FID: Field Indentifier is an index to describe the header fields in the Rule

- FP: Field Position is a list of possible correct values that a field may use

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

- MIC: Message Integrity Check. A fragmentation header field computed over an IPv6 packet before fragmentation, used for error detection after IPv6 packet reassembly.

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

- Rule: A set of header field values.

- Rule ID: An identifier for a rule, SCHC C/D and Dev share the same Rule ID for a specific flow.

- SCHC C/D: Static Context Header Compression Compressor/ Decompressor. A process in the network to achieve compression/ decompressing headers. SCHC C/D uses SCHC rules to perform compression and decompression.

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

- Up: Up Link direction for compression, from Dev to SCHC C/D.

- W: Window bit. A fragmentation header field used in Window mode (see section 9), which carries the same value for all fragments of a window.

4. Static Context Header Compression

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.

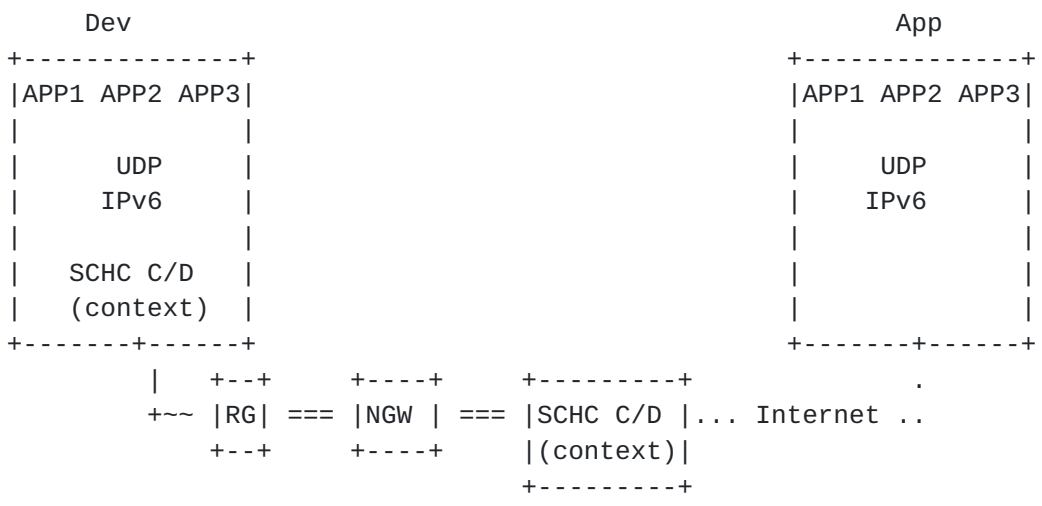


Figure 2: Architecture

Figure 2 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.

4.1. SCHC Rules

The main idea of the SCHC compression scheme is to send the Rule id to the other end instead of sending known field values. This Rule id identifies a rule that match as much as possible the original packet values. When a value is known by both ends, it is not necessary sent through the LPWAN network.

The context contains a list of rules (cf. Figure 3). Each Rule contains itself a list of fields descriptions composed of a field identifier (FID), a field position (FP), a direction indicator (DI), a target value (TV), a matching operator (MO) and a Compression/ Decompression Action (CDA).

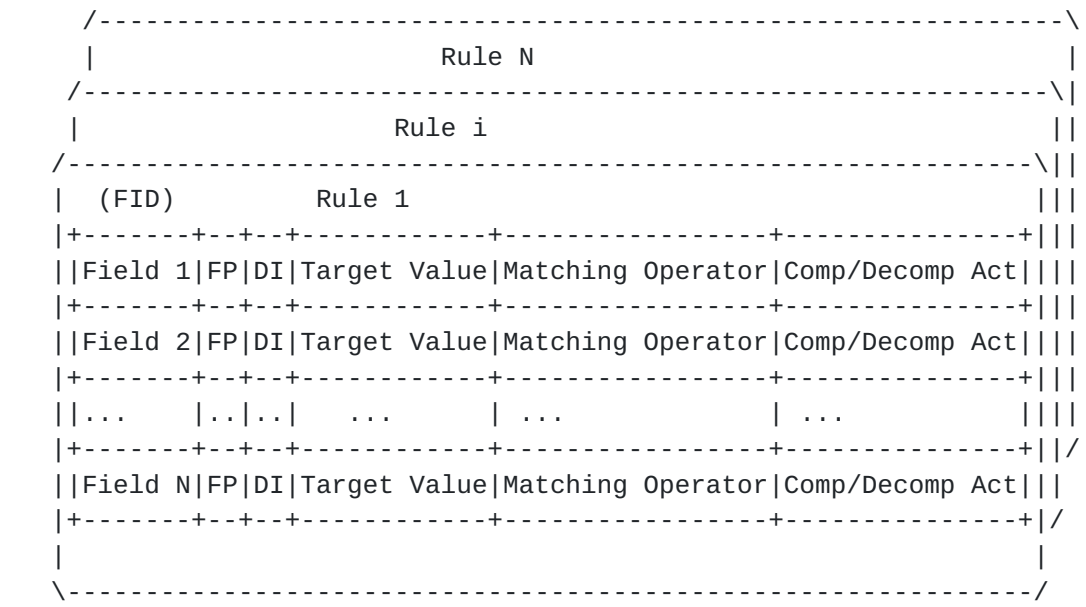


Figure 3: Compression/Decompression Context

The Rule does not describe the original packet format which must be known from the compressor/decompressor. The rule just describes the compression/decompression behavior for the header fields. In the rule, the description of the header field must be performed in the format packet order.

The Rule also describes the compressed header fields which are transmitted regarding their position in the rule which is used for data serialization on the compressor side and data deserialization on the decompressor side.

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

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

- A Field Position (FP) indicating if several instances of the field exist in the headers which one is targeted. The default position is 1

- A direction indicator (DI) indicating the packet direction. Three values are possible:

  * UP LINK (Up) when the field or the value is only present in packets sent by the Dev to the App,

  * DOWN LINK (Dw) when the field or the value is only present in packet sent from the App to the Dev and

  * BIDIRECTIONAL (Bi) when the field or the value is present either upstream or downstream.

- A Target Value (TV) is the value used to make the comparison with the packet header field. The Target Value can be of any type (integer, strings,...). For instance, it can be a single value or a more complex structure (array, list,...), such as a JSON or a CBOR structure.

- A Matching Operator (MO) is the operator used to make the comparison between the Field Value and the Target Value. The Matching Operator may require some parameters. MO is only used during the compression phase.

- A Compression Decompression Action (CDA) is used to describe the compression and the decompression process. The CDA may require some parameters, CDA are used in both compression and decompression phases.

4.2. Rule ID

Rule IDs are sent between both compression/decompression elements. The size of the Rule ID is not specified in this document, it is implementation-specific and can vary regarding the LPWAN technology, the number of flows, among others.

Some values in the Rule ID space may be reserved for goals other than header compression as fragmentation. (See Section 5).

Rule IDs are specific to a Dev. Two Devs may use the same Rule ID for different header compression. To identify the correct Rule ID, the SCHC C/D needs to combine the Rule ID with the Dev L2 identifier to find the appropriate Rule.

4.3. 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 doing compression from Dw to Up the SCHC C/D needs to find the correct Rule to use by identifying its Dev-ID and the Rule-ID. In the Up situation only the Rule-ID is used. The next step is to choose the fields by their direction, using the direction indicator (DI), so the fields that do not correspond to the appropriated DI will be excluded. Next, then the fields are identified according to their field identifier (FID) and field position (FP). If the field position does not correspond then the Rule is not use and

the SCHC take next Rule. Once the DI and the FP correspond to the header information, each 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 (MOs) of a Rule (i.e. all results are True), the fields of the header are then processed according to the Compression/ Decompression Actions (CDAs) and a compressed header is obtained. Otherwise the next rule is tested. If no eligible rule is found, then the header must be sent without compression, in which case the fragmentation process must be required.

- sending: The Rule ID is sent to the other end followed by information resulting from the compression of header fields, directly followed by the payload. The product of field compression is sent in the order expressed in the Rule for the matching fields. The way the Rule ID is sent depends on the specific LPWAN layer two technology and will be specified in a specific document, and is out of the scope of this document. For example, it can be either included in a Layer 2 header or sent in the first byte of the L2 payload. (cf. Figure 4).

- decompression: In both directions, The receiver identifies the sender through its device-id (e.g. MAC address) and selects the appropriate Rule through the Rule ID. This Rule gives the compressed header format and associates these values to the header fields. It applies the CDA action to reconstruct the original header fields. The CDA application order can be different of the order given by the Rule. For instance Compute-* may be applied at end, after the other CDAs.

  If after using SCHC compression and adding the payload to the L2 frame the datagram is not multiple of 8 bits, padding may be used.

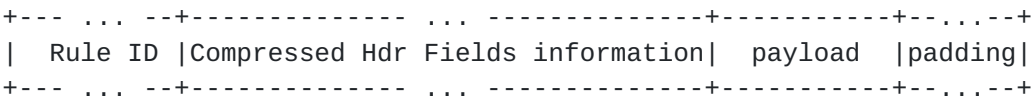


Figure 4: LPWAN Compressed Format Packet

4.4. 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 indifferently 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: A field value in a packet matches with a TV in a Rule if they are equal.

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

- MSB(length): A matching is obtained if the most significant bits of the length field value bits of the header are equal to the TV in the rule. The MSB Matching Operator needs a parameter, indicating the number of bits, to proceed to the matching.

- match-mapping: The goal of mapping-sent is to reduce the size of a field by allocating a shorter value. The Target Value contains a list of values. Each value is identified by a short ID (or index). This operator matches if a field value is equal to one of those target values.

4.5. 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 ctxt
value-sent	send	build from received value
mapping-sent	send index	value from index on a table
LSB(length)	send LSB	TV OR 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 5: Compression and Decompression Functions

Figure 5 sumarizes the basics functions defined to compress and decompress a field. The first column gives the action's name. The second and third columns outlines the compression/decompression behavior.

Compression is done in the rule order and compressed values are sent in that order in the compressed message. The receiver must be able

to find the size of each compressed field which can be given by the rule or may be sent with the compressed header.

If the field is identified as variable, then its size must be sent first using the following coding:

- If the size is between 0 and 14 bytes it is sent using 4 bits.
- For values between 15 and 255, the first 4 bit sent are set to 1 and the size is sent using 8 bits.
- For higher value, the first 12 bits are set to 1 and the size is sent on 2 bytes.

4.5.1. not-sent CDA

Not-sent function is generally used when the field value is specified in the rule and therefore known by the both Compressor and Decompressor. This action is generally used with the "equal" MO. If MO is "ignore", there is a risk to have a decompressed field value different from the compressed field.

The compressor does not send any value on the compressed header for the field on which compression is applied.

The decompressor restores the field value with the target value stored in the matched rule.

4.5.2. value-sent CDA

The value-sent action is generally used when the field value is not known by both Compressor and Decompressor. The value is sent 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 explicitly in the compressed header field by indicating the length. This function is generally used with the "ignore" MO.

4.5.3. mapping-sent

mapping-sent is used to send a smaller index associated to the list of values in the Target Value. This function is used together with the "match-mapping" MO.

The compressor looks in the TV to find the field value and send the corresponding index. The decompressor uses this index to restore the field value.

The number of bits sent is the minimal size to code all the possible indexes.

4.5.4. LSB CDA

LSB action is used to avoid sending the known part of the packet field header to the other end. This action is used together with the "MSB" MO. A length can be specified in the rule to indicate how many bits have to be sent. If not length is specified, the number of bits sent are the field length minus the bits length specified in the MSB MO.

The compressor sends the "length" Least Significant Bits. The decompressor combines the value received with the Target Value.

If this action is made on a variable length field, the remaining size in byte has to be sent before.

4.5.5. DEViid, APPiid CDA

These functions are used to process respectively the Dev and the App Interface Identifiers (Deviid and Appiid) of the IPv6 addresses. Appiid CDA is less common, since current LPWAN technologies frames contain a single address.

The IID value may be computed from the Device ID present in the Layer 2 header. The computation is specific for each LPWAN technology and may depend on the Device ID size.

In the downstream direction, these CDA may be used to determine the L2 addresses used by the LPWAN.

4.5.6. Compute-*

These classes of functions are used by the decompressor to compute the compressed field value based on received information. Compressed fields are elided during compression and reconstructed during decompression.

- compute-length: compute the length assigned to this field. For instance, regarding the field ID, this CDA may be used to compute IPv6 length or UDP length.

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

5. Fragmentation

5.1. Overview

Fragmentation supported in LPWAN is mandatory when the underlying LPWAN technology is not capable of fulfilling the IPv6 MTU requirement. Fragmentation is used after SCHC header compression when the size of the resulting compressed packet is larger than the L2 data unit maximum payload. In LPWAN technologies, the L2 data unit size typically varies from tens to hundreds of bytes. If the entire datagram fits within a single L2 data unit, the fragmentation mechanism is not used and the packet is sent unfragmented. Otherwise, the datagram does not fit a single L2 data unit, it SHALL be broken into fragments.

Moreover, LPWAN technologies impose some strict limitations on traffic; therefore it is desirable to enable optional fragment retransmission, while a single fragment loss should not lead to retransmitting the full datagram. On the other hand, in order to preserve energy, Devices are sleeping most of the time and may receive data during a short period of time after transmission. In order to adapt to the capabilities of various LPWAN technologies, this specification allows a gradation of fragment delivery reliability. This document does not make any decision with regard to which fragment delivery reliability option was used over a specific LPWAN technology.

An important consideration is that LPWAN networks typically follow the star topology, and therefore data unit reordering is not expected in such networks. This specification assumes that reordering will not happen between the entity performing fragmentation and the entity performing reassembly. This assumption allows to reduce complexity and overhead of the fragmentation mechanism.

5.2. Reliability options: definition

This specification defines the following three fragment delivery reliability options:

o No ACK

o Window mode - ACK "always"

o Window mode - ACK on error

The same reliability option MUST be used for all fragments of a packet. It is up to implementers and/or representatives of the underlying LPWAN technology to decide which reliability option to use

and whether the same reliability option applies to all IPv6 packets or not. Note that the reliability option to be used is not necessarily tied to the particular characteristics of the underlying L2 LPWAN technology (e.g. the No ACK reliability option may be used on top of an L2 LPWAN technology with symmetric characteristics for uplink and downlink).

In the No ACK option, the receiver MUST NOT issue acknowledgments (ACK).

In Window mode - ACK "always", an ACK is transmitted by the fragment receiver after a window of fragments have been sent. A window of fragments is a subset of the full set of fragments needed to carry an IPv6 packet. In this mode, the ACK informs the sender about received and/or missed fragments from the window of fragments. Upon receipt of an ACK that informs about any lost fragments, the sender retransmits the lost fragments. When an ACK is not received by the fragment sender, the latter retransmits a fragment, which serves as an ACK request. The maximum number of ACK requests is MAX_ACK_REQUESTS. The default value of MAX_ACK_REQUESTS is not stated in this document, and it is expected to be defined in other documents (e.g. technology- specific profiles).

In Window mode - ACK on error, an ACK is transmitted by the fragment receiver after a window of fragments have been sent, only if at least one of the fragments in the window has been lost. In this mode, the ACK informs the sender about received and/or missed fragments from the window of fragments. Upon receipt of an ACK that informs about any lost fragments, the sender retransmits the lost fragments. The maximum number of ACKs to be sent by the receiver for a specific window, denoted MAX_ACKS_PER_WINDOW, is not stated in this document, and it is expected to be defined in other documents (e.g. technology-specific profiles).

This document does not make any decision as to which fragment delivery reliability option(s) are supported by a specific LPWAN technology.

Examples of the different reliability options described are provided in Appendix A.

5.3. Reliability options: discussion

This section discusses the properties of each fragment delivery reliability option defined in the previous section.

No ACK is the most simple fragment delivery reliability option. With this option, the receiver does not generate overhead in the form of

ACKs. However, this option does not enhance delivery reliability beyond that offered by the underlying LPWAN technology.

The Window mode - ACK on error option is based on the optimistic expectation that the underlying links will offer relatively low L2 data unit loss probability. This option reduces the number of ACKs transmitted by the fragment receiver compared to the Window mode - ACK "always" option. This may be specially beneficial in asymmetric scenarios, e.g. where fragmented data are sent uplink and the underlying LPWAN technology downlink capacity or message rate is lower than the uplink one. However, if an ACK is lost, the sender assumes that all fragments covered by the ACK have been successfully delivered. In contrast, the Window mode - ACK "always" option does not suffer that issue, at the expense of an ACK overhead increase.

The Window mode - ACK "always" option provides flow control. In addition, it is able to handle long bursts of lost fragments, since detection of such events can be done before end of the IPv6 packet transmission, as long as the window size is short enough. However, such benefit comes at the expense of higher ACK overhead.

5.4. Tools

This subsection describes the different tools that are used to enable the described fragmentation functionality and the different reliability options supported. Each tool has a corresponding header field format that is defined in the next subsection. The list of tools follows:

o Rule ID. The Rule ID is used in fragments and in ACKs. The Rule ID in a fragment is set to a value that indicates that the data unit being carried is a fragment. This also allows to interleave non-fragmented IPv6 datagrams with fragments that carry a larger IPv6 datagram. Rule ID may also be used to signal which reliability option is in use for the IPv6 packet being carried. Rule ID may also be used to signal the window size if multiple sizes are supported (see 9.7). In an ACK, the Rule ID signals that the message this Rule ID is prepended to is an ACK.

o Fragment Compressed Number (FCN). The FCN is included in all fragments. This field can be understood as a truncated, efficient representation of a larger-sized fragment number, and does not carry an absolute fragment number. A special FCN value denotes the last fragment that carries a fragmented IPv6 packet. In Window mode, the FCN is augmented with the W bit, which has the purpose of avoiding possible ambiguity for the receiver that might arise under certain conditions.

o Datagram Tag (DTag). The DTag field, if present, is set to the same value for all fragments carrying the same IPv6 datagram, allows to interleave fragments that correspond to different IPv6 datagrams.

o Message Integrity Check (MIC). It is computed by the sender over the complete IPv6 packet before fragmentation by using the TBD algorithm. The MIC allows the receiver to check for errors in the reassembled IPv6 packet, while it also enables compressing the UDP checksum by use of SCHC.

o Bitmap. The bitmap is a sequence of bits included in the ACK for a given window, that provides feedback on whether each fragment of the current window has been received or not.

5.5. Formats

This section defines the fragment format, the fragmentation header formats, and the ACK format.

5.5.1. Fragment format

A fragment comprises a fragmentation header and a fragment payload, and conforms to the format shown in Figure 6. The fragment payload carries a subset of either an IPv6 packet after header compression or an IPv6 packet which could not be compressed. A fragment is the payload in the L2 protocol data unit (PDU).

Figure 6: Fragment format.

5.5.2. Fragmentation header formats

In the No ACK option, fragments except the last one SHALL contain the fragmentation header as defined in Figure 7. The total size of this fragmentation header is R bits.

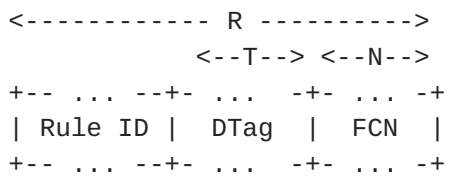


Figure 7: Fragmentation Header for Fragments except the Last One, No ACK option

In any of the Window mode options, fragments except the last one SHALL
contain the fragmentation header as defined in Figure 8. The total size of this fragmentation header is R bits.

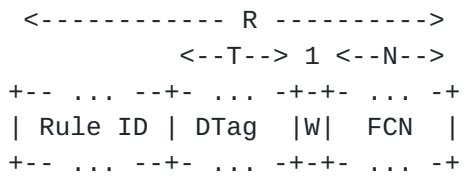


Figure 8: Fragmentation Header for Fragments except the Last One, Window mode

In the No ACK option, the last fragment of an IPv6 datagram SHALL contain a fragmentation header that conforms to the format shown in Figure 9. The total size of this fragmentation header is R+M bits.

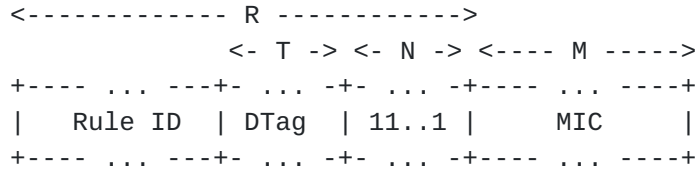


Figure 9: Fragmentation Header for the Last Fragment, No ACK option

In any of the Window modes, the last fragment of an IPv6 datagram SHALL contain a fragmentation header that conforms to the format shown in Figure 10. The total size of this fragmentation header is R+M bits.

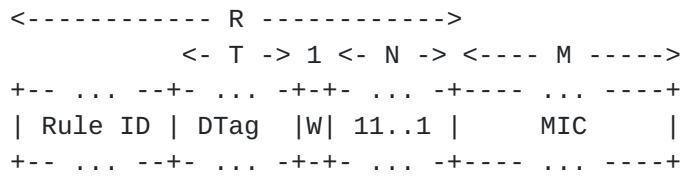


Figure 10: Fragmentation Header for the Last Fragment, Window mode

- Rule ID: This field has a size of R - T - N - 1 bits when Window mode is used. In No ACK mode, the Rule ID field has a size of R - T - N bits.

- DTag: The size of the DTag field is T bits, which may be set to a value greater than or equal to 0 bits. The DTag field in all fragments that carry the same IPv6 datagram MUST be set to the

same value. DTag MUST be set sequentially increasing from 0 to 2^T - 1, and MUST wrap back from 2^T - 1 to 0.

- FCN: This field is an unsigned integer, with a size of N bits, that carries the FCN of the fragment. In the No ACK option, N=1. For the rest of options, N equal to or greater than 3 is recommended. The FCN MUST be set sequentially decreasing from the highest FCN in the window (which will be used for the first fragment), and MUST wrap from 0 back to the highest FCN in the window. The highest FCN in the window, denoted MAX_WIND_FCN, MUST be a value equal to or smaller than 2^N-2, see further details on this at the end of 9.5.3. (Example 1: for N=5, MAX_WIND_FCN may be configured to be 30, then subsequent FCNs are set sequentially and in decreasing order, and FCN will wrap from 0 back to 30. Example 2: for N=5, MAX_WIND_FCN may be set to 23, then subsequent FCNs are set sequentially and in decreasing order, and the FCN will wrap from 0 back to 23). The FCN for the last fragment has all bits set to 1. Note that, by this definition, the FCN value of 2^N - 1 is only used to identify a fragment as the last fragment carrying a subset of the IPv6 packet being transported, and thus the FCN does not correspond to the N least significant bits of the actual absolute fragment number. It is also important to note that, for N=1, the last fragment of the packet will carry a FCN equal to 1, while all previous fragments will carry a FCN of 0.

- W: W is a 1-bit field. This field carries the same value for all fragments of a window, and it is complemented for the next window. The initial value for this field is 1.

- MIC: This field, which has a size of M bits, carries the MIC for the IPv6 packet.

The values for R, N, MAX_WIND_FCN, T and M are not specified in this document, and have to be determined in other documents (e.g. technology-specific profile documents).

5.5.3. ACK format

The format of an ACK is shown in Figure 11:

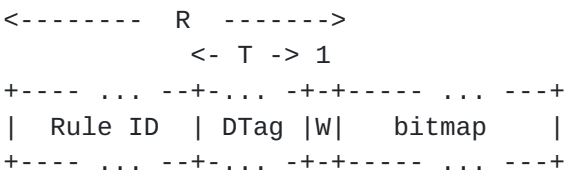


Figure 11: Format of an ACK

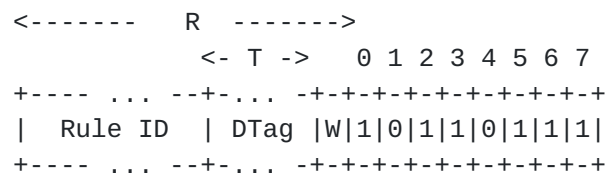
Rule ID: In all ACKs, Rule ID has a size of R - T - 1 bits.

DTag: DTag has a size of T bits. DTag carries the same value as the DTag field in the fragments carrying the IPv6 datagram for which this ACK is intended.

W: This field has a size of 1 bit. In all ACKs, the W bit carries the same value as the W bit carried by the fragments whose reception is being positively or negatively acknowledged by the ACK.

bitmap: This field carries the bitmap sent by the receiver to inform the sender about whether fragments in the current window have been received or not. Size of the bitmap field of an ACK can be equal to 0 or Ceiling(Number_of_Fragments/8) octets, where Number_of_Fragments denotes the number of fragments of a window. The bitmap is a sequence of bits, where the n-th bit signals whether the n-th fragment transmitted in the current window has been correctly received (n-th bit set to 1) or not (n-th bit set to 0). Remaining bits with bit order greater than the number of fragments sent (as determined by the receiver) are set to 0, except for the last bit in the bitmap, which is set to 1 if the last fragment of the window has been correctly received, and 0 otherwise. Feedback on reception of the fragment with FCN = 2^N - 1 (last fragment carrying an IPv6 packet) is only given by the last bit of the corresponding bitmap. Absence of the bitmap in an ACK confirms correct reception of all fragments to be acknowledged by means of the ACK. Note that absence of the bitmap in an ACK may be determined based on the size of the L2 payload.

Figure 12 shows an example of an ACK (N=3), where the bitmap indicates that the second and the fifth fragments have not been correctly received.




```
            <- T ->   0 1 2 3 4 5 6 7
+---- ... --+-... -+-+-+-+-+-+-+-+-+-+
|  Rule ID  | DTag |W|1|0|1|1|0|1|1|1|
+---- ... --+-... -+-+-+-+-+-+-+-+-+-+
```

Figure 12: Example of the bitmap in an ACK (in Window mode, for N=3)

Figure 13 illustrates an ACK without a bitmap.

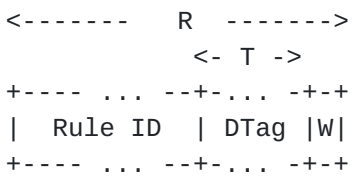


Figure 13: Example of an ACK without a bitmap

Note that, in order to exploit the available L2 payload space to the fullest, a bitmap may have a size smaller than 2^N bits. In that case, the window in use will have a size lower than 2^N-1 fragments. For example, if the maximum available space for a bitmap is 56 bits, N can be set to 6, and the window size can be set to a maximum of 56 fragments, thus MAX_WIND_FCN will be equal to 55 in this example.

5.6. Baseline mechanism

The receiver of link fragments SHALL use (1) the sender's L2 source address (if present), (2) the destination's L2 address (if present), (3) Rule ID and (4) DTag (the latter, if present) to identify all the fragments that belong to a given IPv6 datagram. The fragment receiver may determine the fragment delivery reliability option in use for the fragment based on the Rule ID field in that fragment.

Upon receipt of a link fragment, the receiver starts constructing the original unfragmented packet. It uses the FCN and the order of arrival of each fragment to determine the location of the individual fragments within the original unfragmented packet. For example, it may place the data payload of the fragments within a payload datagram reassembly buffer at the location determined from the FCN and order of arrival of the fragments, and the fragment payload sizes. In Window mode, the fragment receiver also uses the W bit in the received fragments. Note that the size of the original, unfragmented IPv6 packet cannot be determined from fragmentation headers.

When Window mode - ACK on error is used, the fragment receiver starts a timer (denoted "ACK on Error Timer") upon reception of the first fragment for an IPv6 datagram. The initial value for this timer is not provided by this specification, and is expected to be defined in additional documents. This timer is reset and restarted every time that a new fragment carrying data from the same IPv6 datagram is received. In Window mode - ACK on error, after reception of the last fragment of a window (i.e. the fragment with FCN=0 or FCN=2^N-1), if fragment losses have been detected by the fragment receiver in the current window, the fragment receiver MUST transmit an ACK reporting its available information with regard to successfully received and missing fragments from the current window. Upon expiration of the

"ACK on Error Timer", an ACK MUST be transmitted by the fragment receiver to report received and not received fragments for the current window. The "ACK on Error Timer" is then reset and restarted. When the last fragment of the IPv6 datagram is received, if all fragments of that last window of the packet have been received, the "ACK on Error Timer" is stopped. In Window mode - ACK on error, the fragment sender retransmits any lost fragments reported in an ACK. The maximum number of ACKs to be sent by the receiver for a specific window, denoted MAX_ACKS_PER_WINDOW, is not stated in this document, and it is expected to be defined in other documents (e.g. technology-specific profiles). In Window mode - ACK on error, when a fragment sender has transmitted the last fragment of a window, or it has retransmitted the last fragment within the set of lost fragments reported in an ACK, it is assumed that the time the fragment sender will wait to receive an ACK is smaller than the transmission time of MAX_WIND_FCN + 1 fragments (i.e. the time required to transmit a complete window of fragments). This aspect must be carefully considered if Window mode - ACK on error is used, in particular taking into account the latency characteristics of the underlying L2 technology.

Note that, in Window mode, the first fragment of the window is the one with FCN set to MAX_WIND_FCN. Also note that, in Window mode, the fragment with FCN=0 is considered the last fragment of its window, except for the last fragment of the whole packet (with all FCN bits set to 1, i.e. FCN=2^N-1), which is also the last fragment of the last window.

If Window mode - ACK "always" is used, upon receipt of the last fragment of a window (i.e. the fragment with CFN=0 or CFN=2^N-1), or upon receipt of the last retransmitted fragment from the set of lost fragments reported by the last ACK sent by the fragment receiver (if any), the fragment receiver MUST send an ACK to the fragment sender. The ACK provides feedback on the fragments received and those not received that correspond to the last window. Once all fragments of a window have been received by the fragment receiver (including retransmitted fragments, if any), the latter sends an ACK without a bitmap to the sender, in order to report successful reception of all fragments of the window to the fragment sender.

When Window mode - ACK "always" is used, the fragment sender starts a timer (denoted "ACK Always Timer") after the first transmission attempt of the last fragment of a window (i.e. the fragment with FCN=0 or FCN=2^N-1). In the same reliability option, if one or more fragments are reported by an ACK to be lost, the sender retransmits those fragments and starts the "ACK Always Timer" after the last retransmitted fragment (i.e. the fragment with the lowest FCN) among the set of lost fragments reported by the ACK. The initial value for

the "ACK Always Timer" is not provided by this specification, and it is expected to be defined in additional documents. Upon expiration of the timer, if no ACK has been received since the timer start, the next action to be performed by the fragment sender depends on whether the current window is the last window of the IPv6 packet or not. If the current window is not the last one, the sender retransmits the last fragment sent at the moment of timer expiration (which may or may not be the fragment with FCN=0), and it reinitializes and restarts the timer. Otherwise (i.e. the current window is the last one), the sender retransmits the fragment with FCN=2^N-1; if the fragment sender knows that the fragment with FCN=2^N-1 has already been successfully received, the fragment sender MAY opt to send a fragment with FCN=2^N-1 and without a data payload. Note that retransmitting a fragment sent as described serves as an ACK request. The maximum number of requests for a specific ACK, denoted MAX_ACK_REQUESTS, is not stated in this document, and it is expected to be defined in other documents (e.g. technology-specific profiles). In Window mode - ACK "Always", the fragment sender retransmits any lost fragments reported in an ACK. When the fragment sender receives an ACK that confirms correct reception of all fragments of a window, if there are further fragments to be sent for the same IPv6 datagram, the fragment sender proceeds to transmitting subsequent fragments of the next window.

If the recipient receives the last fragment of an IPv6 datagram (i.e. the fragment with FCN=2^N-1), it checks for the integrity of the reassembled IPv6 datagram, based on the MIC received. In No ACK, if the integrity check indicates that the reassembled IPv6 datagram does not match the original IPv6 datagram (prior to fragmentation), the reassembled IPv6 datagram MUST be discarded. In Window mode, a MIC check is also performed by the fragment receiver after reception of each subsequent fragment retransmitted after the first MIC check. In Window mode - ACK "always", if a MIC check indicates that the IPv6 datagram has been successfully reassembled, the fragment receiver sends an ACK without a bitmap to the fragment sender. In the same reliability option, after receiving a fragment with FCN=2^N-1, the fragment receiver sends an ACK to the fragment sender, even if it is not the first fragment with FCN=2^N-1 received by the fragment receiver.

If a fragment recipient disassociates from its L2 network, the recipient MUST discard all link fragments of all partially reassembled payload datagrams, and fragment senders MUST discard all not yet transmitted link fragments of all partially transmitted payload (e.g., IPv6) datagrams. Similarly, when either end of the LPWAN link first receives a fragment of a packet, it starts a reassembly timer. When this time expires, if the entire packet has not been reassembled, the existing fragments MUST be discarded and

the reassembly state MUST be flushed. The value for this timer is not provided by this specification, and is expected to be defined in technology-specific profile documents.

5.7. Supporting multiple window sizes

For Window mode operation, 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 may be used for IPv6 packets that need to be carried by a large number of fragments. However, when the number of fragments required to carry an IPv6 packet is low, a smaller window size, and thus a shorter bitmap, may be sufficient to provide feedback on all fragments. If multiple window sizes are supported, the Rule ID may be used to signal the window size in use for a specific IPv6 packet transmission.

5.8. Aborting fragmented IPv6 datagram transmissions

For several reasons, a fragment sender or a fragment receiver may want to abort the on-going transmission of one or several fragmented IPv6 datagrams. The entity (either the fragment sender or the fragment receiver) that triggers abortion transmits to the other endpoint a data unit with an L2 payload that only comprises a Rule ID (of size R bits), which signals abortion of all on-going fragmented IPv6 packet transmissions. The specific value to be used for the Rule ID of this abortion signal is not defined in this document, and is expected to be defined in future documents.

Upon transmission or reception of the abortion signal, both entities MUST release any resources allocated for the fragmented IPv6 datagram transmissions being aborted.

5.9. Downlink fragment transmission

In some LPWAN technologies, as part of energy-saving techniques, downlink transmission is only possible immediately after an uplink transmission. In order to avoid potentially high delay for fragmented IPv6 datagram transmission in the downlink, the fragment receiver MAY perform an uplink transmission as soon as possible after reception of a fragment that is not the last one. Such uplink transmission may be triggered by the L2 (e.g. an L2 ACK sent in response to a fragment encapsulated in a L2 frame that requires an L2 ACK) or it may be triggered from an upper layer.

6. SCHC Compression for IPv6 and UDP headers

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

6.1. IPv6 version field

This field always holds the same value, therefore the TV is 6, the MO is "equal" and the "CDA "not-sent"".

6.2. IPv6 Traffic class field

If the DiffServ field identified by the rest of the rule do not vary and is known by both sides, the TV should contain this well-known value, the MO should be "equal" and the CDA must be "not-sent.

If the DiffServ field identified by the rest of the rule varies over time or is not known by both sides, then there are two possibilities depending on the variability of the value, the first one is to do not compressed the field and sends the original value, or the second where the values can be computed by sending only the LSB bits:

- TV is not set to any value, MO is set to "ignore" and CDA is set to "value-sent"

- TV contains a stable value, MO is MSB(X) and CDA is set to LSB

6.3. Flow label field

If the Flow Label field identified by the rest of the rule does not vary and is known by both sides, the TV should contain this well-known value, the MO should be "equal" and the CDA should be "not-sent".

If the Flow Label field identified by the rest of the rule varies during time or is not known by both sides, there are two possibilities depending on the variability of the value, the first one is without compression and then the value is sent and the second where only part of the value is sent and the decompressor needs to compute the original value:

- TV is not set, MO is set to "ignore" and CDA is set to "value-sent"

- TV contains a stable value, MO is MSB(X) and CDA is set to LSB

6.4. Payload Length field

If the LPWAN technology does not add padding, this field can be elided for the transmission on the LPWAN network. The SCHC C/D recomputes the original payload length value. The TV is not set, the MO is set to "ignore" and the 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)" and the CDA to "LSB". The 's' parameter depends on the expected maximum packet length.

On other cases, the payload length field must be sent and the CDA is replaced by "value-sent".

6.5. Next Header field

If the Next Header field identified by the rest of the rule does not vary and is known by both sides, the TV should contain this Next Header value, the MO should be "equal" and the CDA should be "not-sent".

If the Next header field identified by the rest of the rule varies during time or is not known by both sides, then TV is not set, MO is set to "ignore" and CDA is set to "value-sent". A matching-list may also be used.

6.6. Hop Limit field

The End System is generally a device and does not forward packets, therefore the Hop Limit value is constant. So the TV is set with a default value, the MO is set to "equal" and the CDA is set to "not-sent".

Otherwise the value is sent on the LPWAN: TV is not set, MO is set to ignore and CDA is set to "value-sent".

Note that the field behavior differs in upstream and downstream. In upstream, since there is no IP forwarding between the Dev and the SCHC C/D, the value is relatively constant. On the other hand, the downstream value depends of Internet routing and may change more frequently. One solution could be to use the Direction Indicator (DI) to distinguish both directions to elide the field in the upstream direction and send the value in the downstream direction.

6.7. 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 a single rule, these values are identified by their role (DEV or APP) and not by their position in the frame (source or destination). The SCHC C/D must be aware of the traffic direction (upstream, downstream) to select the appropriate field.

6.7.1. IPv6 source and destination prefixes

Both ends must be synchronized with the appropriate prefixes. For a specific flow, the source and destination prefix 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 contains the values, the MO is set to "equal" and the CDA is set to "not-sent".

In case the rule allows several prefixes, mapping-list must be used. The different prefixes are listed in the TV associated with a short ID. The MO is set to "match-mapping" and the CDA is set to "mapping-sent".

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

6.7.2. 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 is generally carrying a single device identifier corresponding to the DEV. The SCHC C/D may also not be aware of these values.

If the DEV address has a static value that is not derived from an IEEE EUI-64, then TV contains the actual Dev address value, the MO operator is set to "equal" and the CDA is set to "not-sent".

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

Otherwise the value variation of the IID may be reduced to 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 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".

6.8. IPv6 extensions

No extension rules are currently defined. They can be based on the MOs and CDAs described above.

6.9. UDP source and destination port

To allow a single rule, the UDP port values are identified by their role (DEV or APP) and not by their position in the frame (source or destination). The SCHC C/D must be aware of the traffic direction (upstream, downstream) 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 this values, the MO is set to "match-mapping" and the CDA is set to "mapping-sent".

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

6.10. UDP length field

If the LPWAN technology does not introduce padding, 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-UDP-length".

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

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

6.11. UDP Checksum field

IPv6 mandates a checksum in the protocol above IP. Nevertheless, if a more efficient mechanism such as L2 CRC or MIC is carried by or over the L2 (such as in the LPWAN fragmentation process (see section Section 5)), the UDP checksum transmission can be avoided. In that case, the TV is not set, the MO is set to "ignore" and the CDA is set to "compute-UDP-checksum".

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

7. Security considerations

7.1. Security considerations for header compression

A malicious header compression could cause the reconstruction of a wrong packet that does not match with the original one, such corruption may be detected with end-to-end authentication and integrity mechanisms. Denial of Service may be produced but its arise other security problems that may be solved with or without header compression.

7.2. Security considerations for fragmentation

This subsection describes potential attacks to LPWAN fragmentation and suggests possible countermeasures.

A node can perform a buffer reservation attack by sending a first fragment to a target. Then, the receiver will reserve buffer space for the IPv6 packet. Other incoming fragmented 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 fragments of multiple packets can be stored in the reassembly buffer. To further increase the attack cost, the reassembly buffer can be split into 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 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 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 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 fragments to be transmitted by a node, by applying content-chaining to the different fragments, based on cryptographic hash functionality. The aim of this technique is to allow a receiver to identify illegitimate fragments.

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

8. Acknowledgements

Thanks to Dominique Barthel, Carsten Bormann, Philippe Clavier, Arunprabhu Kandasamy, Antony Markovski, Alexander Pelov, Pascal Thubert, Juan Carlos Zuniga and Diego Dujovne for useful design consideration and comments.

9. References

9.1. Normative References

[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>.

[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>.

9.2. Informative References

[I-D.ietf-lpwan-overview]
Farrell, S., "LPWAN Overview", draft-ietf-lpwan-overview-06 (work in progress), July 2017.

Appendix A. SCHC Compression Examples

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

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

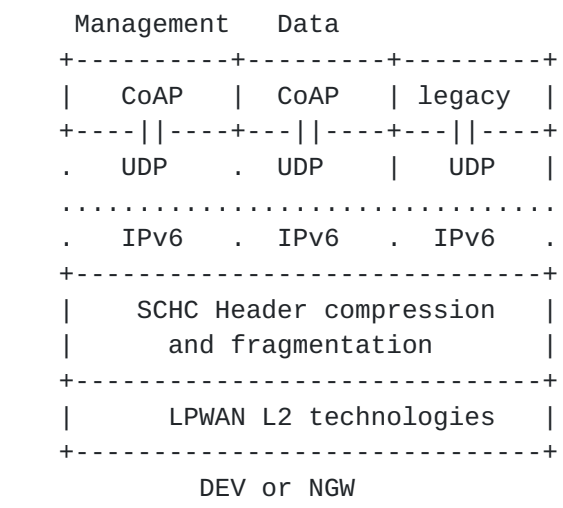


Figure 14: Simplified Protocol Stack for LP-WAN

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

Rule 0

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

Rule 1

Field	FP	DI	Value	Match Opera.	Action Action	Sent [bits]
IPv6 version	1	Bi	6	equal	not-sent	
IPv6 DiffServ	1	Bi	0	equal	not-sent	
IPv6 Flow Label	1	Bi	0	equal	not-sent	
IPv6 Length	1	Bi		ignore	comp-length	
IPv6 Next Header	1	Bi	17	equal	not-sent	
IPv6 Hop Limit	1	Bi	255	ignore	not-sent	
IPv6 DEVprefix	1	Bi	[alpha/64, fe80::/64]	match-mapping	mapping-sent	[1]
IPv6 DEViid	1	Bi		ignore	DEViid	
IPv6 APPprefix	1	Bi	[beta/64, alpha/64, fe80::64]	match-mapping	mapping-sent	[2]
IPv6 APPiid	1	Bi	::1000	equal	not-sent	
UDP DEVport	1	Bi	5683	equal	not-sent	
UDP APPport	1	Bi	5683	equal	not-sent	
UDP Length	1	Bi		ignore	comp-length	
UDP checksum	1	Bi		ignore	comp-chk	

Rule 2

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

Figure 15: Context rules

All the fields described in the three rules depicted on Figure 15 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 of different fragment delivery reliability options possible on the basis of this specification.

Figure 16 illustrates the transmission of an IPv6 packet that needs 11 fragments in the No ACK option.

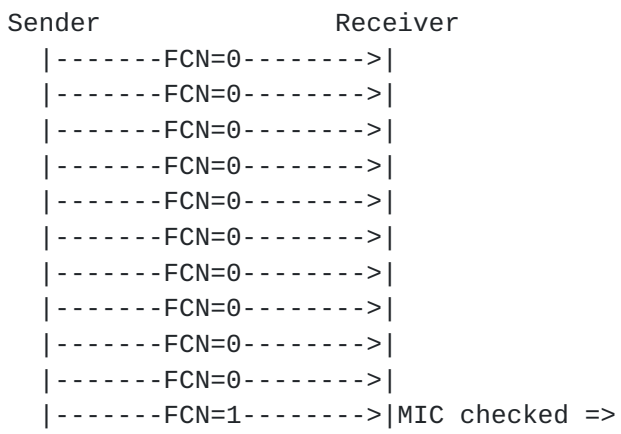


Figure 16: Transmission of an IPv6 packet carried by 11 fragments in the No ACK option

Figure 17 illustrates the transmission of an IPv6 packet that needs 11 fragments in Window mode - ACK on error, for N=3, without losses.

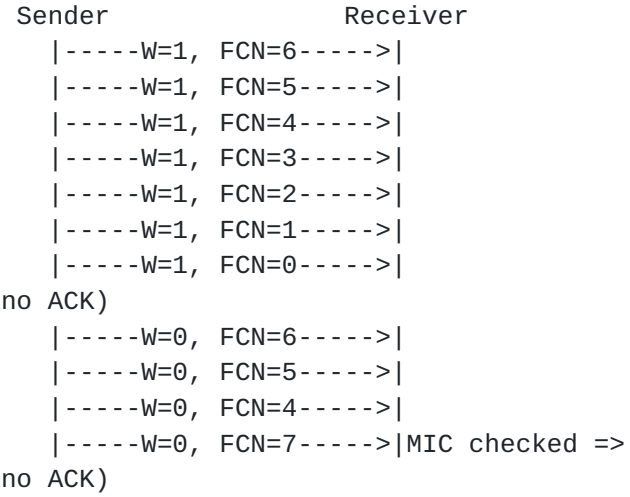


Figure 17: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK on error, for N=3 and MAX_WIND_FCN=6, without losses.

Figure 18 illustrates the transmission of an IPv6 packet that needs 11 fragments in Window mode - ACK on error, for N=3, with three losses.

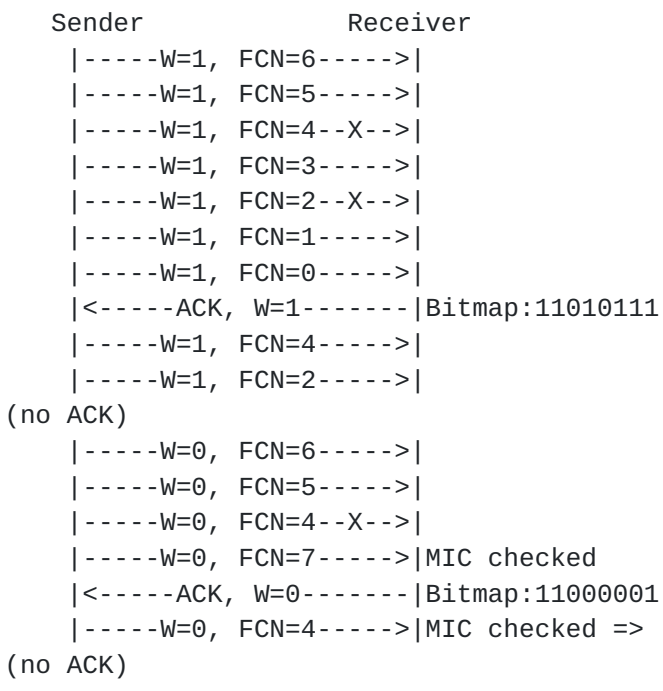


Figure 18: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK on error, for N=3 and MAX_WIND_FCN=6, three losses.

Figure 19 illustrates the transmission of an IPv6 packet that needs 11 fragments in Window mode - ACK "always", for N=3 and MAX_WIND_FCN=6, without losses. Note: in Window mode, an additional bit will be needed to number windows.

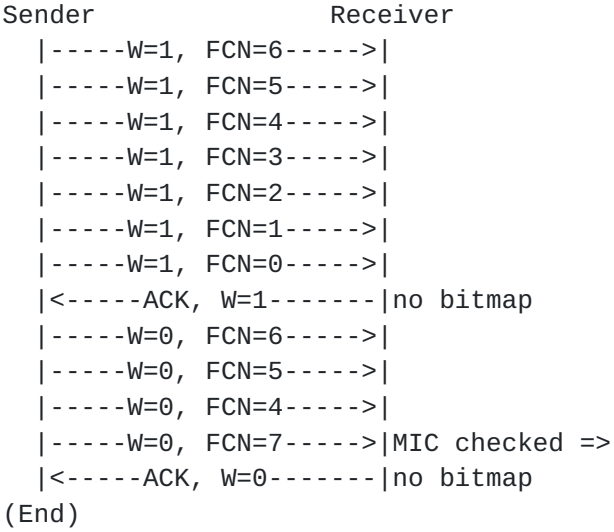


Figure 19: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK "always", for N=3 and MAX_WIND_FCN=6, no losses.

Figure 20 illustrates the transmission of an IPv6 packet that needs 11 fragments in Window mode - ACK "always", for N=3 and MAX_WIND_FCN=6, with three losses.

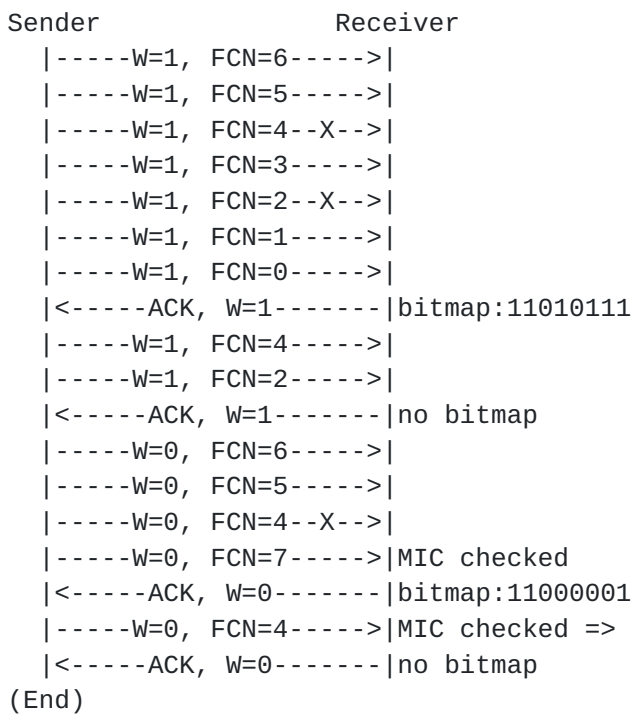
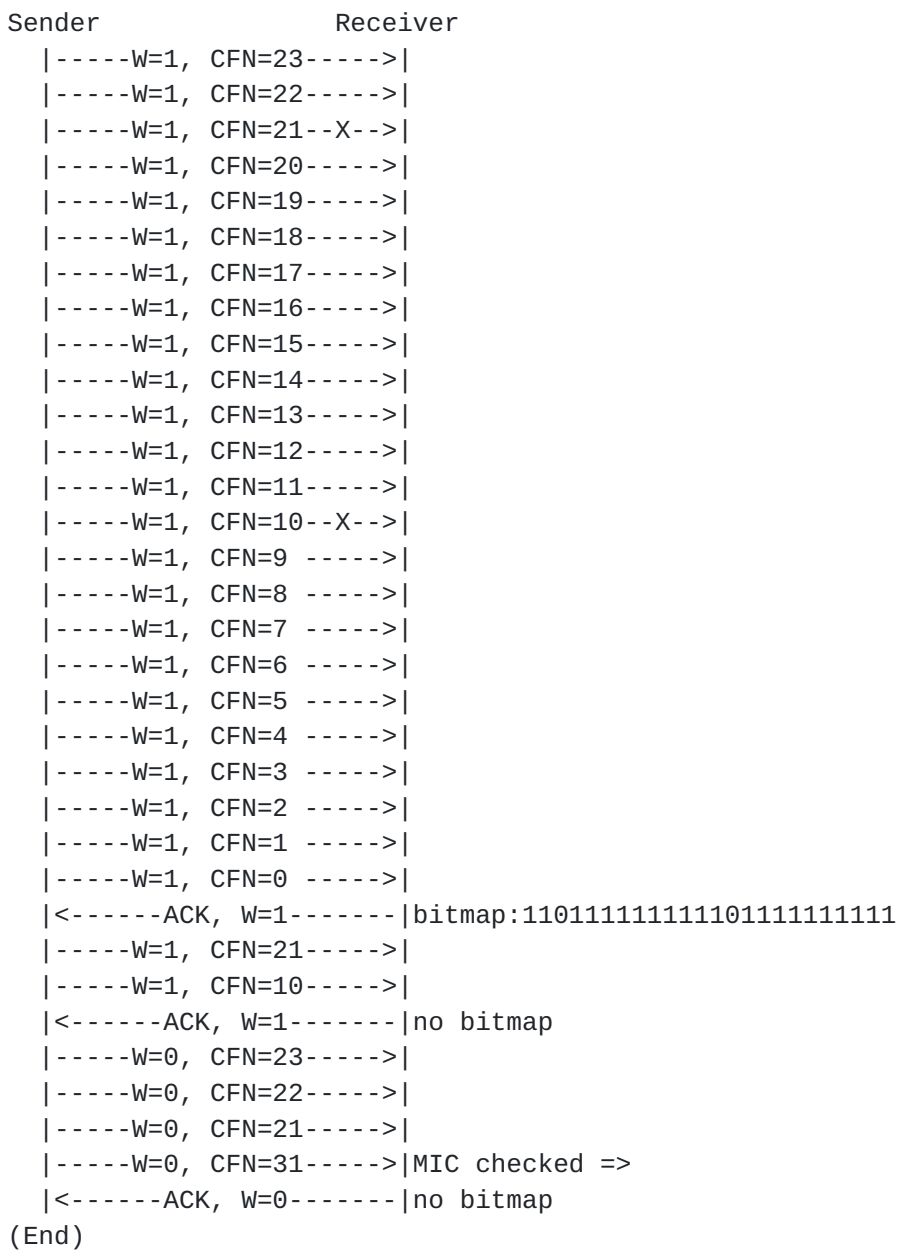


Figure 20: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK "Always", for N=3, and MAX_WIND_FCN=6, with three losses.

Appendix C illustrates the transmission of an IPv6 packet that needs 28 fragments in Window mode - ACK "always", for N=5 and MAX_WIND_FCN=23, with two losses. Note that MAX_WIND_FCN=23 may be useful when the maximum possible bitmap size, considering the maximum lower layer technology payload size and the value of R, is 3 bytes. Note also that the FCN of the last fragment of the packet is the one with FCN=31 (i.e. FCN=2^N-1 for N=5, or equivalently, all FCN bits set to 1).



Appendix C. Allocation of Rule IDs for fragmentation

A set of Rule IDs are allocated to support different aspects of fragmentation functionality as per this document. The allocation of IDs is to be defined in other documents. The set MAY include:

- one ID or a subset of IDs to identify a fragment as well as its reliability option and its window size, if multiple of these are supported.

- one ID to identify the ACK message.

- one ID to identify the Abort message as per Section 9.8.

Appendix D. 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
2bis rue de la Chataigneraie
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