lpwan Working Group
Internet-Draft
Intended status: Informational
Expires: April 23, 2018

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

October 20, 2017

LPWAN Static Context Header Compression (SCHC) and fragmentation for IPv6 and UDP
draft-ietf-lpwan-ipv6-static-context-hc-07

Abstract

This document describes a header compression scheme and fragmentation functionality for very low bandwidth networks. These techniques are specially 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. SCHC compression is based on a common static context stored in a LPWAN device and in the network. Static context means that the stored information does not change during the packet transmission. The context describes the field values and keeps information that will not be transmitted through the constrained network.

SCHC must be used for LPWAN networks because it avoids complex resynchronization mechanisms, which are incompatible with LPWAN characteristics. And also because in most cases, IPv6/UDP headers are reduced to a small identifier called Rule ID. Eventhough sometimes, a SCHC compressed packet will not fit in one L2 PDU, and the SCHC fragmentation protocol will be used. The SCHC fragmentation and reassembly mechanism 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.

This document describes the SCHC compression/decompression framework and applies it to IPv6/UDP headers. This document also specifies a fragmentation and reassembly mechanism that is used to support the IPv6 MTU requirement over LPWAN technologies. Fragmentation is mandatory for IPv6 datagrams that, after SCHC compression or when it has not been possible to apply such compression, still exceed the L2 maximum payload size. Similar solutions for other protocols such as CoAP will be described in separate documents.

Status of This Memo

Copyright Notice

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 of 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 of 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 requirement 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)

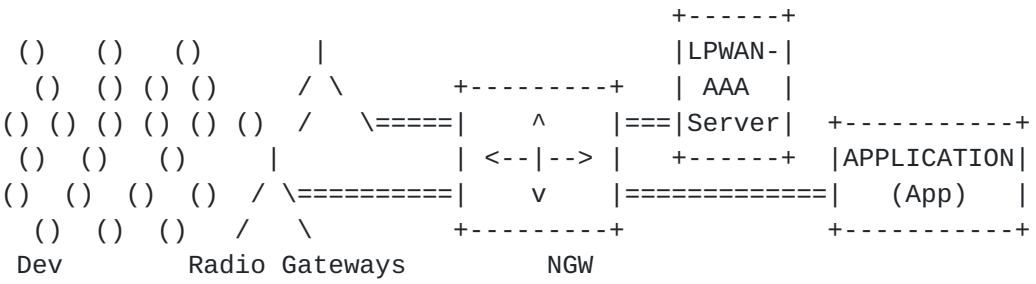


Figure 1: LPWAN Architecture

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 performed for both functionalities 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. A 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 Identifier is an index to describe the header fields in the Rule

- FL: Field Length is a value to identify if the field is fixed or variable length.

- FP: Field Position is a value that is used to identify each instance a field apears in the header.

- 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. A set of Rule IDs are used to support fragmentation functionality.

- 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.

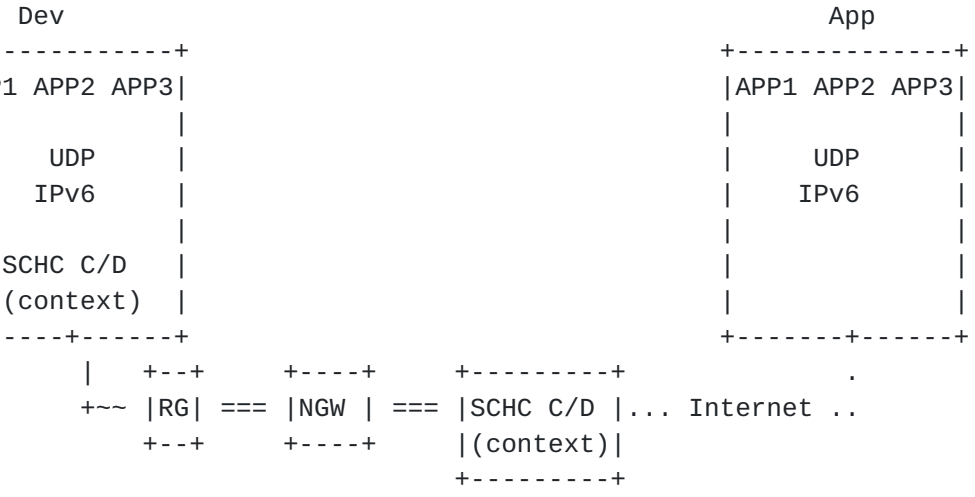


Figure 2: Architecture

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. The resulting information is sent to 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 an 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 matches as much as possible the original packet values. When a value is known by both ends, it is not necessary to send it 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 length (FL), a field position (FP), a direction indicator (DI), a target value (TV), a matching operator (MO) and a Compression/Decompression Action (CDA).

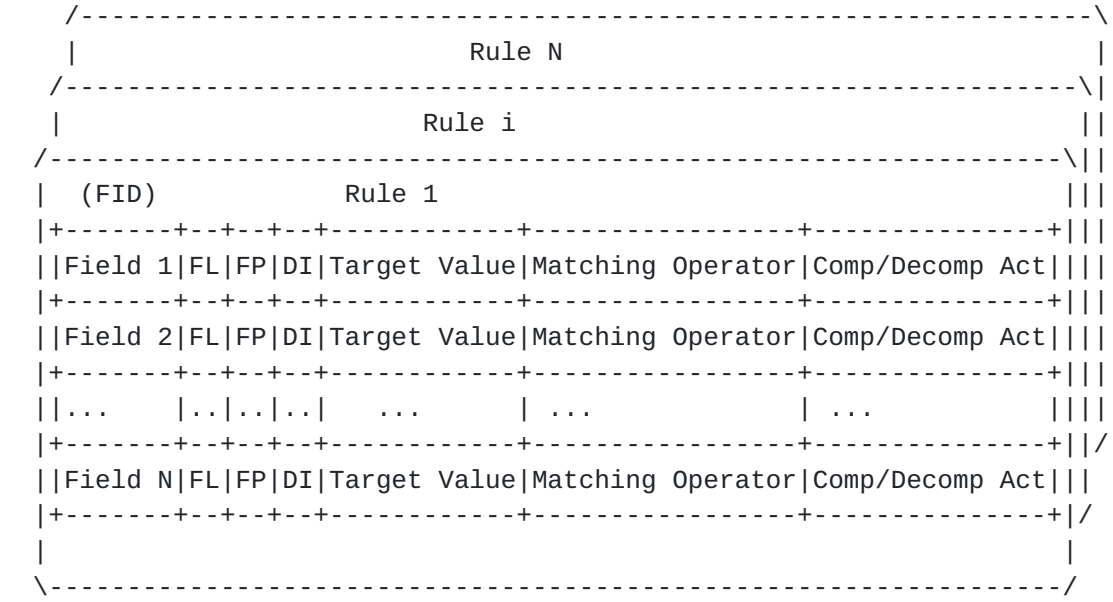


Figure 3: Compression/Decompression Context

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 Length (FL) is the length of the field that can be of fixed length as in IPv6 or UDP headers or variable length as in CoAP options. Fixed length fields shall be represented by its actual value in bits. Variable length fields shall be represented by a function or a variable.

- 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:

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

  * DOWNLINK (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 are reserved for other functionalities 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 be used 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 used 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 the 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 from the order given by the Rule. For instance Compute-* may be applied at the end, after all 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.

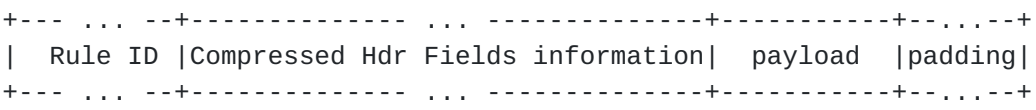


Figure 4: LPWAN Compressed Format Packet

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 summarizes the basics functions defined to compress and decompress a field. The first column gives the action's name. The second and third columns outline 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 being 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 bits 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

The 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 in 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 with the list of values in the Target Value. This function is used together with the "match-mapping" MO.

The compressor looks on 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 for coding 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 the length is not specified, the number of bits sent is 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

In LPWAN technologies, the L2 data unit size typically varies from tens to hundreds of bytes. If the entire IPv6 datagram after applying SCHC header compression or when SCHC is not possible, fits within a single L2 data unit, the fragmentation mechanism is not used and the packet is sent. Otherwise, the datagram SHALL be broken into fragments.

LPWAN technologies impose some strict limitations on traffic, devices are sleeping most of the time and may receive data during a short period of time after transmission to preserve battery. To adapt the SCHC fragmentation to the capabilities of LPWAN technologies, it is desirable to enable optional fragment retransmission and to allow a

gradation of fragment delivery reliability. This document does not make any decision with regard to which fragment delivery reliability option may be 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

This specification defines the following three fragment delivery reliability options:

o No ACK. No ACK is the simplest fragment delivery reliability option. The receiver does not generate overhead in the form of acknowledgments (ACKs). However, this option does not enhance delivery reliability beyond that offered by the underlying LPWAN technology. In the No ACK option, the receiver MUST NOT issue ACKs.

o Window mode - ACK always (ACK-always).
The 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 the end of the IPv6 packet transmission, as long as the window size is short enough. However, such benefit comes at the expense of ACK use. In 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 an all-1 empty 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).

o Window mode - ACK-on-error. The ACK-on-error option is suitable for links offering relatively low L2 data unit loss probability. This option reduces the number of ACKs transmitted by the fragment receiver. 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. However, if an ACK is lost, the sender assumes that all

fragments covered by the ACK have been successfully delivered. And the receiver will abort the fragmentation.
In ACK-on-error, an ACK is transmitted by the fragment receiver after a window of fragments has 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 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).
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. Functionalities

This subsection describes the different fields in the fragmentation header that are used to enable the described fragmentation functionalities and the different reliability options supported.

o Rule ID. The Rule ID in the fragmentation part is used to identify the fragmentation mode used, also to idenitfy fragments from ACK and Abort frames. The also allows to interleave non-fragmented IPv6 datagrams with fragments that carry a larger IPv6 datagram. In the fragments format 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 see format section.

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. There are two reserved values used for the control of the fragmentation. The FCN value when all the bits equals 1 (all-1) denotes the last fragment of a packet. And the FCN value when all the bits equals 0 (all-0) denotes the last fragment of the windonw in any window mode or the fragments in No ACK mode. The rest of the FCN values are used in a sequential and decreasing order,

which has the purpose to avoid possible ambiguity for the receiver that might arise under certain conditions. In the fragments, this field is an unsigned integer, with a size of N bits. In the No ACK mode it is set to 1 bit (N=1). For the other modes it is recommended to use a number of bits (N) equal to or greater than 3. 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 FCN for the last fragment in a window is an all-0, which indicates that the window is finished and it proceeds according to the mode in use: either an ack is sent or the next window fragments are expected when there is no error. The FCN for the last fragment is an all-1. It is also important to note that, for No ACK mode or 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.

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. In the fragment formats the size of the DTag field is T bits, which may be set to a value greater than or equal to 0 bits. DTag MUST be set sequentially increasing from 0 to 2^T - 1, and MUST wrap back from 2^T - 1 to 0. In the ACK format, DTag carries the same value as the DTag field in the fragments for which this ACK is intended.

o W (window): 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 0. In the ACK format, 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.

o Message Integrity Check (MIC). This field, which has a size of M bits. It is computed by the sender over the complete packet (i.e. a SCHC compressed or an uncompressed IPv6 packet) before fragmentation. The algorithm to be used to compute the MIC is not defined in this document, and needs to be defined in other documents (e.g. technology-specific profiles). The MIC allows the receiver to check errors in the reassembled packet, while it also enables compressing the UDP checksum by use of SCHC compression.

o Bitmap. The bitmap is a sequence of bits included in the ACK for a given window, each bit in the Bitmap identifies a fragment. It provides feedback on whether each fragment of the current window has been received or not. FCN set to All-0 and All-1 fragments are set to the right-most position on the bitmap in this order. Highest FCN is set to the left-most position. A bit set to 1 indicates that the

corresponding FCN fragment has been correctly sent and received. TODO (it is missing to explain the optimization of bitmap in order to have a way to send an abort)

5.4. Formats

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

5.4.1. 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 a SCHC header or an IPv6 header or the original IPv6 packet payload which could not be compressed. A fragment is the payload in the L2 protocol data unit (PDU).

Figure 6: Fragment format.

5.4.2. 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.

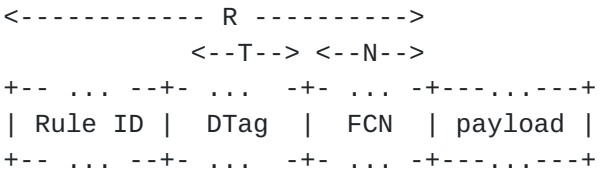


Figure 7: Fragmentation Header for Fragments except the Last One, No ACK option

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.

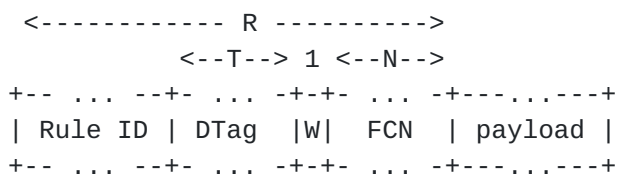


Figure 8: Fragmentation Header for Fragments except the Last One, Window mode

5.4.3. ACK format

The format of an ACK is shown in Figure 9:

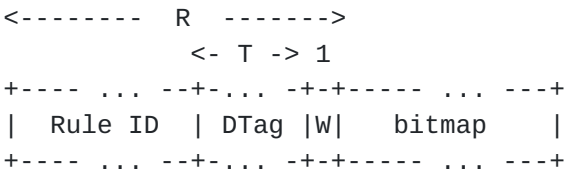


Figure 9: Format of an ACK

Figure 10 shows an example of an ACK (N=3), where the bitmap indicates that the second and the fifth fragments have not been correctly received.

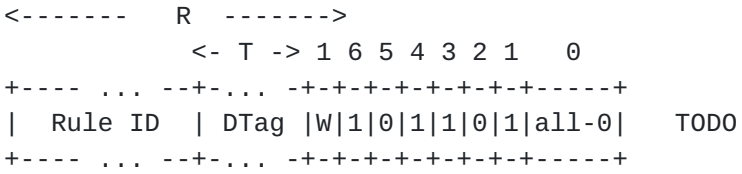


Figure 10: Example of the bitmap in Window mode, in any window unless the last one, for N=3)

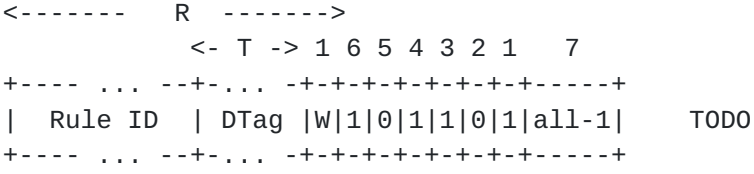


Figure 11: Example of the bitmap in Window mode for the last window, for N=3)

5.4.4. All-1 and All-0 formats

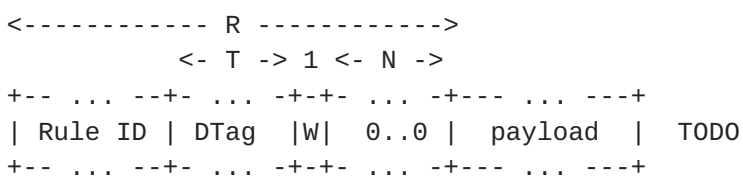


Figure 12: All-0 format fragment

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 14. The total size of this fragmentation header is R+M bits.

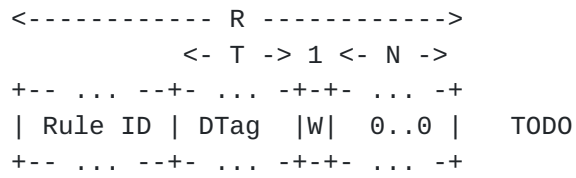


Figure 13: All-0 empty format fragment

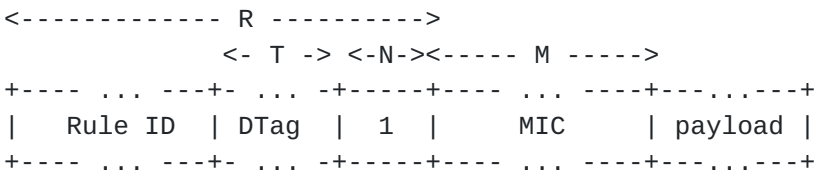


Figure 14: All-1 Fragmentation Header for the Last Fragment, No ACK option

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 15. The total size of this fragmentation header is R+M bits. It is used for retransmissions

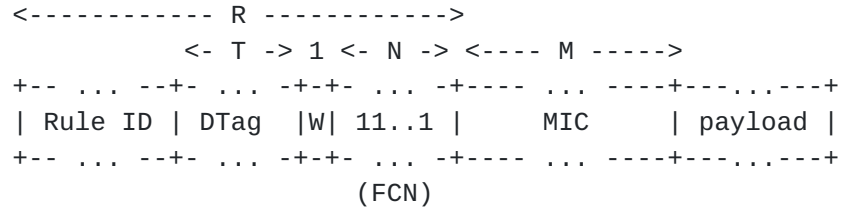


Figure 15: All-1 Fragmentation Header for the Last Fragment, Window mode

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

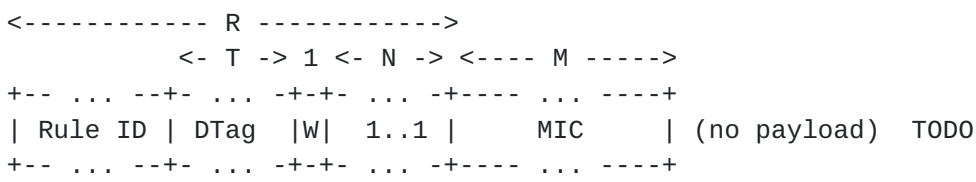


Figure 16: All-1 for Retries format fragment also called All-1 empty

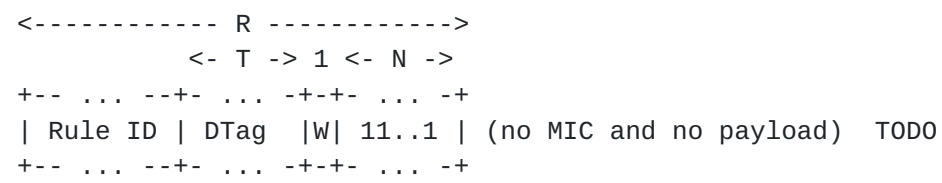


Figure 17: All-1 for Abort format fragment

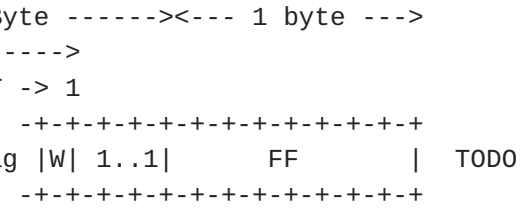


Figure 18: ACK Abort format fragment

5.5. Baseline mechanism

The receiver needs to identify all the fragments that belong to a given IPv6 datagram. To this end, the receiver SHALL use: * The sender's L2 source address (if present), * The destination's L2 address (if present), * Rule ID and * DTag (the latter, if present). Then, the fragment receiver may determine the fragment delivery reliability option that is used for this 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. A fragment payload may carry bytes from a SCHC compressed IPv6 header, an uncompressed IPv6 header or an IPv6 datagram data payload. An unfragmented packet could be a SCHC compressed or an uncompressed IPv6 packet (header and data). For example, the receiver may place the fragment payload within a payload datagram reassembly buffer at

the location determined from: the FCN, the arrival order 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 packet cannot be determined from fragmentation headers.

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 all-0 is considered the last fragment of its window, except for the last fragment of the whole packet (all-1), which is also the last fragment of the last window.

If the recipient receives the last fragment of a datagram (all-1), it checks for the integrity of the reassembled datagram, based on the MIC received. In No ACK, if the integrity check indicates that the reassembled datagram does not match the original datagram (prior to fragmentation), the reassembled 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 ACK always, if a MIC check indicates that the datagram has been successfully reassembled, the fragment receiver sends an ACK without a bitmap to the fragment sender.

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.

TODO (explain the Bitmap optimization)

5.6. 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 packets that need to be carried by a large number of fragments. However, when the number of fragments required to carry an 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 packet transmission.

TODO (does it works for ACK-on-error?)

5.7. Aborting fragmented 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.

TODO (explain the abort format packets)

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

5.8. 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 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.

5.9. Fragmentation Mode of Operation Description

The fragmentation is based on the FCN value, which has a length of N bits. The All-1 and All-0 values are reserved, and are used to control the fragmentation transmission. The FCN will be sent in downwards position this means from larger to smaller and the number of bits depends on the implementation. The last fragment in all modes must contains a MIC which is used to check if there are error or missing fragments.

5.9.1. No ACK Mode

In the No ACK mode there is no feedback communication. The sender will send the fragments until the last one whithout any possibility to know if there were an error or lost. As there is not any control one bit FCN is used, where FCN all-0 will be sent for all the fragments except the last one which will use FCN to all-1 and will send the MIC. Figure 19 shows the state machine for the sender.

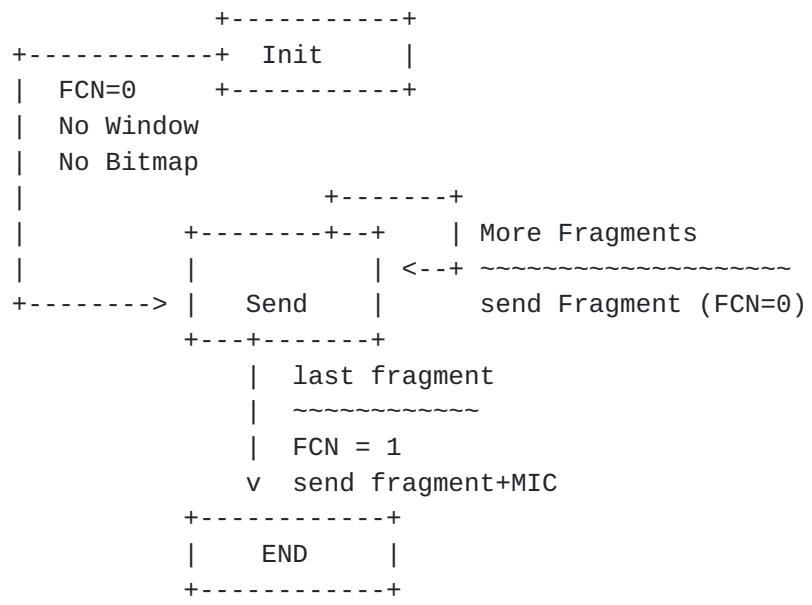


Figure 19: Sender State Machine for the No ACK Mode

The receiver waits for fragments and will set a timer in order to see if there is no missing fragments. The No ACK mode will use the MIC contained in the last fragment to check error. The FCN is set to all-1 for the last fragment. Figure 20 shows the state machine for the receiver. When the Timer expires or when the check of MIC gives an error it will abort the communication and go to error state, all the fragments will be dropped. The Inactivity Timer will be based on the LPWAN technology and will be defined in the specific technology document.

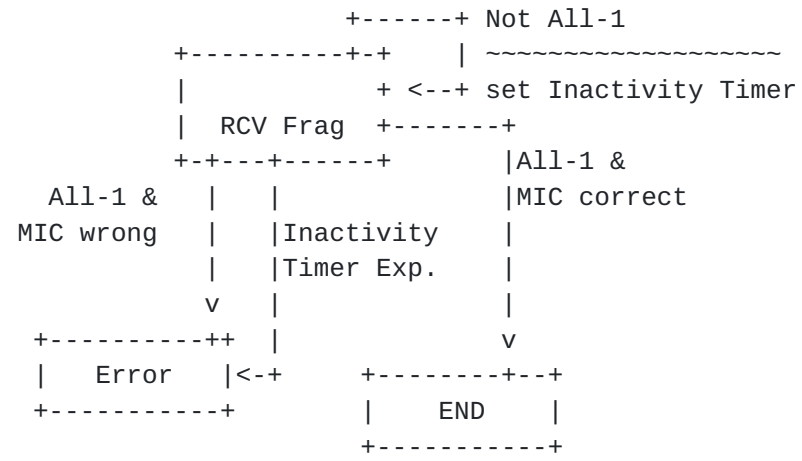


Figure 20: Receiver State Machine for the No ACK Mode

5.9.2. The Window modes

The jumping window protocol is using two windows alternatively 0 and 1. The FCN to all-0 fragment means that the window is over and allows to switch from one window to another. The FCN to all-1 fragment indicates that it is the last fragment and there will not be another window.

In all the cases, the sender may not have to send all the fragments contained in the window. To ease FN (fragment number) reconstruction from FCN, it is recommended to send sequentially all the fragments on a window and for all non-terminating window to fill entirely the window.

The receiver generates the bitmap which may have the size of a single frame based on the size of downlink frame of the LPWAN technology used. When the bitmap cannot be sent in one frame or for the last window,

, then first the FCN should be set to the lowest possible value.

The Window mode has two different mode of operation: The ACK on error and the ACK always.

5.9.3. ACK Always

The Figure 21 finite state machine describes the sender behavior. Intially, when a fragmented packet need to be sent, the window is set to 0, a local_bit map is set to 0, and FCN is set the the highest possible value depending on the number of fragment that will be sent in the window (INIT STATE).

The sender starts sending fragments (SEND STATE), the sender will indicate in the fragmentation header: the current window and the FCN number. A delay between each fragment can be added to respect regulation rules or constraints imposed by the applications. Each time a fragment is sent the FCN is decreased of one value and the bitmap is set. The send state can be leaved for different reasons (for both reasons it goes to WAIT BITMAP STATE):

- The FCN reaches value 0 and there are more fragments. In that case an all-0 fragmet is sent and the timer is set. The sender will wait for the bitmap acknowledged by the receiver.

- The last fragment is sent. In that case an all-1 fragment with the MIC is sent and the sender will wait for the bitmap acknowledged by the receiver. The sender set a timer to wait for the ack.

During the transition between the SEND state of the current window and the WAIT BITMAP, the sender start listening to the radio and start a timer. This timer is dimensioned to the receiving window depending on the LPWAN technology.

In ACK Always, if the timer expire, an empty All-0 (or All-1 if the last fragment has been sent) fragment is sent to ask the receiver to resent its bitmap. The window number is not changed.

The sender receives a bitmap, it checks the window value. Acknowledgment with the non expected window are discarded.

If the window number on the received bitmap is correct, the sender compares the local bitmap with the received bitmap. If they are equal all the fragments sent during the window have been well received. If at least one fragment need to be sent, the sender clear the bitmap, stop the timer and move its sending window to the next value. If no more fragments have to be sent, then the fragmented packet transmission is terminated.

If some fragments are missing (not set in the bit map) then the sender resend the missing fragments. When the retransmission is finished, it start listening to the bitmap (even if a All-0 or All-1 has not been sent during the retransmission) and returns to the waiting bitmap state.

If the local-bitmap is different from the received bitmap the counter Attemps is increased and the sender resend the missing fragments again, when a MAX_ATTEMPS is reached the sender sends an Abort and goes to error.

Figure 21: Sender State Machine for the ACK Always Mode

The Figure 22 finite state machine describes the receiver behavior. The receiver starts with window 0 as the expecting window and

maintain a local_bitmap indicating which fragments it has received (all-0 and all-1 occupy the same position).

Any fragment not belonging to the current window is discarded. Fragment belonging to the correct window are accepted, FN is computed based on the FCN value. The receiver leaves this state when receiving a:

- All-0 fragment which indicates that all the fragments have been sent in the current window. Since the sender is not obliged to send a full window, some fragment number not set in the local_bitmap may not correspond to losses.

- All-1 fragment which indicated that the transmission is finished. Since the last window is not full, the MIC will be used to detect if all the fragments have been received.

A correct MIC indicates the end of the transmission. The receiver must stay in this state during a period of time to answer to empty all-1 frag the sender may send if the bitmap is lost.

If All-1 frag has not been received, the receiver expect a new window. It waits for the next fragment. If the window value has not changed, the received fragments are part of a retransmission. A receiver that has already received a frag should discard it (not represented in the state machine), otherwise it completes its bitmap. If all the bit of the bitmap are set to one, the receiver may send a bitmap without waiting for a all-0 frag.

If the window value is set to the next value, this means that the sender has received a correct bitmap, which means that all the fragments have been received. The receiver change the value of the expected window.

If the receiver receives an all-0 fragment, it stays in the same state. Sender may send more one fragment per window or more. Otherwise some fragments in the window have been lost.

If the receiver receives an all-1 fragment this means that the transmission should be finished. If the MIC is incorrect some fragments have been lost. It sends its bitmap.

In case of an incorrect MIC, the receivers wait for fragment belonging to the same window.

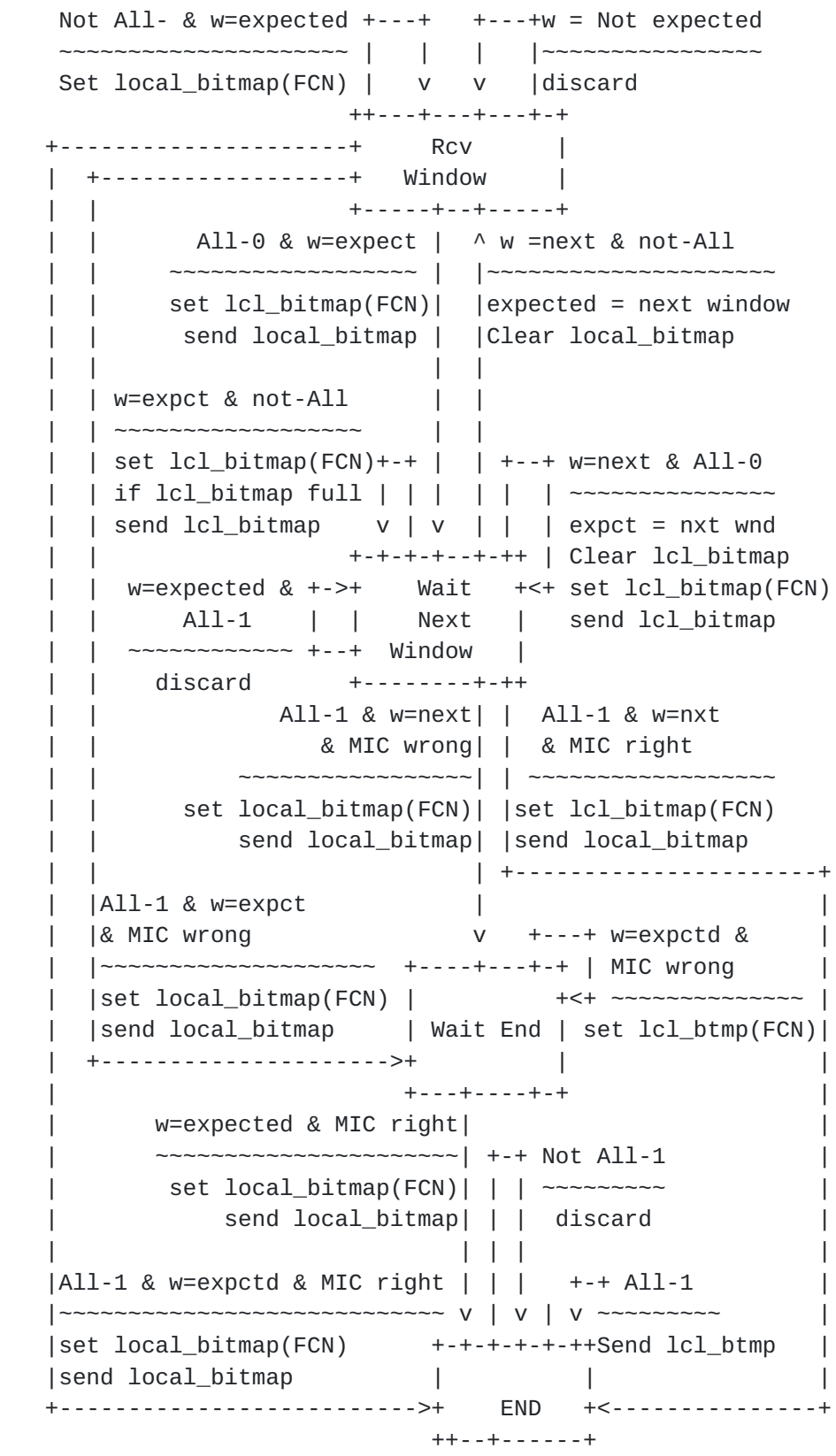


Figure 22: Receiver State Machine for the ACK Always Mode

5.9.4. ACK on error

The ACK on error sender is very similar to the ACK always sender, Intially, when a fragmented packet is sent, the window is set to 0, a local_bit map is set to 0, and FCN is set the highest possible value depending on the number of fragment that will be sent in the window. See Figure 23

The sender starts sending fragments indicating in the fragmentation header with the current window and the FCN number. A delay between each fragment can be added to respect regulation rules or constraints imposed by the applications. This state can be leaved for different reasons:

- The FCN reaches value 0. In that case a all-0 fragmet is sent and the sender will wait for the bitmap acknowledged by the receiver.

- The last fragment is sent. In that case a all-1 fragment is sent and the sender will wait for the bitmap acknowledged by the receiver.

During the transition between the sending the fragment of the current window and waiting for bitmap, the sender start listening to the radio and start a timer. This timer is dimensioned to the receiving window depending on the LPWAN technology.

In Ack on error mode, the timer expiration will be considered as a positive acknowledgment. If there are no more fragments then the fragmentation is finished.

If the sender receives a bitmap, it checks the window value. Acknowledgment with the non expected window are discarded.

If the window number on the received bitmap is correct, the sender compare the local bitmap with the received bitmap. If they are equal all the fragments sent during the window have been well received. If at least one fragment need to be sent, the sender clear the bitmap, stop the timer and move its sending window to the next value. If no more fragments have to be sent, then the fragmented packet transmission is terminated.

If some fragments are missing (not set in the bit map) then the sender resend the missing fragments. When the retransmission is finished, it start listening to the bitmap (even if a All-0 or All-1 has not been sent during the retransmission) and returns to the waiting bitmap state.

If the local-bitmap is different from the received bitmap the counter Attemps is increased and the sender resend the missing fragments again, when a MAX_ATTEMPS is reached the sender sends an Abort and goes to error.

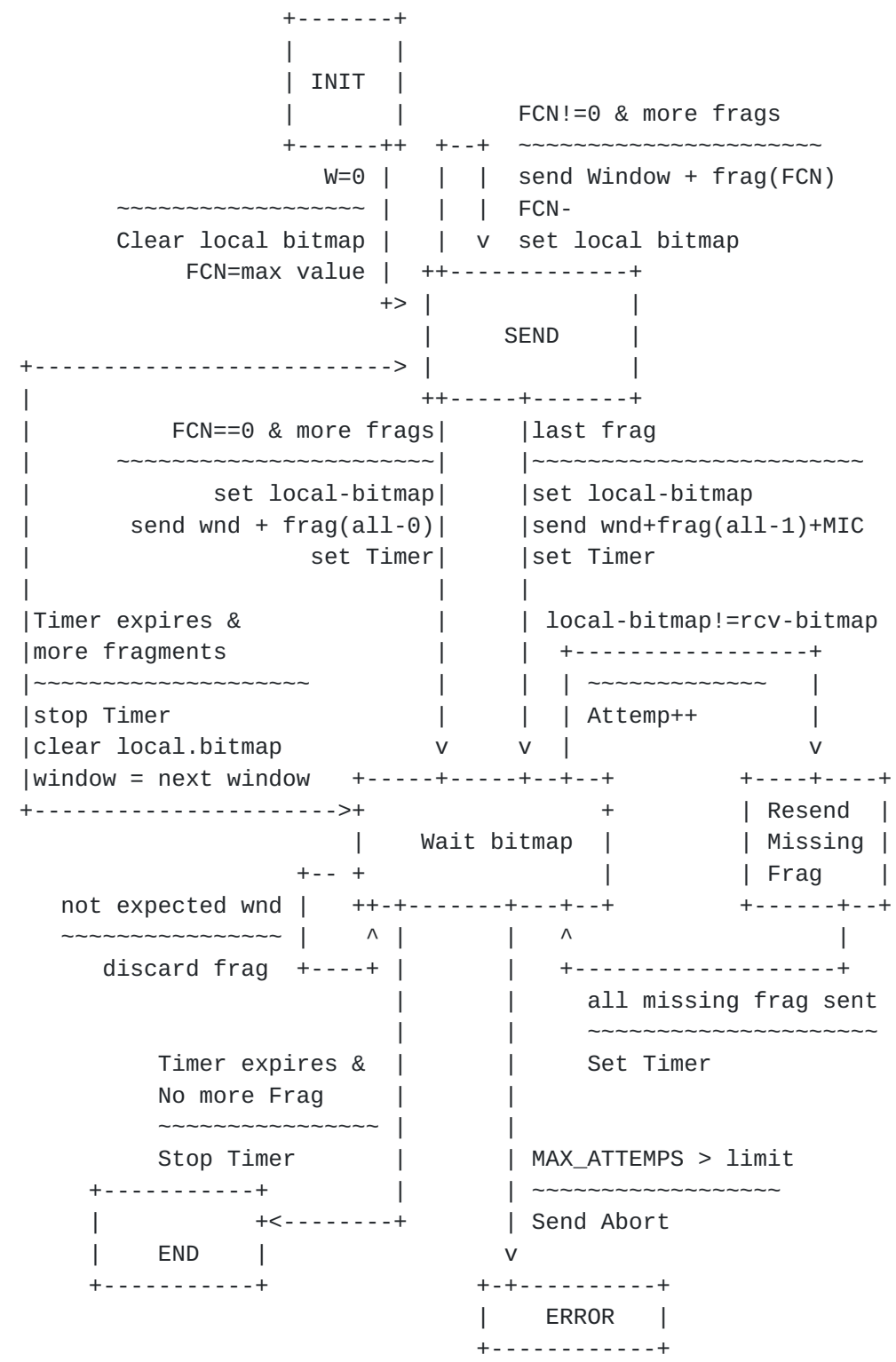


Figure 23: Sender State Machine for the ACK on error Mode

Unlike the sender, the receiver for ACK on error has some differences. First we are not sending the bitmap unless there is an

error or an unexpected behavior. The Figure 24 finite state machine describes the receiver behavior. The receiver starts with an the expecting window and maintain a local_bitmap indicating which fragments it has received (all-0 and all-1 occupy the same position).

Any fragment not belonging to the current window is discarded. Fragment belonging to the correct window are accepted, FN is computed based on the FCN value. When an All-0 fragment is received and the bitmap is full the receiver changes the window value and clear the bitmap. The receiver leaves this state when receiving a:

- All-0 fragment and not a full bitmap indicate that all the fragments have been sent in the current window. Since the sender is not obliged to send a full window, some fragment number not set in the local_bitmap may not correspond to losses. As the receiver does not know if the missing fragments are looses or normal missing fragments it sned s a local bitmap.

- All-1 fragment which indicates that the transmission is finished. Since the last window is not full, the MIC will be used to detect if all the fragments have been received. A correct MIC indicates the end of the transmission.

If All-1 frag has not been received, the receiver expect a new window. It waits for the next fragment. If the window value has not changed, the received fragments are part of a retransmission. A receiver that has already received a frag should discard it (not represented in the state machine), otherwise it completes its bitmap. If all the bits of the bitmap are set to one, the receiver clear the bitmap and wait for the next window without waiting for a all-0 frag. While the receiver waits for next window and if the window value is set to the next value, and all-1 fragment with the next value window arrived the receiver goes to error and abort the transmission, it drops the fragments.

If the receiver receives an all-0 fragment, it stays in the same state. Sender may send more one fragment per window or more. Otherwise some fragments in the window have been lost.

If the receiver receives an all-1 fragment this means that the transmission should be finished. If the MIC is incorrect some fragments have been lost. It sends its bitmap.

In case of an incorrect MIC, the receivers wait for fragment belonging to the same window.

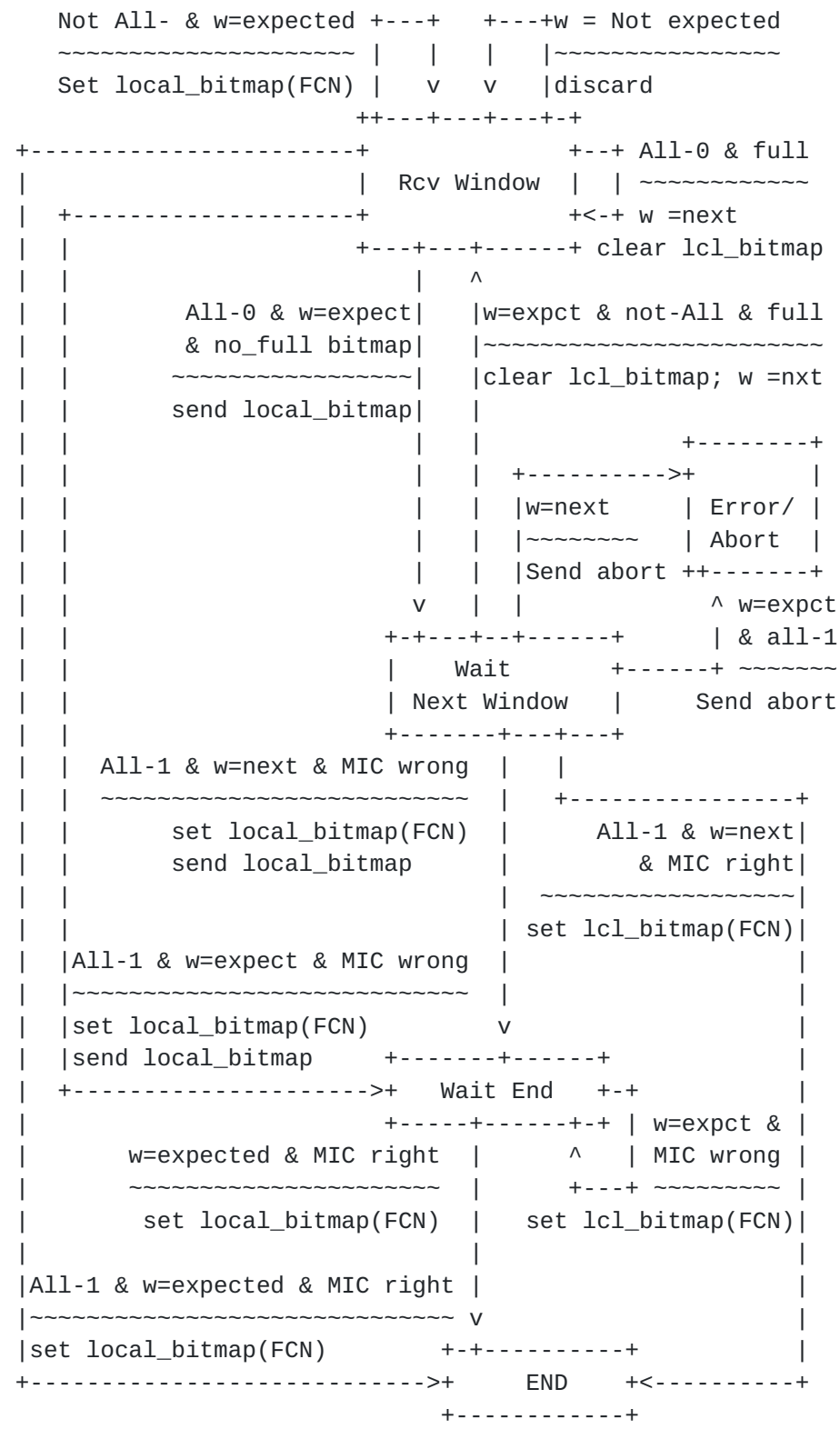


Figure 24: Receiver State Machine for the ACK on error Mode

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.

In Window mode - ACK on error, a malicious node may force a fragment sender to resend a fragment a number of times, with the aim to increase consumption of the fragment sender's resources. To this end, the malicious node may repeatedly send a fake ACK to the fragment sender, with a bitmap that reports that one or more fragments have been lost. In order to mitigate this possible attack, MAX_FRAG_RETRIES may be set to a safe value which allows to limit the maximum damage of the attack to an acceptable extent. However, note that a high setting for MAX_FRAG_RETRIES benefits fragment delivery reliability, therefore the trade-off needs to be carefully considered.

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-07 (work in progress), October 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 25 presents the protocol stack for this Device. IPv6 and UDP are represented with dotted lines since these protocols are compressed on the radio link.

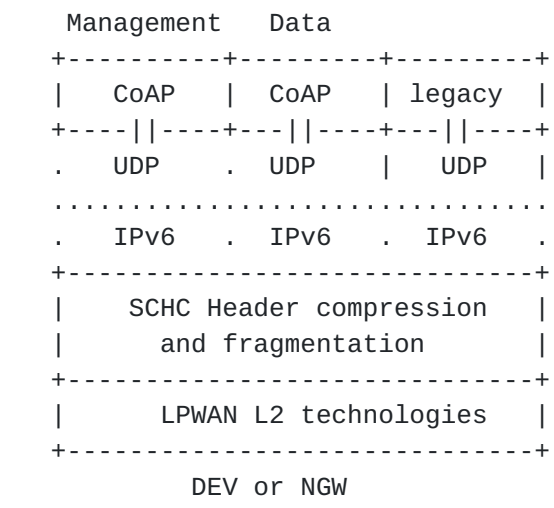


Figure 25: Simplified Protocol Stack for LP-WAN

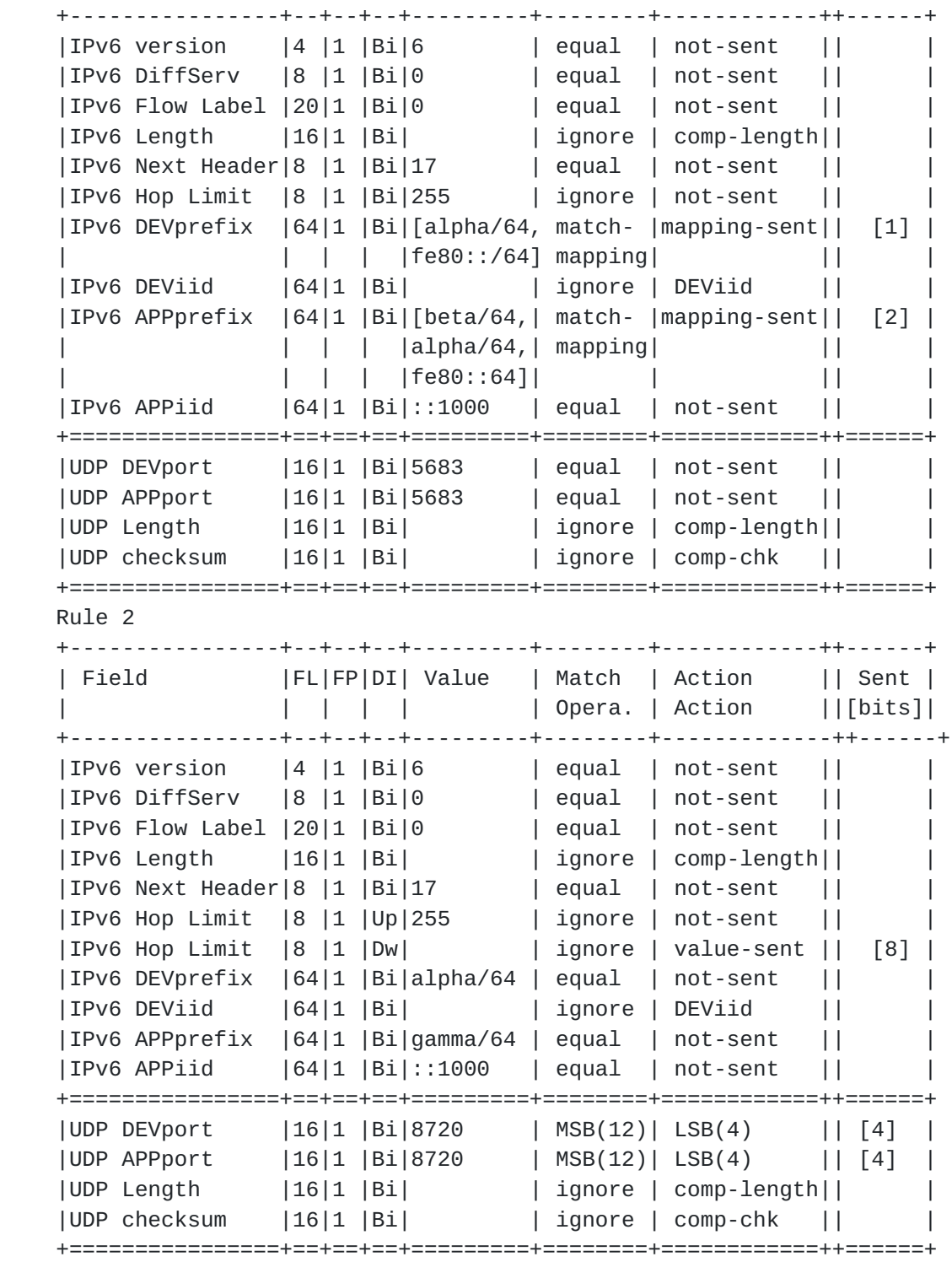
Note that in some LPWAN technologies, only the Devs have a device ID. Therefore, when such technologies are used, it is necessary to define statically an IID for the Link Local address for the SCHC C/D.

Rule 0

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

Rule 1

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



```
+----------------+--+--+--+---------+--------+------------++------+
|IPv6 version    |4 |1 |Bi|6        | equal  | not-sent   ||      |
|IPv6 DiffServ   |8 |1 |Bi|0        | equal  | not-sent   ||      |
|IPv6 Flow Label |20|1 |Bi|0        | equal  | not-sent   ||      |
|IPv6 Length     |16|1 |Bi|         | ignore | comp-length||      |
|IPv6 Next Header|8 |1 |Bi|17       | equal  | not-sent   ||      |
|IPv6 Hop Limit  |8 |1 |Bi|255      | ignore | not-sent   ||      |
|IPv6 DEVprefix  |64|1 |Bi|[alpha/64, match- |mapping-sent||  [1] |
|                |  |  |  |fe80::/64] mapping|            ||      |
|IPv6 DEViid     |64|1 |Bi|         | ignore | DEViid     ||      |
|IPv6 APPprefix  |64|1 |Bi|[beta/64,| match- |mapping-sent||  [2] |
|                |  |  |  |alpha/64,| mapping|            ||      |
|                |  |  |  |fe80::64]|        |            ||      |
|IPv6 APPiid     |64|1 |Bi|::1000   | equal  | not-sent   ||      |
+================+==+==+==+=========+========+============++======+
|UDP DEVport     |16|1 |Bi|5683     | equal  | not-sent   ||      |
|UDP APPport     |16|1 |Bi|5683     | equal  | not-sent   ||      |
|UDP Length      |16|1 |Bi|         | ignore | comp-length||      |
|UDP checksum    |16|1 |Bi|         | ignore | comp-chk   ||      |
+================+==+==+==+=========+========+============++======+
Rule 2
+----------------+--+--+--+---------+--------+------------++------+
| Field          |FL|FP|DI| Value   | Match  | Action     || Sent |
|                |  |  |  |         | Opera. | Action     ||[bits]|
+----------------+--+--+--+---------+--------+------------++------+
|IPv6 version    |4 |1 |Bi|6        | equal  | not-sent   ||      |
|IPv6 DiffServ   |8 |1 |Bi|0        | equal  | not-sent   ||      |
|IPv6 Flow Label |20|1 |Bi|0        | equal  | not-sent   ||      |
|IPv6 Length     |16|1 |Bi|         | ignore | comp-length||      |
|IPv6 Next Header|8 |1 |Bi|17       | equal  | not-sent   ||      |
|IPv6 Hop Limit  |8 |1 |Up|255      | ignore | not-sent   ||      |
|IPv6 Hop Limit  |8 |1 |Dw|         | ignore | value-sent ||  [8] |
|IPv6 DEVprefix  |64|1 |Bi|alpha/64 | equal  | not-sent   ||      |
|IPv6 DEViid     |64|1 |Bi|         | ignore | DEViid     ||      |
|IPv6 APPprefix  |64|1 |Bi|gamma/64 | equal  | not-sent   ||      |
|IPv6 APPiid     |64|1 |Bi|::1000   | equal  | not-sent   ||      |
+================+==+==+==+=========+========+============++======+
|UDP DEVport     |16|1 |Bi|8720     | MSB(12)| LSB(4)     || [4]  |
|UDP APPport     |16|1 |Bi|8720     | MSB(12)| LSB(4)     || [4]  |
|UDP Length      |16|1 |Bi|         | ignore | comp-length||      |
|UDP checksum    |16|1 |Bi|         | ignore | comp-chk   ||      |
+================+==+==+==+=========+========+============++======+
```

Figure 26: Context rules

All the fields described in the three rules depicted on Figure 26 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 27 illustrates the transmission of an IPv6 packet that needs 11 fragments in the No ACK option, FCN is always 1 bit.

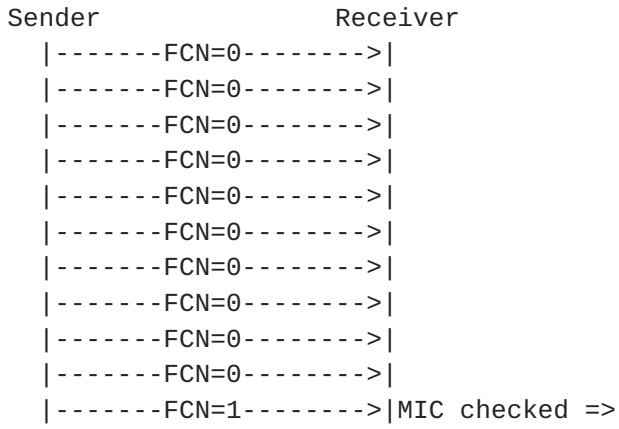


Figure 27: Transmission of an IPv6 packet carried by 11 fragments in the No ACK option

Figure 28 illustrates the transmission of an IPv6 packet that needs 11 fragments in Window mode - ACK on error, for N=3, without losses.

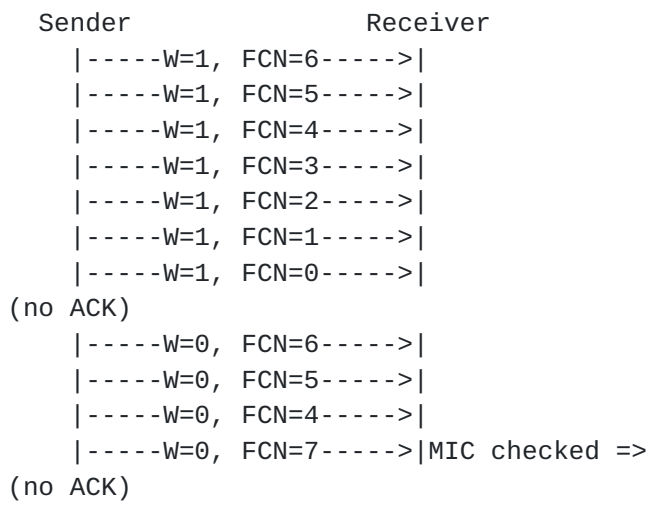


Figure 28: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK on error, for N=3 and MAX_WIND_FCN=6, without losses.

Figure 29 illustrates the transmission of an IPv6 packet that needs 11 fragments in Window mode - ACK on error, for N=3, with three losses.

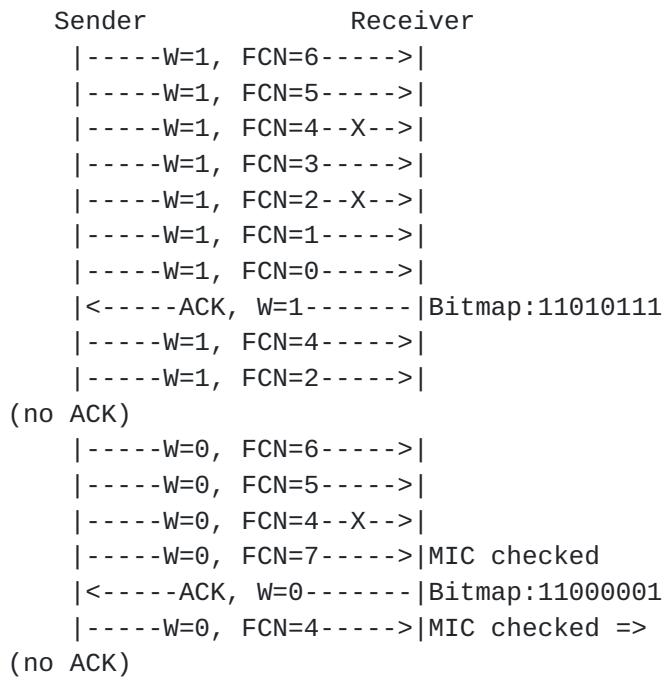


Figure 29: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK on error, for N=3 and MAX_WIND_FCN=6, three losses.

Figure 30 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.

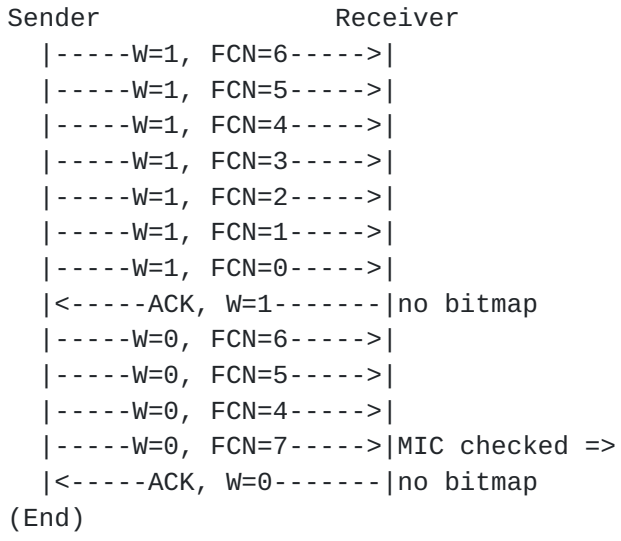


Figure 30: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK "always", for N=3 and MAX_WIND_FCN=6, no losses.

Figure 31 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.

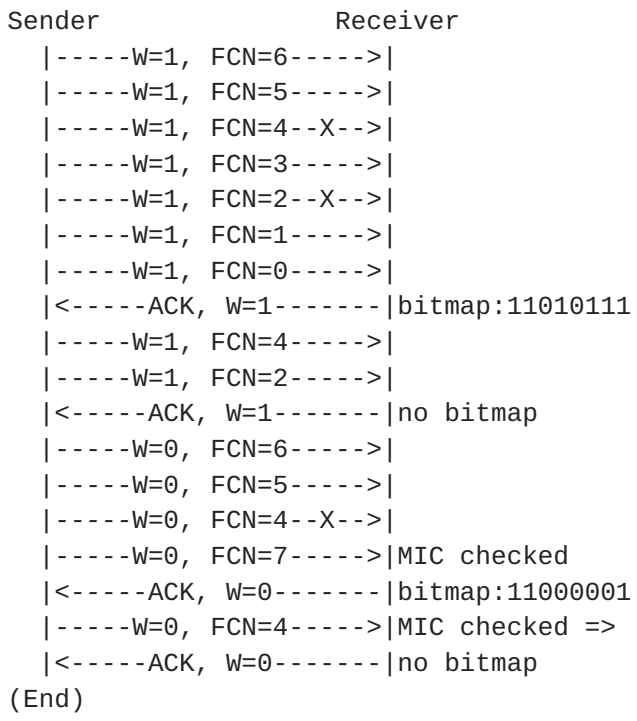


Figure 31: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK "Always", for N=3, and MAX_WIND_FCN=6, with three losses.

Figure 32 illustrates the transmission of an IPv6 packet that needs 6 fragments in Window mode - ACK "always", for N=3 and MAX_WIND_FCN=6, with three losses, and only one retry is needed for each lost fragment. Note that, since a single window is needed for transmission of the IPv6 packet in this case, the example illustrates behavior when losses happen in the last window.

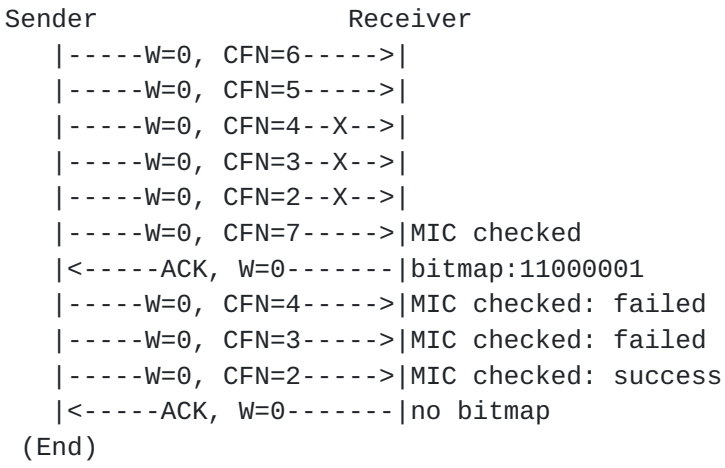


Figure 32: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK "Always", for N=3, and MAX_WIND_FCN=6, with three losses, and only one retry is needed for each lost fragment.

Figure 33 illustrates the transmission of an IPv6 packet that needs 6 fragments in Window mode - ACK "always", for N=3 and MAX_WIND_FCN=6, with three losses, and the second ACK is lost. Note that, since a single window is needed for transmission of the IPv6 packet in this case, the example illustrates behavior when losses happen in the last window.

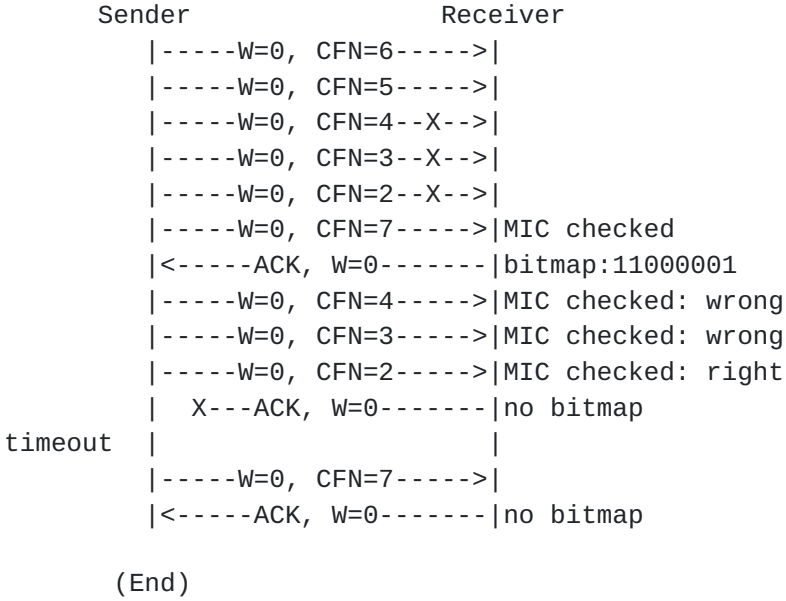


Figure 33: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK "Always", for N=3, and MAX_WIND_FCN=6, with three losses, and the second ACK is lost.

Figure 34 illustrates the transmission of an IPv6 packet that needs 6 fragments in Window mode - ACK "always", for N=3 and MAX_WIND_FCN=6, with three losses, and one retransmitted fragment is lost. Note that, since a single window is needed for transmission of the IPv6 packet in this case, the example illustrates behavior when losses happen in the last window.

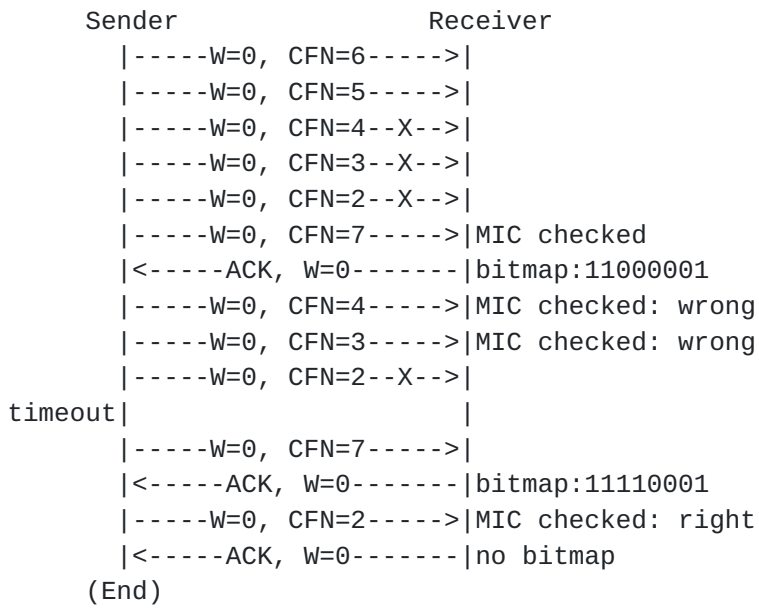
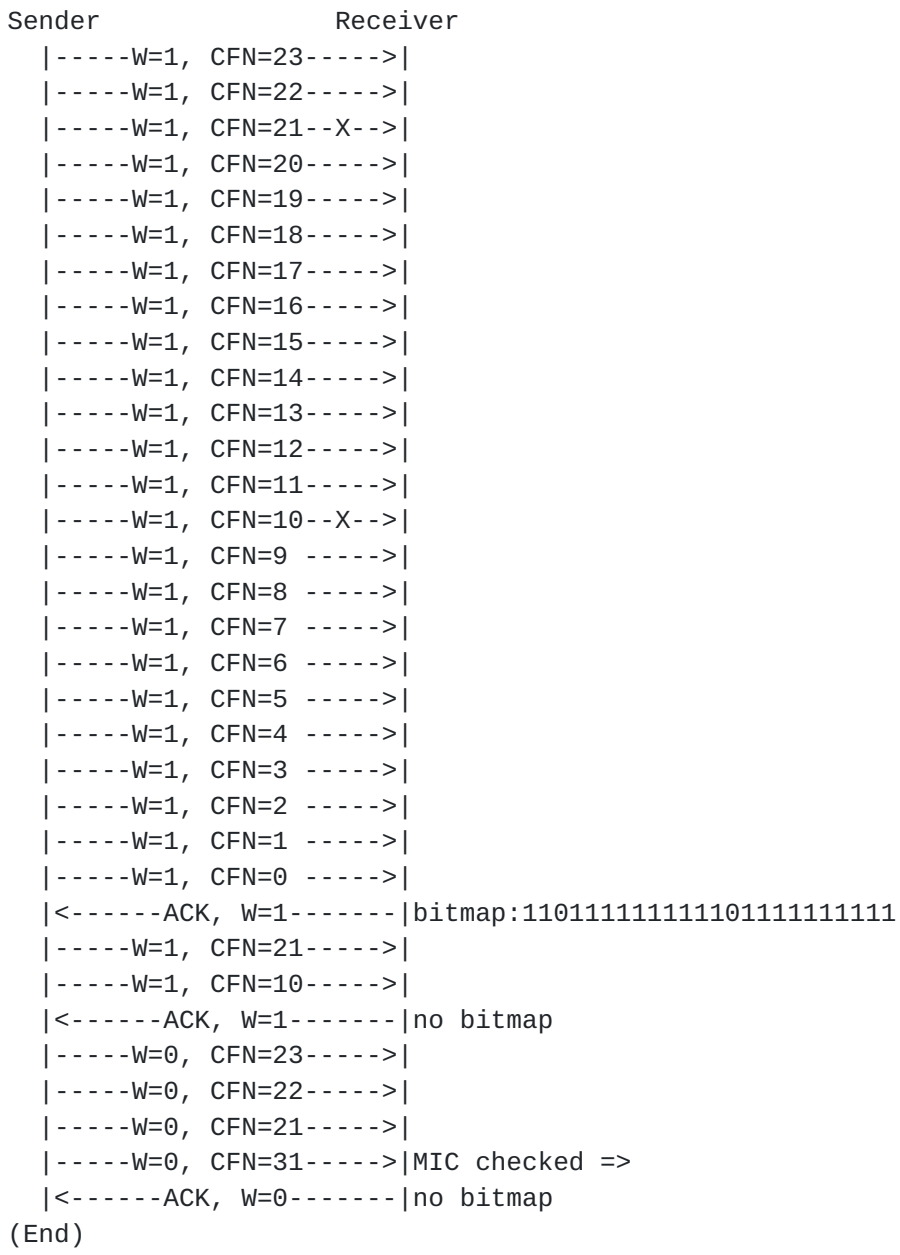


Figure 34: Transmission of an IPv6 packet carried by 11 fragments in Window mode - ACK "Always", for N=3, and MAX_WIND_FCN=6, with three losses, and one retransmitted fragment is lost.

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