lpwan Working Group
Internet-Draft
Intended status: Informational
Expires: November 6, 2017

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

May 05, 2017

LPWAN Static Context Header Compression (SCHC) and fragmentation for IPv6 and UDP
draft-ietf-lpwan-ipv6-static-context-hc-03

Abstract

This document describes a header compression scheme and fragmentation functionality for IPv6/UDP protocols. These techniques are especially tailored for LPWAN (Low Power Wide Area Network) networks and could be extended to other protocol stacks.

The Static Context Header Compression (SCHC) offers a great level of flexibility when processing the header fields. Static context means that information stored in the context which, describes field values, does not change during the packet transmission, avoiding complex resynchronization mechanisms, incompatible with LPWAN characteristics. In most of the cases, IPv6/UDP headers are reduced to a small identifier.

This document describes the generic compression/decompression process and applies it to IPv6/UDP headers. Similar mechanisms for other protocols such as CoAP will be described in a separate document. Moreover, this document specifies fragmentation and reassembly mechanims for SCHC compressed packets exceeding the L2 pdu size and for the case where the SCHC compression is not possible then the IPv6/UDP packet is sent.

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 November 6, 2017.

Table of Contents

1. Introduction

Header compression is mandatory to efficiently bring Internet connectivity to the node within a LPWAN network [I-D.minaburo-lp-wan-gap-analysis].

Some LPWAN networks properties can be exploited for 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 order. This context is static (the values on the header fields do not change during 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 is indedependent of the specific LPWAN technology over which it will be used.

On the other hand, LPWAN technologies are 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 these to support the IPv6 MTU requirement of 1280 bytes [RFC2460] is the use of a fragmentation mechanism at the adaptation layer below IPv6. This specification defines 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. Vocabulary

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

- 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.

- APP: LPWAN Application. An application sending/consuming IPv6 packets to/from the Device.

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

- MO: Matching Operator. An operator used to compare 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. Rule ID is sent on the LPWAN.

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

3. 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. Based on the fact that the nature of data flows is highly predictable in LPWAN networks, a static context may be stored on the Device (DEV). The context must be stored in both ends. It can also be learned by using a provisionning protocol that is out of the scope of this draft.

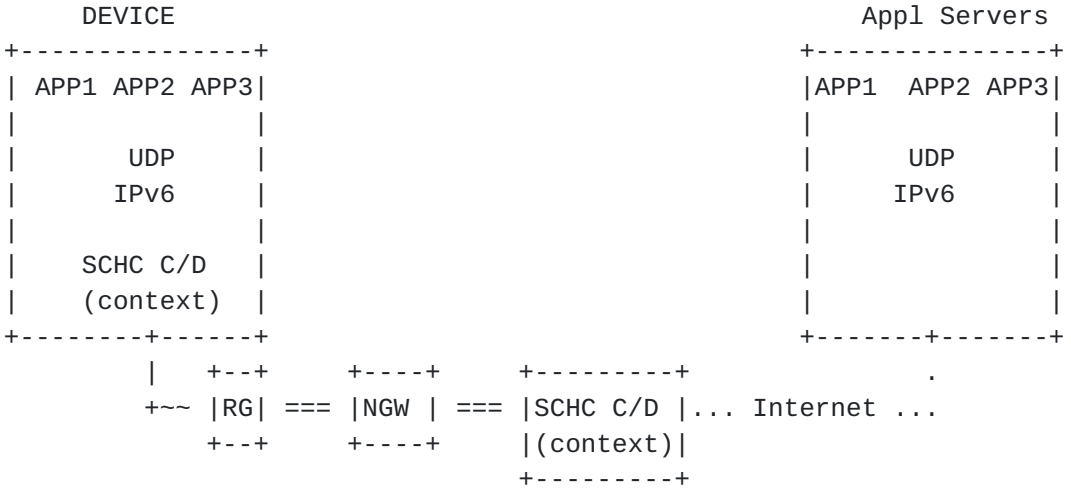

Figure 1: Architecture

Figure 1 based on [I-D.ietf-lpwan-overview] terminology represents the architecture for compression/decompression. The Device is running applications which produce IPv6 or IPv6/UDP flows. These flows are compressed by an Static Context Header Compression Compressor/Decompressor (SCHC C/D) to reduce the headers size. Resulting information is sent on a layer two (L2) frame to the LPWAN Radio Network to a Radio Gateway (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 places if a tunnel is established between the NGW and the SCHC C/D. This architecture forms a star topology. After decompression, the packet can be sent on the Internet to one or several LPWAN Application Servers (APP).

The principle is exactly the same in the other direction.

The context contains a list of rules (cf. Figure 2). 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).

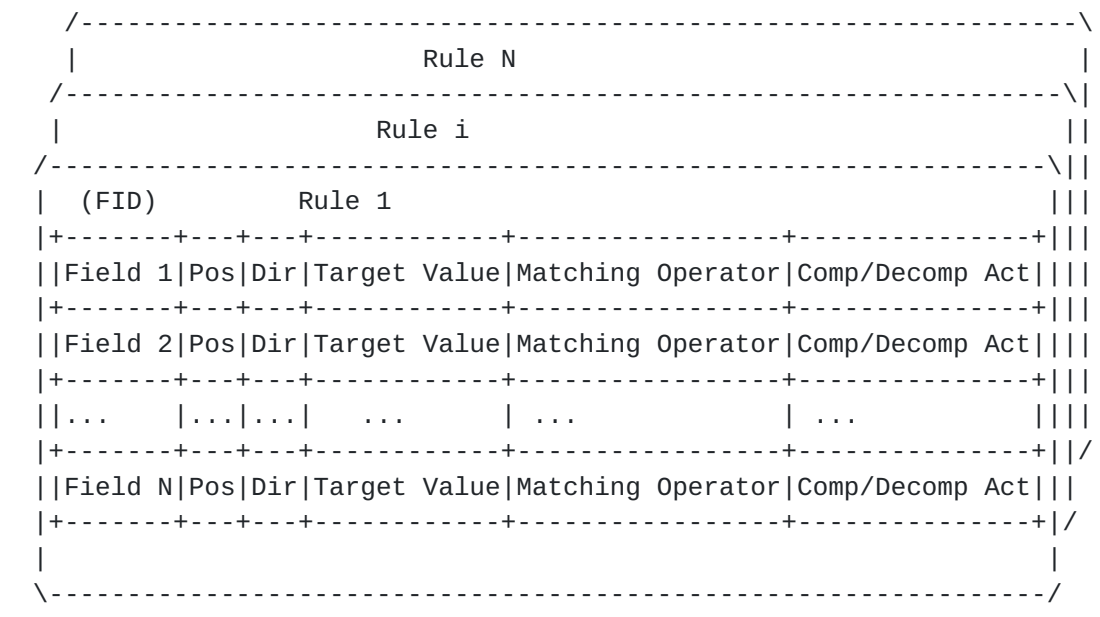

Figure 2: Compression Decompression Context

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 done in the same order they appear in the packet.

On the other hand, the rule describes the compressed header 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 main idea of the compression scheme is to send the rule id to the other end instead of known field values. When a value is known by both ends, it is not necessary to send it on the LPWAN network.

The field description is composed of different entries:

- A Field ID (FID) is a unique value to define the field.
- A Field Position (FP) indicating if several instances of the field exist in the headers which one is targeted.
- A direction indicator (DI) indicating the packet direction. Three values are possible:

  * upstream when the field or the value is only present in packets sent by the DEV to the APP,

  * downstream when the field or the value is only present in packet sent from the APP to the DEV and

  * bi-directional 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,...). It can be a single value or a more complex structure (array, list,...). It can be considered as 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, which can be considered as a CBOR structure. 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, which can be considered as a CBOR structure.

3.1. Rule ID

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

Some values in the rule ID space may be reserved for goals other than header compression, for example fragmentation.

Rule IDs are specific to a DEV. Two DEVs may use the same rule ID for different header compression. The SCHC C/D needs to combine the rule ID with the DEV L2 address to find the appropriate rule.

3.2. 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 headers. Each field is associated to a matching operator for compression. Each header field's value is compared to the corresponding target value stored in the rule for that field using the matching operator. This comparison includes the direction indicator and the field position in the header. If

all the fields in the packet's header satisfy all the matching operators (excluding unappropriate direction or position) of a rule, the packet is processed using Compression Decompression Function associated with the fields. Otherwise the next rule is tested. If no eligible rule is found, then the packet is sent without compression, which may require using the fragmentation procedure.

In the downstrean direction, the rule is also used to find the device ID.

- sending: The rule ID is sent to the other end followed by information resulting from the compression of header fields. This information is sent in the order expressed in the rule for the matching fields. The way the rule ID is sent depends on the layer two technology and will be specified in a specific document. For example, it can either be included in a Layer 2 header or sent in the first byte of the L2 payload. (cf. Figure 3)

- decompression: 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 header fields. It applies the CDA action to reconstruct the original header fields. The CDA order can be different of the order given by the rule. For instance Compute-* may be applied after the other CDAs.

Figure 3: LPWAN Compressed Format Packet

4. Matching operators

This document describes basic matching operators (MO)s which must be known 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 type. The MOs and their definitions are provided next:

- equal: a field value in a packet matches with a field value in a rule if they are equal.

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

- MSB(length): a field value of a size equal to "length" bits in a packet matches with a field value in a rule if the most significant "length" bits are equal.

- 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 pairs. Each pair is composed of a value and a short ID (or index). This operator matches if a field value is equal to one of the pairs' values.

Matching Operators and match-mapping needs a parameter to proceed to the matching. Match-mapping requires a list of values associated to an index and MSB requires an integer indicating the number of bits to test.

5. Compression Decompression Actions (CDA)

The Compression Decompression Actions (CDA) describes the action 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	ctxt value OR rcvd value
compute-length	elided	compute length
compute-checksum	elided	compute UDP checksum
DEViid-DID	elided	build IID from L2 DEV addr
APPiid-DID	elided	build IID from L2 APP addr

Figure 4: Compression and Decompression Functions

Figure 4 sumarizes the 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.

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 that field on which compression is applied.

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

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.

The compressor sends the Target Value stored on the rule in the compressed header message. The decompressor restores the field value with the one received from the LPWAN

5.3. mapping-sent

mapping-sent is used to send a smaller index associated to the field value 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 bit sent is the minimal number to code all the indexes.

5.4. LSB CDA

LSB action is used to send a fixed part of the packet field header to the other end. This action is used together with the "MSB" MO. A length can be specified to indicate how many bits have to be sent.

If not length is specified, the number of bit sent are the field length minus the bit length specified in the MSB MO.

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

5.5. DEViid-DID, APPiid-DID CDA

These functions are used to process respectively the Device and the Application Device Identifier (DID). APPiid-DID CDA is less common, since current LPWAN technologies frames contain a single address.

The IID value is computed from the Device ID present in the Layer 2 header. The computation depends on the technology and the Device ID size.

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

5.6. Compute-*

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

6. Application to 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 wellknown 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 there is without compression and the original value is sent, or the sencond where the values can be computed by sending only the LSB bits:

- 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(8-X)

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 dpending 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(20-X)

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 recompute 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 is small, the TV can be set to 0x0000, the MO set to "MSB (16-s)" and the CDA to "LSB (s)". The 's' parameter depends on the 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 host 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-DID" or "APPiid-DID". 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.

For privacy reasons or if the DEV address is changing over time, it maybe better to use a static value. In that case, the TV contains the 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 IID, 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 CDF 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 knows 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 XXXX)), 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. Examples

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

7.1. IPv6/UDP compression

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

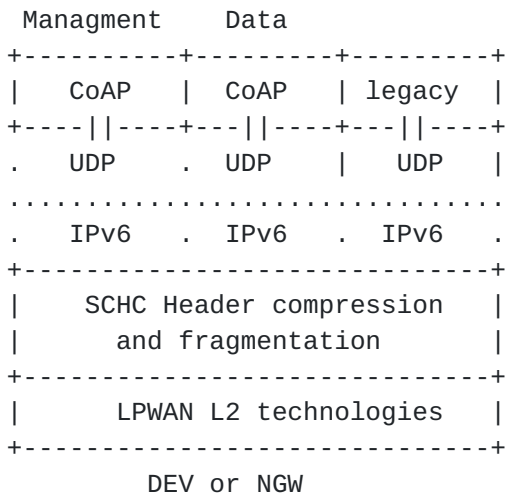

Figure 5: Simplified Protocol Stack for LP-WAN

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

Rule 0

Field	Value	Match	Function	Sent
IPv6 version	6	equal	not-sent	

```
|IPv6 DiffServ   |0        | equal  | not-sent    ||      |
|IPv6 Flow Label |0        | equal  | not-sent    ||      |
|IPv6 Length     |         | ignore | comp-length ||      |
|IPv6 Next Header|17       | equal  | not-sent    ||      |
|IPv6 Hop Limit  |255      | ignore | not-sent    ||      |
|IPv6 DEVprefix  |FE80::/64| equal  | not-sent    ||      |
|IPv6 DEViid     |         | ignore | DEViid-DID  ||      |
|IPv6 APPprefix  |FE80::/64| equal  | not-sent    ||      |
|IPv6 APPiid     |::1      | equal  | not-sent    ||      |
+================+=========+========+=============++======+
|UDP DEVport     |123      | equal  | not-sent    ||      |
|UDP APPport     |124      | equal  | not-sent    ||      |
|UDP Length      |         | ignore | comp-length ||      |
|UDP checksum    |         | ignore | comp-chk    ||      |
+================+=========+========+=============++======+
```

Rule 1

```
+----------------+---------+--------+-------------++------+
| Field          | Value   | Match  | Function    || Sent |
+----------------+---------+--------+-------------++------+
|IPv6 version    |6        | equal  | not-sent    ||      |
|IPv6 DiffServ   |0        | equal  | not-sent    ||      |
|IPv6 Flow Label |0        | equal  | not-sent    ||      |
|IPv6 Length     |         | ignore | comp-length ||      |
|IPv6 Next Header|17       | equal  | not-sent    ||      |
|IPv6 Hop Limit  |255      | ignore | not-sent    ||      |
|IPv6 DEVprefix  |alpha/64 | equal  | not-sent    ||      |
|IPv6 DEViid     |         | ignore | DEViid-DID  ||      |
|IPv6 APPprefix  |beta/64  | equal  | not-sent    ||      |
|IPv6 APPiid     |::1000   | equal  | not-sent    ||      |
+================+=========+========+=============++======+
|UDP DEVport     |5683     | equal  | not-sent    ||      |
|UDP APPport     |5683     | equal  | not-sent    ||      |
|UDP Length      |         | ignore | comp-length ||      |
|UDP checksum    |         | ignore | comp-chk    ||      |
+================+=========+========+=============++======+
```

Rule 2

```
+----------------+---------+--------+-------------++------+
| Field          | Value   | Match  | Function    || Sent |
+----------------+---------+--------+-------------++------+
|IPv6 version    |6        | equal  | not-sent    ||      |
|IPv6 DiffServ   |0        | equal  | not-sent    ||      |
|IPv6 Flow Label |0        | equal  | not-sent    ||      |
|IPv6 Length     |         | ignore | comp-length ||      |
|IPv6 Next Header|17       | equal  | not-sent    ||      |
|IPv6 Hop Limit  |255      | ignore | not-sent    ||      |
|IPv6 DEVprefix  |alpha/64 | equal  | not-sent    ||      |
```

IPv6 DEViid		ignore	DEViid-DID		
IPv6 APPprefix	gamma/64	equal	not-sent		
IPv6 APPiid	::1000	equal	not-sent		
UDP DEVport	8720	MSB(12)	LSB(4)		lsb
UDP APPport	8720	MSB(12)	LSB(4)		lsb
UDP Length		ignore	comp-length		
UDP checksum		ignore	comp-chk		

Figure 6: Context rules

All the fields described in the three rules Figure 6 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.

8. Fragmentation

8.1. Overview

Fragmentation support in LPWAN is mandatory when the underlying LPWAN technology is not capable of fulfilling the IPv6 MTU requirement. Fragmentation is used if, after SCHC header compression, the size of the resulting IPv6 packet is larger than the L2 data unit maximum payload. Fragmentation is also used if SCHC header compression has not been able to compress an IPv6 packet that 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 IPv6 datagram fits within a single L2 data unit, the fragmentation mechanism is not used and the packet is sent unfragmented.
If the datagram does not fit within 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 IPv6 datagram. On the other hand, in order to preserve energy, Things (End Systems) 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 for a gradation of fragment

delivery reliability. This document does not make any decision with regard to which fragment delivery reliability option is 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.

8.2. Reliability options: definition

This specification defines the following five fragment delivery reliability options:

o No ACK

o Packet mode - ACK "always"

o Packet mode - ACK on error

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 the underlying LPWAN technology to decide which reliability option to use and whether the same reliability option applies to all IPv6 packets. Note that the reliability option to be used is not necessarily tied to the particular characteristics of the underlying L2 LPWAN technology (e.g. a reliability option without receiver feedback 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 Packet mode - ACK "always", the receiver transmits one ACK after all fragments carrying an IPv6 packet have been transmitted. The ACK informs the sender about received and/or missing fragments from the IPv6 packet.

In Packet mode - ACK on error, the receiver transmits one ACK after all fragments carrying an IPv6 packet have been transmitted, only if at least one of those fragments has been lost. The ACK informs the sender about received and/or missing fragments from the IPv6 packet.

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 missing fragments from the window of fragments.

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 missing fragments from the window of fragments.

In Packet or Window mode, upon receipt of an ACK that informs about any lost fragments, the sender retransmits the lost fragments, up to a maximum number of ACK and retransmission rounds that is TBD.

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

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

8.3. Reliability options: discussion

This section discusses the properties of each fragment delivery reliability option defined in the previous section. Figure Figure 7 summarizes advantages and disadvantages of the reliability options that provide receiver feedback.

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.

ACK on error options are based on the optimistic expectation that the underlying links will offer relatively low L2 data unit loss probability. ACK on error reduces the number of ACKs transmitted by the fragment receiver compared to ACK "always" options. This may be especially 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.

The Packet mode - ACK on error option provides reliability with low ACK overhead. However, if an ACK is lost, the sender assumes that all fragments carrying the IPv6 datagram have been successfully delivered. In contrast, the Packet mode - ACK "always" option does not suffer that issue, at the expense of a moderate ACK overhead. An

issue with any of the Packet modes is that detection of a long burst of lost frames is only possible after relatively long time (i.e. at the end of the transmission of all fragments carrying an IPv6 datagram).

In contrast with Packet modes, the Window mode - ACK "always" option provides flow control. In addition, it is able to better handle long bursts of lost fragments, since detection of such events can be done earlier than with any of the Packet modes. However, the benefits of Window mode - ACK "always" come at the expense of higher ACK overhead.

With regard to the Window mode - ACK on error option, there is no known use case for it at the time of the writing.

	Packet mode	Window mode
ACK on error	+ Low ACK overhead - Long loss burst - No flow control	(Use case unknown)
ACK "always"	+ Moderate ACK overh. - Long loss burst - No flow control	+ Flow control + Long loss burst - Higher ACK overhead

Figure 7: Summary of fragment delivery options that provide receiver feedback, and their main advantages (+) and disadvantages (-).

8.4. Fragment format

A fragment comprises a fragmentation header and a fragment payload, and conforms to the format shown in Figure 8. The fragment payload carries a subset of either the 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 8: Fragment format.

8.5. Fragmentation header formats

In any of the Window modes, fragments except the last one SHALL contain the fragmentation header as defined in Figure 9. The total size of this fragmentation header is R bits.

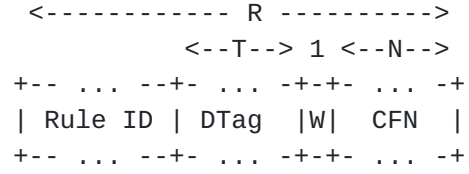

Figure 9: Fragmentation Header for Fragments except the Last One, Window mode

In any of the Packet modes, fragments (except the last one) that are transmitted for the first time SHALL contain the fragmentation header shown in Figure 10. The total size of this fragmentation header is R bits.

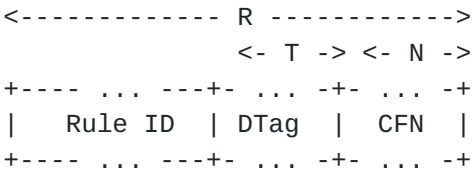

Figure 10: Fragmentation Header for Fragments except the Last One, in a Packet mode; first transmission attempt

In any of the Packet modes, fragments (except the last one) that are retransmitted SHALL
contain the fragmentation header as defined in Figure 11.

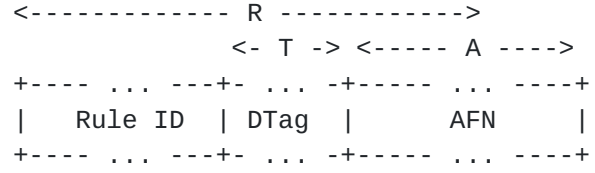

Figure 11: Fragmentation Header for Retransmitted Fragments (Except the Last One) in a Packet mode

The last fragment of an IPv6 datagram, regardless of whether a Packet mode or Window mode is in use, SHALL contain a fragmentation header that conforms to the format shown in Figure 12. The total size of this fragmentation header is R+M bits.

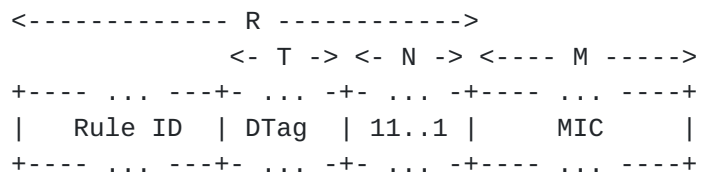

Figure 12: Fragmentation Header for the Last Fragment

- o Rule ID: this field has a size of R - T - N - 1 bits in all fragments that are not the last one, when Window mode is used. In all other fragments, the Rule ID field has a size of R - T - N bits. 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 be used to signal which reliability option is in use. In any of the Packet modes, Rule ID is also used to indicate whether the fragment is a first transmission or a retransmission.

- o DTag: DTag stands for Datagram Tag. 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. The DTag field allows to interleave fragments that correspond to different IPv6 datagrams. DTag MUST be set sequentially increasing from 0 to 2^T - 1, and MUST wrap back from 2^T - 1 to 0.

- o CFN: CFN stands for Compressed Fragment Number. The size of the CFN field is N bits. In the No ACK option, N=1. For the rest of options,
  N equal to or greater than 3 is recommended. This field is an unsigned integer that carries a non-absolute fragment number. The CFN MUST be set sequentially decreasing from 2^N - 2 for the first fragment, and MUST wrap from 0 back to 2^N - 2 (e.g. for N=3, the first fragment has CFN=6, subsequent CFNs are set sequentially and in decreasing order, and CFN will wrap from 0 back to 6). The CFN for the last fragment has all bits set to 1. Note that, by this definition, the CFN 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 CFN does not strictly 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 CFN equal to 1, while all previous fragments will carry a CFN of 0.

- o W: W is a 1-bit flag that is used in Window mode. Its purpose is avoiding possible ambiguity for the receiver that might arise under certain conditions. This flag carries the same value for

all fragments of a window, and it is set to the other value for the next window. The initial value for this flag is 1.

- AFN: AFN stands for Absolute Fragment Number. This field has a size of A bits. 'A' may be greater than N. The AFN is an unsigned integer that carries the absolute fragment number that corresponds to a fragment from an IPv6 packet. The AFN MUST be set sequentially and in increasing order, starting from 0.

- MIC: MIC stands for Message Integrity Check. This field has a size of M bits. It is computed by the sender over the complete IPv6 packet before fragmentation by using the TBD algorithm. The MIC allows to check for errors in the reassembled IPv6 packet, while it also enables compressing the UDP checksum by use of SCHC.

  The values for R, N, A and M are not specified in this document, and have to be determined by the underlying LPWAN technology.

8.6. ACK format

The format of an ACK is shown in Figure 13:

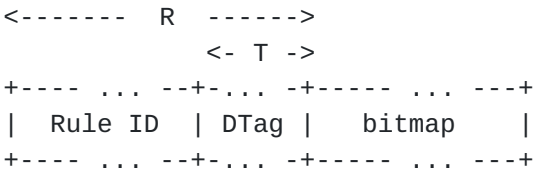

Figure 13: Format of an ACK

Rule ID: In all ACKs, Rule ID has a size of R bits and SHALL be set to TBD_ACK to signal that the message is an ACK.

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.

bitmap: 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 (in Window mode) or the number of fragments that carry the IPv6 packet (in Packet mode). The bitmap is a sequence of bits, where the n-th bit signals whether the n-th fragment transmitted 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 (carrying the MIC) has been correctly received, and 0 otherwise. Absence of the bitmap in an ACK

confirms correct reception of all fragments to be acknowledged by means of the ACK.

Figure 14 shows an example of an ACK in Packet mode, where the bitmap indicates that the second and the ninth fragments have not been correctly received. In this example, the IPv6 packet is carried by eleven fragments in total, therefore the bitmap has a size of two bytes.

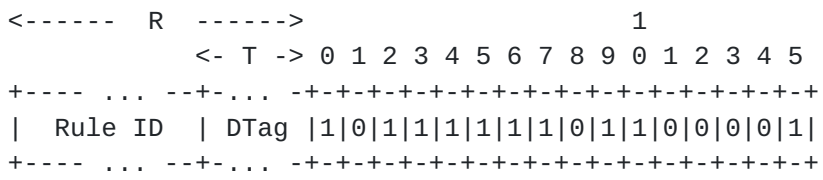

Figure 14: Example of the Bitmap in an ACK

Figure 15 shows an example of an ACK in Window mode (N=3), where the bitmap indicates that the second and the fifth fragments have not been correctly received.

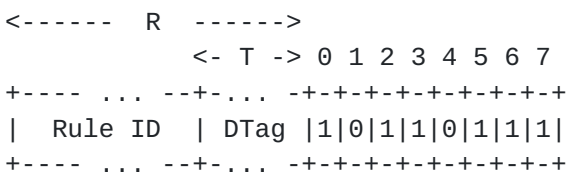

Figure 15: Example of the bitmap in an ACK (in Window mode, for N=3)

Figure 16 illustrates an ACK without bitmap.

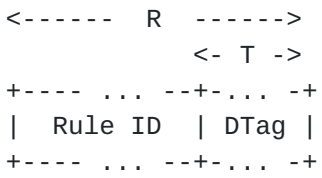

Figure 16: Example of an ACK without bitmap

8.7. 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 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 CFN 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 CFN and order of arrival of the fragments, and the fragment payload sizes. In Window mode, the fragment receiver also uses the W flag in the received fragments. Note that the size of the original, unfragmented IPv6 packet cannot be determined from fragmentation headers.

When ACK on error is used (for either Packet mode or Window mode), 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 every time that a new fragment carrying data from the same IPv6 datagram is received. In Packet mode - ACK on error, upon timer expiration, if the last fragment of the IPv6 datagram (i.e. carrying all CFN bits set to 1) has not been received, an ACK MUST be transmitted by the fragment receiver to indicate received and not received fragments for that IPv6 datagram.
In Window mode - ACK on error, upon timer expiration, if neither the last fragment of the IPv6 datagram nor the last fragment of the current window (with CFN=0) have been received, an ACK MUST be transmitted by the fragment receiver to indicate received and not received fragments for the current window.

Note that, in Window mode, the first fragment of the window is the one sent with CFN=2^N-2. Also note that, in Window mode, the fragment with CFN=0 is considered the last fragment of its window, except for the last fragment of the whole packet (with all CFN bits set to 1), which is also the last fragment of the last window. Upon receipt of the last fragment of a window, if Window mode - ACK "Always" is used, the fragment receiver MUST send an ACK to the fragment sender. The ACK provides feedback on the fragments received and lost that correspond to the last window.

If the recipient receives the last fragment of an IPv6 datagram, it checks for the integrity of the reassembled IPv6 datagram, based on the MIC received. In No ACK mode, 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. If ACK "Always" is used, the recipient MUST transmit an ACK to the fragment sender. The ACK

provides feedback on the whole set of fragments sent that carry the complete IPv6 packet (Packet mode) or on the fragments that correspond to the last window (Window mode). If ACK on error is used, the recipient MUST NOT transmit an ACK to the sender if no losses have been detected for the whole IPv6 packet (Packet mode) or in the last window (Window mode). If losses have been detected, the recipient MUST then transmit an ACK to the sender to provide feedback on the whole IPv6 packet (Packet mode) or in the last window (Window mode).

When ACK "Always" is used (in either Packet mode or Window mode), the fragment sender starts a timer (denoted "ACK Always Timer") after transmitting the last fragment of a fragmented IPv6 datagram. The initial value for this timer is not provided by this specification, and is expected to be defined in additional documents. Upon expiration of the timer, if no ACK has been received for this IPv6 datagram, the sender retransmits the last fragment, and it reinitializes and restarts the timer. In Window mode - ACK "Always", the fragment sender also starts the ACK Always Timer after transmitting the last fragment of a window. Upon expiration of the timer, if no ACK has been received for this window, the sender retransmits the last fragment, and it reinitializes and restarts the timer. Note that retransmitting the last fragment of a packet or a window as described serves as an ACK request. The maximum number of ACK requests in Packet mode or in Window mode is TBD.

In all reliability options, except for the No ACK option, the fragment sender retransmits any lost fragments reported in an ACK. In Packet modes, in order to minimize the probability of ambiguity with the CFN of different retransmitted fragments, the fragment sender
renumbers the CFNs of the fragments to be retransmitted by following the same approach as for a sequence of new fragments: the CFN for retransmitted fragments is set sequentially decreasing from 2^N - 2 for the first fragment, and MUST wrap from 0 back to 2^N - 2. However, the last fragment of the set of retransmitted fragments only carries a CFN with all bits set to 1 if it is actually a retransmission of the last fragment of the packet (i.e. the last fragment had been lost in the first place). Examples of fragment renumbering for retransmitted fragments in Packet modes can be found in Appendix A.

A maximum of TBD iterations of ACK and fragment retransmission rounds are allowed per-window or per-IPv6-packet in Window mode or in Packet mode, respectively.

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 a node 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.

8.8. Aborting a fragmented IPv6 datagram transmission

For several reasons, a fragment sender or a fragment receiver may want to abort the transmission of a fragmented IPv6 datagram.

If the fragment sender triggers abortion, it transmits to the receiver a format equivalent to a fragmentation header (with the format for a fragment that is not the last one), with the Rule ID field (of size R - T - N bits) set to TBD_ABORT_TX and all CFN bits set to 1. No data is carried along with this fragmentation header.

If the fragment receiver triggers abortion, it transmits to the fragment sender a Rule ID (of size R bits) set to TBD_ABORT_RX. The entity that triggers abortion (either a fragment sender or a fragment receiver) MUST release any resources allocated for the fragmented IPv6 datagram transmission being aborted.

When a fragment receiver receives an L2 frame containing a Rule ID set to TBD ABORT_TX and a CFN field with all bits set to 1, the receiver MUST release any resources allocated for the fragmented IPv6 datagram transmission being aborted.

When a fragment sender receives an L2 frame containing a Rule ID set to TBD_ABORT_RX, the fragment sender MUST abort transmission of the fragmented IPv6 datagram being transmitted, and MUST release any resources allocated for the fragmented IPv6 datagram transmission being aborted.

A further Rule ID value may be used by an entity to signal abortion of all on- going, possibly interleaved, fragmented IPv6 datagram transmissions.

8.9. 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.

9. Security considerations

9.1. Security considerations for header compression

TBD

9.2. Security considerations for fragmentation

This subsection describes potential attacks to LPWAN fragmentation and proposes countermeasures, based on existing analysis of attacks to 6LoWPAN fragmentation {HHWH}.

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 whole packet on the basis of the datagram size announced in that first fragment. 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. 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.

10. Acknowledgements

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

11. References

11.1. Normative References

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://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, <http://www.rfc-editor.org/info/rfc4944>.

11.2. Informative References

[I-D.ietf-lpwan-overview] Farrell, S., "LPWAN Overview", draft-ietf-lpwan-overview-01 (work in progress), February 2017.

[I-D.minaburo-lp-wan-gap-analysis] Minaburo, A., Pelov, A., and L. Toutain, "LP-WAN GAP Analysis", draft-minaburo-lp-wan-gap-analysis-01 (work in progress), February 2016.

Appendix A. Fragmentation examples

This section provides examples of different fragment delivery reliability options possible on the basis of this specification.

Figure 17 illustrates the transmission of an IPv6 packet that needs 11 fragments in the No ACK option.

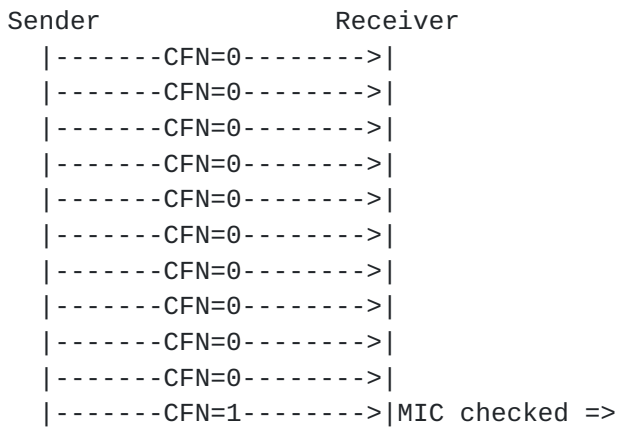

Figure 17: Transmission of an IPv6 packet carried by 11 fragments in the No ACK option

Figure 18 illustrates the transmission of an IPv6 packet that needs 11 fragments in Packet mode - ACK on error, for N=3, without losses.

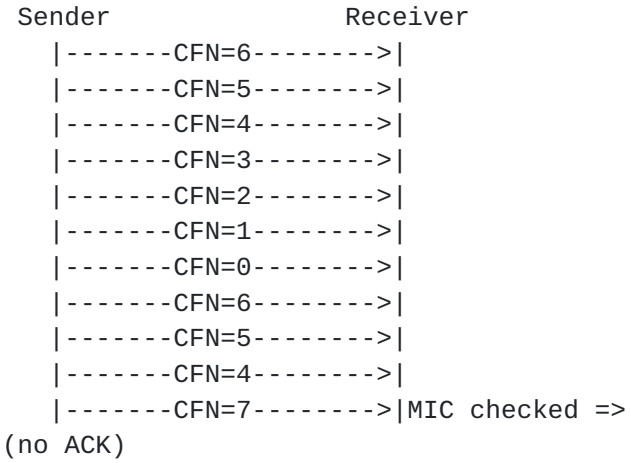

Figure 18: Transmission of an IPv6 packet carried by 11 fragments in Packet mode - ACK on error, for N=3, no losses.

Figure 19 illustrates the transmission of an IPv6 packet that needs 11 fragments in Packet mode - ACK on error, for N=3, with three losses.

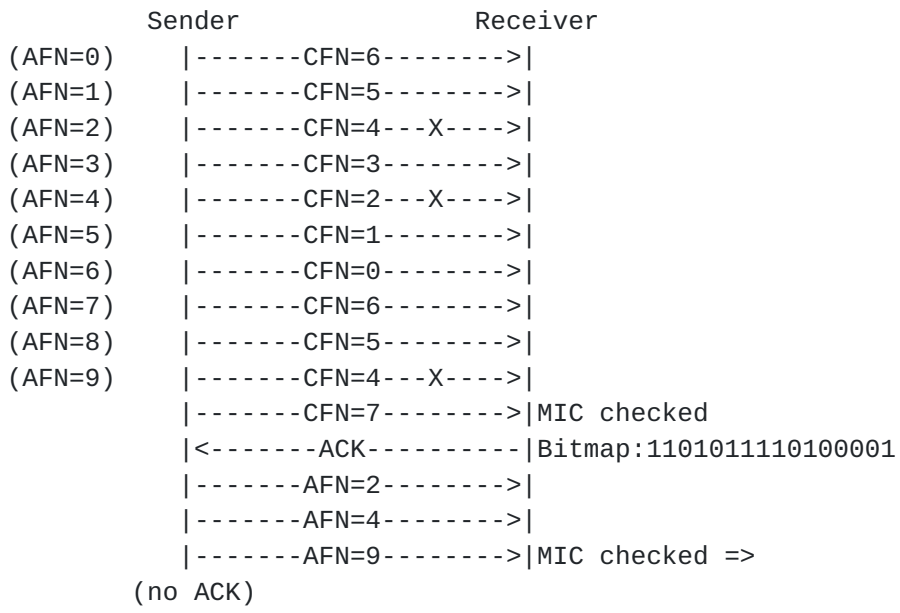

Figure 19: Transmission of an IPv6 packet carried by 11 fragments in Packet mode - ACK on error, for N=3, three losses. In the figure, (AFN=x) indicates the AFN value computed by the sender for each fragment.

Figure 20 illustrates the transmission of an IPv6 packet that needs 11 fragments in Window mode - ACK on error, for N=3, without losses.

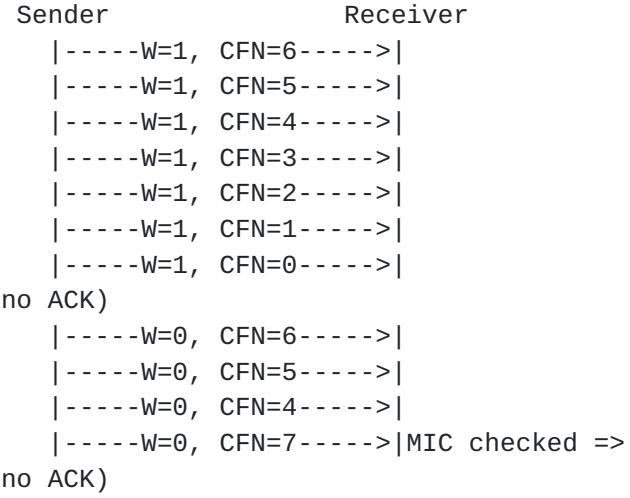

Figure 20: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK on error, for N=3, without losses.

Figure 21 illustrates the transmission of an IPv6 packet that needs 11 fragments in Window mode - ACK on error, for N=3, with three losses.

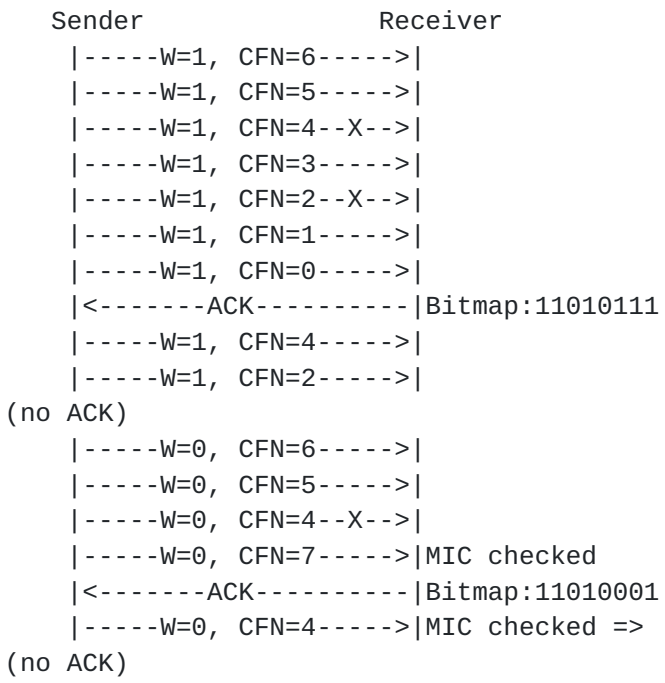

Figure 21: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK on error, for N=3, three losses.

Figure 22 illustrates the transmission of an IPv6 packet that needs 11 fragments in Packet mode - ACK "Always", for N=3, without losses.

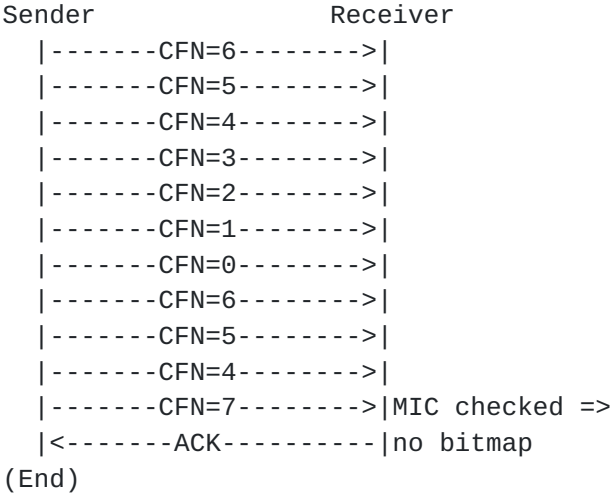

Figure 22: Transmission of an IPv6 packet carried by 11 fragments in Packet mode - ACK "Always", for N=3, no losses.

Figure 23 illustrates the transmission of an IPv6 packet that needs 11 fragments in Packet mode - ACK "Always", for N=3, with three losses.

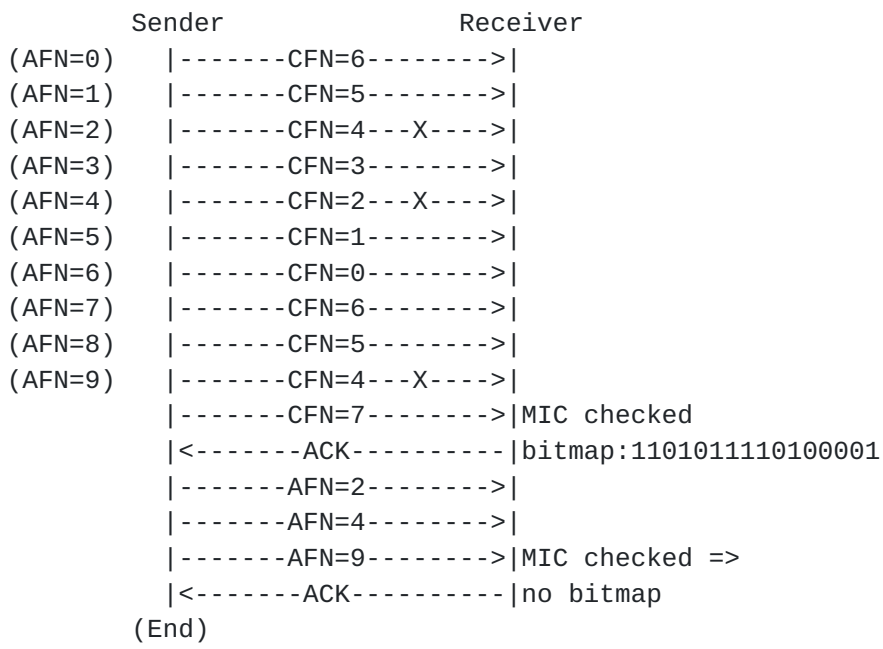

Figure 23: Transmission of an IPv6 packet carried by 11 fragments in Packet mode - ACK "Always", for N=3, with three losses.

Figure 24 illustrates the transmission of an IPv6 packet that needs 11 fragments in Window mode - ACK "Always", for N=3, without losses. Note: in Window mode, an additional bit will be needed to number windows.

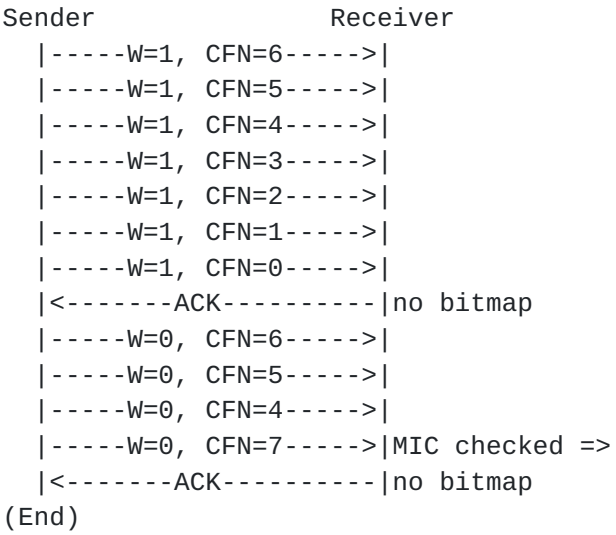

Figure 24: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK "Always", for N=3, no losses.

Figure 25 illustrates the transmission of an IPv6 packet that needs 11 fragments in Window mode - ACK "Always", for N=3, with three losses.

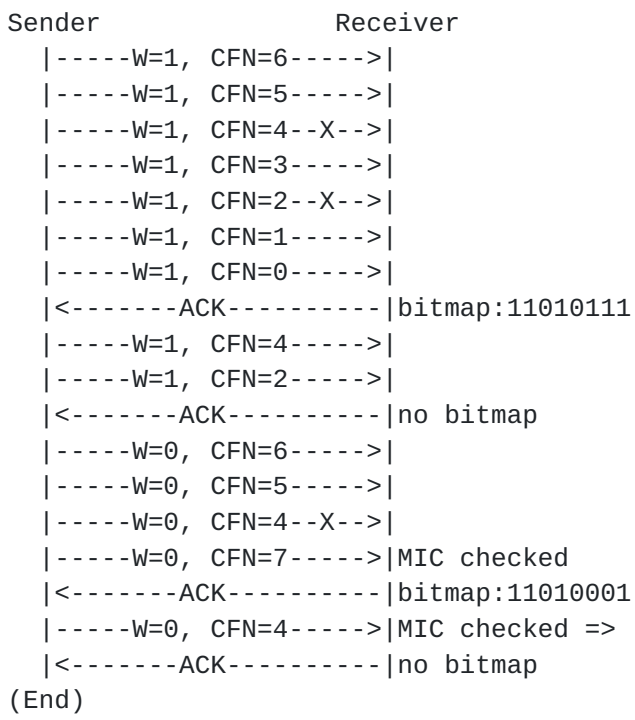

Figure 25: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK "Always", for N=3, with three losses.

Appendix B. 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