lpwan Working Group
Internet-Draft
Intended status: Informational
Expires: June 25, 2018

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

December 22, 2017

LPWAN Static Context Header Compression (SCHC) and fragmentation for IPv6 and UDP

draft-ietf-lpwan-ipv6-static-context-hc-09

Abstract

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

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 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 with SCHC, in most cases IPv6/UDP headers can be reduced to a small identifier called Rule ID. Even though, sometimes, a SCHC compressed packet will not fit in one L2 PDU, and the SCHC fragmentation protocol defined in this document may be used.

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 (RGW), which is the end point of the constrained link.

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

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

o Application Server (App)

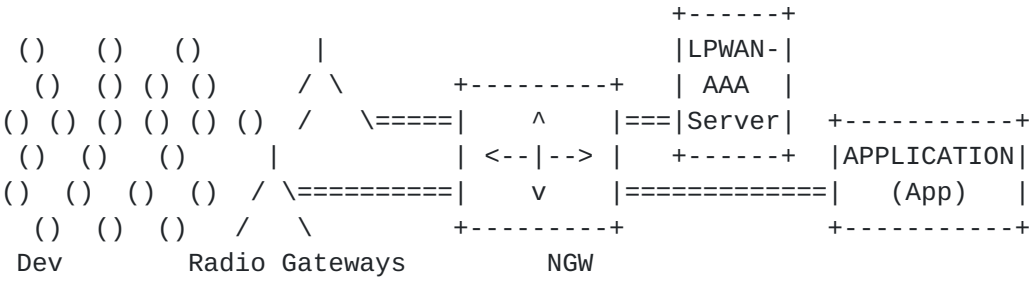


Figure 1: LPWAN Architecture

3. Terminology

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

o All-0. Fragment format for the last frame of a window.

o All-1. Fragment format for the last frame of a packet.

o All-0 empty. Fragment format without payload for requesting the Bitmap when the Retransmission Timer expires in a window that is not the last one for a fragmented packet transmission.

o All-1 empty. Fragment format without payload for requesting the Bitmap when the Retransmission Timer expires in the last window.

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

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

o Bi: Bidirectional, a rule entry that applies in both directions.

- C: Checked bit. Used in an acknowledgment (ACK) header to determine when the MIC is correct (1) or not (0).

- 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 appears in the header.

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

- Inactivity Timer. A timer to end the fragmentation state machine when there is an error and there is no possibility to continue an on-going fragmented packet transmission.

- 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.

- Retransmission Timer. A timer used by the fragment sender during an on-going fragmented packet transmission to detect possible link errors when waiting for a possible incoming ACK.

- Rule: A set of header field values.

- Rule entry: A row in the rule that describes a header field.

- 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 fragment header field used in Window mode (see section 5), which carries the same value for all fragments of a window.

- Window: A subset of the fragments needed to carry a packet (see section 5)

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 are out of the scope of this document.

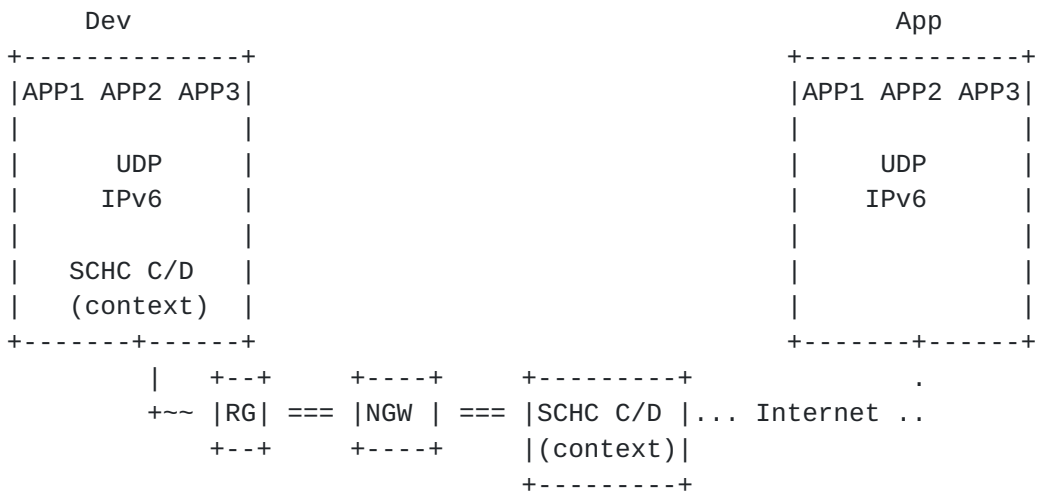


Figure 2: Architecture

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 a 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).

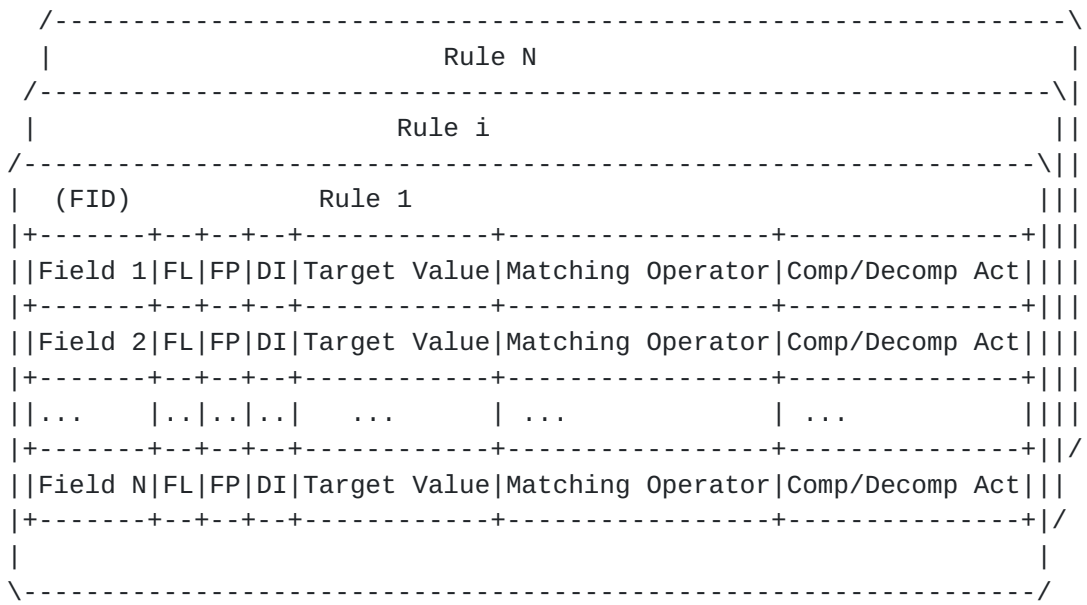


Figure 3: Compression/Decompression Context

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 should 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, etc.). For instance, it can be a single value or a more complex structure (array, list, etc.), 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 in the NGW side the SCHC C/D needs to find the correct

Rule to be used by identifying its Dev-ID and the Rule-ID. In the Dev, only the Rule-ID may be 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.

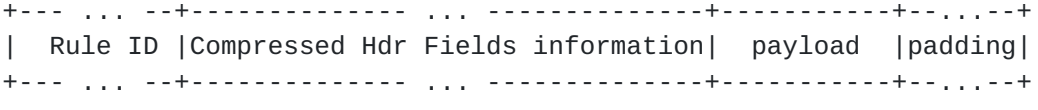


Figure 4: LPWAN Compressed Format Packet

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

The 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 after applying SCHC header compression or when SCHC header compression is not possible the entire IPv6 datagram 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, (e.g.) 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(s) will be used over a specific LPWAN technology.

An important consideration is that LPWAN networks typically follow a 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. Functionalities

This subsection describes the different fields in the fragmentation header frames (see the related formats in Section 5.4), as well as the tools that are used to enable the fragmentation functionalities defined in this document, and the different reliability options supported.

- Rule ID. The Rule ID is present in the fragment header and in the ACK header format. The Rule ID in a fragment header is used to identify that a fragment is being carried, the fragmentation delivery reliability option used and it may indicate the window

size in use (if any). The Rule ID in the fragmentation header also allows to interleave non-fragmented IPv6 datagrams with fragments that carry a larger IPv6 datagram. The Rule ID in an ACK allows to identify that the message is an ACK.

- 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 FCN reserved values that are used for controlling the fragmentation process, as described next. The FCN value with all the bits equal to 1 (All-1) denotes the last fragment of a packet. And the FCN value with all the bits equal to 0 (All-0) denotes the last fragment of a window (when such window is not the last one of the packet) in any window mode or the fragments in No ACK mode. The rest of the FCN values are assigned 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 reliability options, it is recommended to use a number of bits (N) equal to or greater than 3. Nevertheless, the apropriate value will be defined in the corresponding technology documents. 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.
  For windows that are not the last one from a fragmented packet, the FCN for the last fragment in such windows is an All-0. This indicates that the window is finished and communication proceeds according to the reliability option in use. The FCN for the last fragment in the last window 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.

- Datagram Tag (DTag). The DTag field, if present, is set to the same value for all fragments carrying the same IPv6 datagram. This field 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.

- 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 also 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.

- Message Integrity Check (MIC). This field, which has a size of M bits, is computed by the sender over the complete packet (i.e. a SCHC compressed or an uncompressed IPv6 packet) before fragmentation. 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. The CRC32 as 0xEDB88320 is recommended as the default algorithm for computing the MIC. Nevertheless, other algorithm MAY be mandated in the corresponding technology documents (e.g. technology-specific profiles).

- C (MIC checked): C is a 1-bit field. This field is used in the ACK format packets to report the outcome of the MIC check, i.e. whether the reassembled packet was correctly received or not.

- Retransmission Timer. It is used by a fragment sender after the transmission of a window to detect a transmission error of the ACK corresponding to this window. Depending on the reliability option, it will lead to a request for an ACK retransmission on ACK-Always or it will trigger the next window on ACK-on-error. The dureation of this timer is not defined in this document and must be defined in the corresponding technology documents (e.g. technology-specific profiles).

- Inactivity Timer. This timer is used by a fragment receiver to detect when there is a problem in the transmission of fragments and the receiver does not get any fragment during a period of time or a number of packets in a period of time. When this happens, an Abort message needs to be sent. Initially, and each time a fragment is received the timer is reinitialized. The duration of this timer is not defined in this document and must be defined in the specific technology document (e.g. technology-specific profiles).

- Attempts. It is a counter used to request a missing ACK, and in consequence to determine when an Abort is needed, because there are recurrent fragment transmission errors, whose maximum value 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). The Attempts counter is defined per window, it will be initialized each time a new window is used.

- Bitmap. The Bitmap is a sequence of bits carried in an ACK for a given window. Each bit in the Bitmap corresponds to a fragment of the current window, and provides feedback on whether the fragment has been received or not. The right-most position on the Bitmap is used to report whether the All-0 or All-1 fragments have been received or not. Feedback for a fragment with the highest FCN value is provided by the left-most position in the Bitmap. In the Bitmap, a bit set to 1 indicates that the corresponding FCN fragment has been correctly sent and received. However, the sending format of the Bitmap will be truncated until a byte boundary where the last error is given. However, when all the Bitmap is transmitted, it may be truncated, see more details in Section 5.5.3

- Abort. In case of error or when the Inactivity timer expires or MAX_ACK_REQUESTS is reached the sender or the receiver may use the Abort frames. When the receiver needs to abort the on-going fragmented packet transmission, it uses the ACK Abort format packet with all the bits set to 1. When the sender needs to abort the transmission it will use the All-1 Abort format, this fragment is not acked.

- Padding (P). Padding will be used to align the last byte of a fragment with a byte boundary. The number of bits used for padding is not defined and depends on the size of the Rule ID, DTag and FCN fields, and on the layer two payload size.

5.3. Delivery Reliability options

This specification defines the following three fragment delivery reliability options:

- 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.

- Window mode - ACK always (ACK-Always).
  The ACK-always option provides flow control. In addition, this option 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 has 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 sends an ACK request using the All-1 empty fragment.
The maximum number of ACK requests is MAX_ACK_REQUESTS.

- Window mode - ACK-on-error (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.
  In ACK-on-error, an ACK is transmitted by the fragment receiver after a window of fragments have been sent, only if at least one of the fragments in the window has been lost. In this mode, the ACK informs the sender about received and/or missed fragments from the window of fragments. Upon receipt of an ACK that informs about any lost fragments, the sender retransmits the lost fragments. 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 on-going fragmented packet transmission. One exception to this behavior is in the last window, where the receiver MUST transmit an ACK, even if all the fragments in the last window have been correctly received.

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 B.

5.4. Fragmentation Frame Formats

This section defines the fragment format, the All-0 and All-1 frame formats, the ACK format and the Abort frame formats.

5.4.1. Fragment format

A fragment comprises a fragment header, a fragment payload, and Padding bits (if any). A fragment 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 data payload. A fragment is the payload in the L2 protocol data unit (PDU).

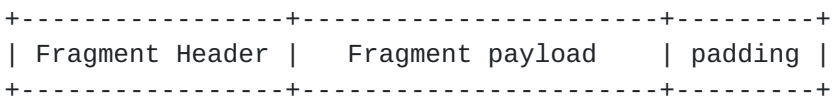


Figure 6: Fragment format.

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

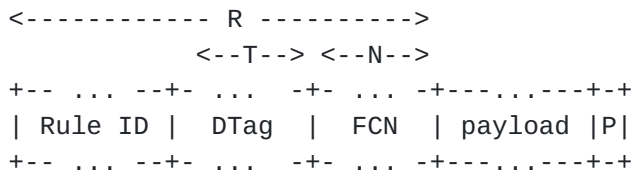


Figure 7: Fragment Format for Fragments except the Last One, No ACK option

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

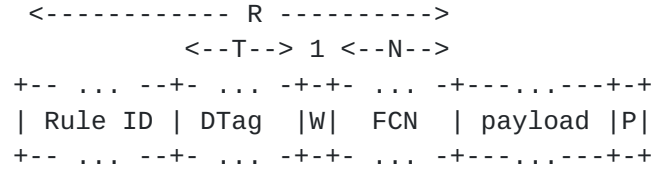


Figure 8: Fragment Format for Fragments except the Last One, Window mode

5.4.2. ACK format

The format of an ACK that acknowledges a window that is not the last one (denoted as ALL-0 window) is shown in Figure 9.

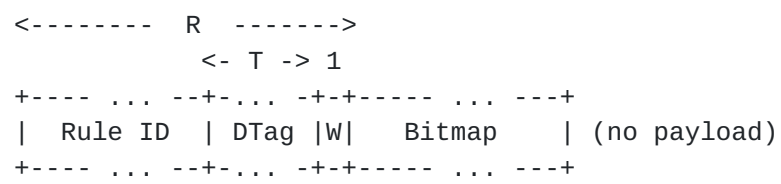


Figure 9: ACK format for All-0 windows

To acknowledge the last window of a packet (denoted as All-1 window), a C bit (i.e. MIC checked) following the W bit is set to 1 to indicate that the MIC check computed by the receiver matches the MIC present in the All-1 fragment. If the MIC check fails, the C bit is set to 0 and the Bitmap for the All-1 window follows.

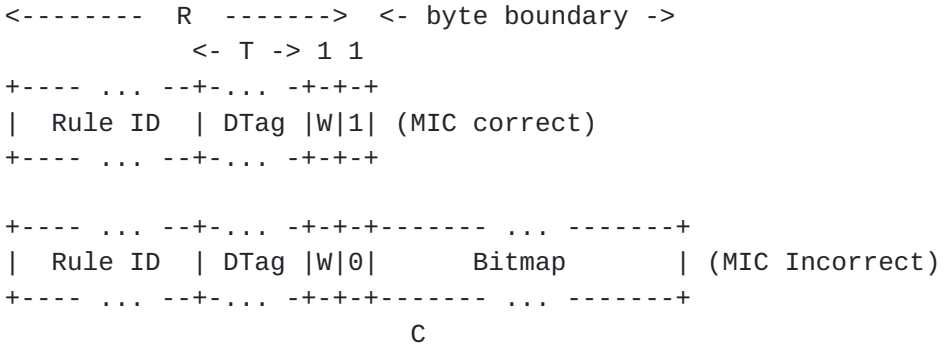


Figure 10: Format of an ACK for All-1 windows

5.4.3. All-1 and All-0 formats

The All-0 format is used for the last fragment of a window that is not the last window of the packet.

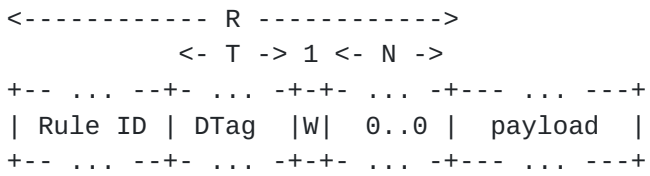


Figure 11: All-0 fragment format

The All-0 empty fragment format is used by a sender to request an ACK in ACK-Always mode

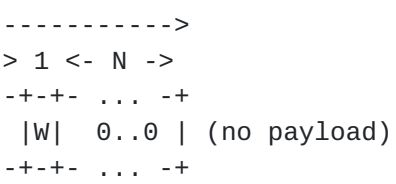


Figure 12: All-0 empty fragment format

In the No ACK option, the last fragment of an IPv6 datagram SHALL contain a fragment header that conforms to the format shown in Figure 13. The total size of this fragment header is R+M bits.

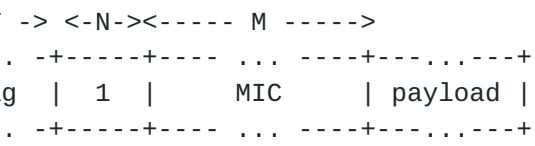


Figure 13: All-1 Fragment Format for the Last Fragment, No ACK option

In any of the Window modes, the last fragment of an IPv6 datagram SHALL contain a fragment header that conforms to the format shown in Figure 14. The total size of the fragment header in this format is R+M bits.

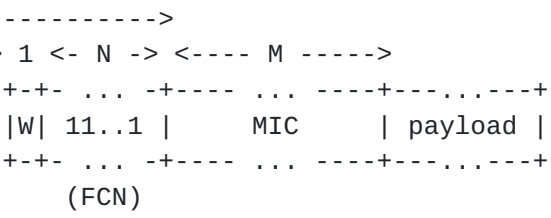


Figure 14: All-1 Fragment Format for the Last Fragment, Window mode

In either ACK-Always or ACK-on-error, in order to request a retransmission of the ACK for the All-1 window, the fragment sender uses the format shown in Figure 15. The total size of the fragment header in this format is R+M bits.

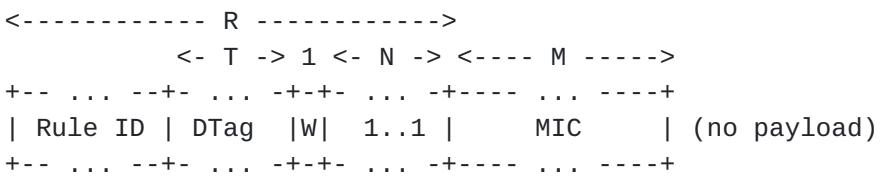


Figure 15: All-1 for Retries format, also called All-1 empty

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).

5.4.4. Abort formats

The All-1 Abort and the ACK abort messages have the following formats.

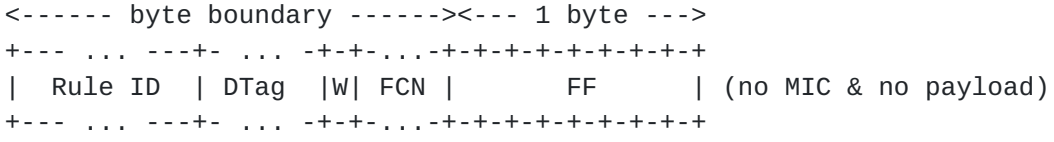


Figure 16: All-1 Abort format

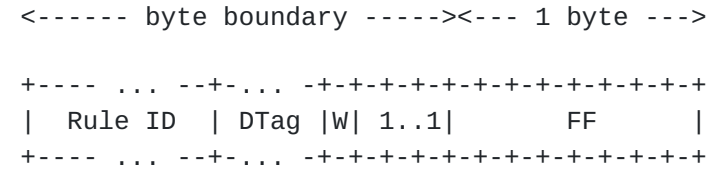


Figure 17: ACK Abort format

5.5. Baseline mechanism

The fragment 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.

Fragmentation functionality uses the FCN value, which has a length of N bits. The All-1 and All-0 FCN values are used to control the fragmentation transmission. The FCN will be assigned sequentially in a decreasing order starting from 2^N-2, i.e. the highest possible FCN value depending on the FCN number of bits, but excluding the All-1 value. In all modes, the last fragment of a packet must contains a MIC which is used to check if there are errors or missing fragments, and must use the corresponding All-1 fragment format. Also note that, a fragment with an All-0 format is considered the last fragment of the current 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.

5.5.1. No ACK

In the No ACK mode there is no feedback communication from the fragment receiver. The sender will send the fragments of a packet until the last one without any possibility to know if errors or a losses have occurred. As in this mode there is not a need to identify specific fragments a one-bit FCN is used, therefore FCN All-0 will be used in all fragments except the last one. The latter will carry an All-1 FCN and will also carry the MIC. The receiver

will wait for fragments and will set the Inactivity timer. The No ACK mode will use the MIC contained in the last fragment to check error. When the Inactivity Timer expires or when the MIC check indicates that the reassembled packet does not match the original one, the receiver will release all resources allocated to reassembly of the packet. The initial value of the Inactivity Timer will be determined based on the characteristics of the underlying LPWAN technology and will be defined in other documents (e.g. technology-specific profile documents).

5.5.2. The Window modes

In Window modes, a jumping window protocol uses two windows alternatively, identified as 0 and 1. A fragment with all FCN bits set to 0 (i.e. an All-0 fragment) indicates that the window is over (i.e. the fragment is the last one of the window) and allows to switch from one window to the next one. The All-1 FCN in a fragment indicates that it is the last fragment of the packet being transmitted and therefore there will not be another window for the packet.

The Window mode offers two different reliability option modes: ACK-on-error and ACK-always.

5.5.2.1. ACK-Always

In ACK-Always, the sender sends fragments by using the two-jumping window procedure. 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 by one. When the FCN reaches value 0 and there are more fragments to be sent, an All-0 fragment is sent and the Retransmission Timer is set. The sender waits for an ACK to know if transmission errors have occurred. Then, the receiver sends an ACK reporting whether any fragments have been lost or not by setting the corresponding bits in the Bitmap, otherwise, an ACK without Bitmap will be sent, allowing transmission of a new window. When the last fragment of the packet is sent, an All-1 fragment (which includes a MIC) is used. In that case, the sender sets the Retransmission Timer to wait for the ACK corresponding to the last window. During this period, the sender starts listening to the radio and starts the Retransmission Timer, which needs to be dimensioned based on the received window available for the LPWAN technology in use. If the Retransmission Timer expires, an empty All-0 (or an empty All-1 if the last fragment has been sent) fragment is sent to ask the receiver to resend its ACK. The window number is not changed.

When the sender receives an ACK, it checks the W bit carried by the ACK. Any ACK carrying an unexpected W bit is discarded. If the W bit value of the received ACK is correct, the sender analyzes the received Bitmap. If all the fragments sent during the window have been well received, and if at least one more fragment needs to be sent, the sender moves its sending window to the next window value and sends the next fragments. If no more fragments have to be sent, then the fragmented packet transmission is finished.

However, if one or more fragments have not been received as per the ACK (i.e. the corresponding bits are not set in the Bitmap) then the sender resends the missing fragments. When all missing fragments have been retransmitted, the sender starts the Retransmission Timer (even if an All-0 or an All-1 has not been sent during the retransmission) and waits for an ACK. Upon receipt of the ACK, if one or more fragments have not yet been received, the counter Attempts is increased and the sender resends the missing fragments again. When Attempts reaches MAX_ACK_REQUESTS, the sender aborts the on-going fragmented packet transmission by sending an Abort message and releases any resources for transmission of the packet. The sender also aborts an on-going fragmented packet transmission when a failed MIC check is reported by the receiver.

On the other hand, at the beginning, the receiver side expects to receive window 0. Any fragment received but not belonging to the current window is discarded. All fragments belonging to the correct window are accepted, and the actual fragment number managed by the receiver is computed based on the FCN value. The receiver prepares the Bitmap to report the correctly received and the missing fragments for the current window. After each fragment is received the receiver initializes the Inactivity timer, if the Inactivity Timer expires the transmission is aborted.

When an All-0 fragment is received, it indicates that all the fragments have been sent in the current window. Since the sender is not obliged to always send a full window, some fragment number not set in the receiver memory may not correspond to losses. The receiver sends the corresponding ACK, the Inactivity Timer is set and the transmission of the next window by the sender can start.

If an All-0 fragment has been received and all fragments of the current window have also been received, the receiver then expects a new Window and waits for the next fragment. Upon receipt of a fragment, if the window value has not changed, the received fragments are part of a retransmission. A receiver that has already received a fragment should discard it, otherwise, it updates the Bitmap. If all the bits of the Bitmap are set to one, the receiver may send an ACK

without waiting for an All-0 fragment and the Inactivity Timer is initialized.

On the other hand, if the window value of the next received fragment is set to the next expected window value, this means that the sender has received a correct Bitmap reporting that all fragments have been received. The receiver then updates the value of the next expected window.

If the receiver receives an All-0 fragment, the sender may send one or more fragments per window. Otherwise, some fragments in the window have been lost.

When an All-1 fragment is received, it indicates that the last fragment of the packet has been sent. Since the last window is not always full, the MIC will be used to detect if all fragments of the packet have been received. A correct MIC indicates the end of the transmission but the receiver must stay alive for an Inactivity Timer period to answer to any empty All-1 fragments the sender may send if ACKs sent by the receiver are lost. If the MIC is incorrect, some fragments have been lost. The receiver sends the ACK regardless of successful fragmented packet reception or not, the Inactitivity Timer is set. In case of an incorrect MIC, the receiver waits for fragments belonging to the same window. After MAX_ACK_REQUESTS, the receiver will abort the on-going fragmented packet transmission. The receiver also Aborts upon Inactivity Timer expiration.

5.5.2.2. ACK-on-error

The ACK-on-error sender is similar to ACK-Always, the main difference being that in ACK-on-error the ACK is not sent at the end of each window but only when at least one fragment of the current window has been lost (with the exception of the last window, see next paragraph). In Ack-on-error, the Retransmission Timer expiration will be considered as a positive acknowledgment. The Retransmission Timer is set when sending an All-0 or an All-1 fragment. When the All-1 fragment has been sent, then the on-going fragmented packet transmission fragmentation is finished and the sender waits for the last ACK. At the receiver side, when the All-1 fragment is sent and the MIC check indicates successful packet reception, an ACK is also sent to confirm the end of a correct transmission. If the Retransmission Timer expires, an All-1 empty request for the last ACK MUST be sent by the sender to complete the fragmented packet transmission.

If the sender receives an ACK, it checks the window value. ACKs with an unexpected window number are discarded. If the window number on the received Bitmap is correct, the sender verifies if the receiver

has received all fragments of the current window. When at least one fragment has been lost, the counter Attempts is increased by one and the sender resends the missing fragments again. When Attempts reaches MAX_ACK_REQUESTS, the sender sends an Abort message and releases all resources for the on-going fragmented packet transmission. When the retransmission of missing fragments is finished, the sender starts listening for an ACK (even if an All-0 or an All-1 has not been sent during the retransmission) and initializes and starts the Retransmission Timer. After sending an All-1 fragment, the sender listens for an ACK, initializes Attempts, and initializes and starts the Retransmission Timer. If the Retransmission Timer expires, Attempts is increased by one and an empty All-1 fragment is sent to request the ACK for the last window. If Attempts reaches MAX_ACK_REQUESTS, the on-going fragmented packet transmission is aborted.

Unlike the sender, the receiver for ACK-on-error has a larger amount of differences compared with ACK-Always. First, an ACK is not sent unless there is a lost fragment or an unexpected behavior (with the exception of the last window, where an ACK is always sent regardless of fragment losses or not). The receiver starts by expecting fragments from window 0 and maintains the information regarding which fragments it receives. After receiving a fragment, the Inactivity Timer is set, if no fragment has been received and the Inactivity Timer expires the transmission is aborted.

Any fragment not belonging to the current window is discarded. The actual fragment number is computed based on the FCN value. When an All-0 fragment is received and all fragments have been received, the receiver updates the expected window value.

If an All-0 fragment is received, even if another fragment is missing, all fragments from the current window have been sent. Since the sender is not obligated to send a full window, a fragment number not used may not necessarily correspond to losses. As the receiver does not know if the missing fragments are lost or not, it sends an ACK and reinitialises the Inactivity Timer.

On the other hand, after receiving an All-0 fragment, the receiver expects a new window and waits for the next fragment.
If the window value of the next fragment has not changed, the received fragment is a retransmission. A receiver that has already received a fragment should discard it. If all fragments of a window (that is not the last one) have been received, the receiver does not send an ACK. While the receiver waits for the next window and if the window value is set to the next value, and if an All-1 fragment with the next value window arrived the receiver aborts the on-going

fragmented packet transmission, and it drops the fragments of the aborted packet transmission.

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. Regardless of fragment losses, the receiver sends an ACK and initializes the Inactivity Timer.

Reception of an All-1 fragment indicates the last fragment of the packet has been sent. Since the last window is not always full, the MIC will be used to detect if all fragments of the window have been received. A correct MIC check indicates the end of the fragmented packet transmission. An ACK is sent by the fragment receiver. In case of an incorrect MIC, the receiver waits for fragments belonging to the same window or the expiration of the Inactivity Timer. The latter will lead the receiver to abort the on-going fragmented packet transmission.

5.5.3. Bitmap Optimization

The Bitmap is transmitted by a receiver as part of the ACK format when there are some missing fragments in a window. An ACK message may introduce padding at the end to align transmitted data to a byte boundary. The first byte boundary includes one or more complete bytes, depending on the size of Rule ID and DTag.

Note that the ACK sent in response to an All-1 fragment includes the C bit. Therefore, the window size and thus the Bitmap size need to be determined taking into account the available space in the layer two frame payload, where there will be 1 bit less for an ACK sent in response to an All-1 fragment than in other ACKs.

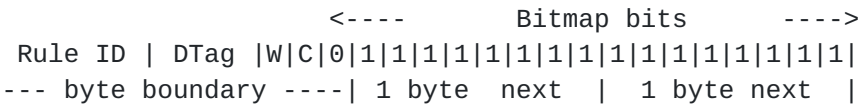


Figure 18: Bitmap

The Bitmap, when transmitted, MUST be optimized in size to reduce the resulting frame size. The right-most bytes with all Bitmap bits set to 1 MUST NOT be transmitted. As the receiver knows the Bitmap size, it can reconstruct the original Bitmap without this optimization. In the example Figure 19, the last 2 bytes of the Bitmap shown in Figure 18 comprise all bits set to 1, therefore, the last 2 bytes of the Bitmap are not sent.

In the last window, when checked bit C value is 1, it means that the received MIC matches the one computed by the receiver, and thus the Bitmap is not sent. Otherwise, the Bitmap needs to be sent after the C bit. Note that the introduction of a C bit may force to reduce the number of fragments in a window to allow the bitmap to fit in a frame.

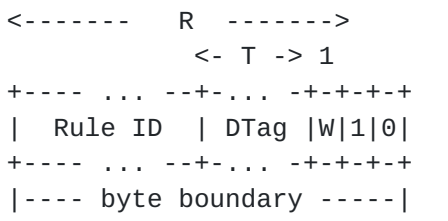


Figure 19: Bitmap transmitted fragment format

Figure 20 shows an example of an ACK (for N=3), where the Bitmap indicates that the second and the fifth fragments have not been correctly received.

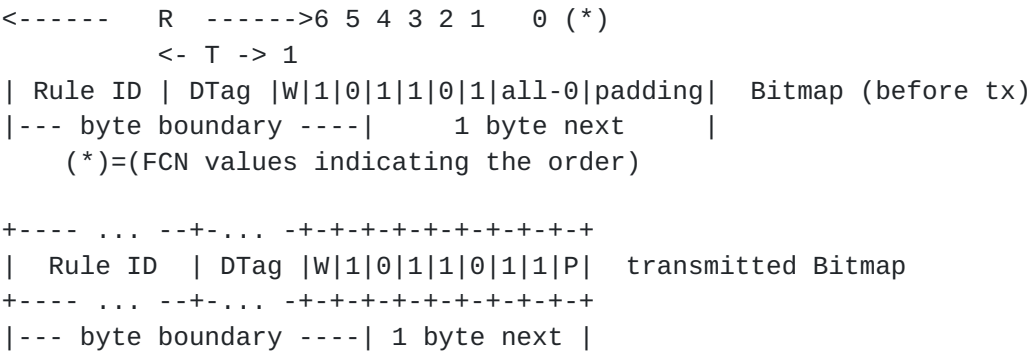


Figure 20: Example of a Bitmap before transmission, and the transmitted one, in any window except the last one, for N=3

Figure 21 shows an example of an ACK (for N=3), where the Bitmap indicates that the MIC check has failed but there are no missing fragments.

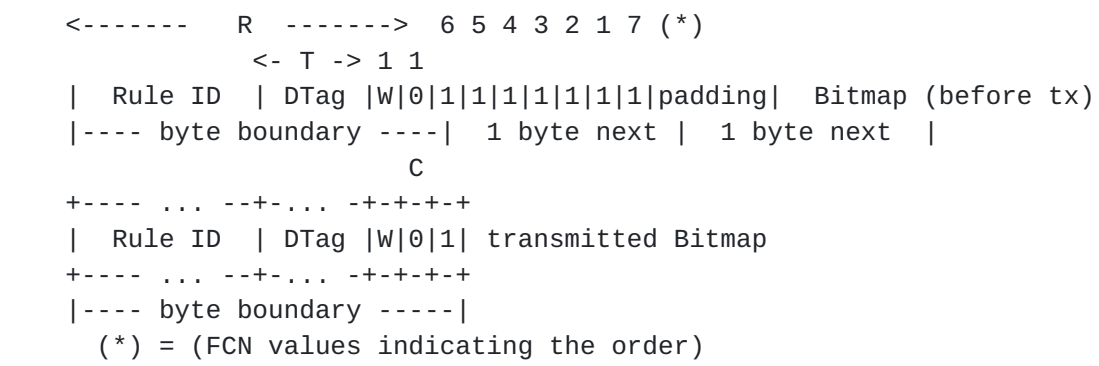


Figure 21: Example of the Bitmap in Window mode for the last window, for N=3)

5.6. Supporting multiple window sizes

For ACK-Always or ACK-on-error, implementers may opt to support a single window size or multiple window sizes. The latter, when feasible, may provide performance optimizations. For example, a large window size 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 a 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.

Note that the same window size MUST be used for the transmission of all fragments that belong to a packet.

5.7. 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.

For fragmented packet transmission in the downlink, and when ACK Always is used, the fragment receiver MAY support timer-based ACK retransmission. In this mechanism, the fragment receiver initializes and starts a timer (the Inactivity Timer is used) after the transmission of an ACK, except when the ACK is sent in response to

the last fragment of a packet (All-1 fragment). In the latter case, the fragment receiver does not start a timer after transmission of the ACK.

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

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

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

6. Padding management

SCHC header, either for compression, fragmentation or acknowledgment does not preserve byte alignment. Since most of the LPWAN network technologies payload is expressed in an integer number of bytes; the sender will introduce at the end some padding bits while the receiver must be able to eliminate them.

The algorithm for padding bit elimination for compressed or fragmented frames is simple. Based on the following principle: * The SCHC header is not aligned on a byte boundary, but its size in bits is given by the rule.

- The data size is variable, but always a multiple of 8 bits.

- Padding bits MUST never exceed 7 bits.

In that case, a receiver after decoding the SCHC header, must take the maximum multiple of 8 bits as data. The remaining bits are padding bits.

7. SCHC Compression for IPv6 and UDP headers

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

7.1. IPv6 version field

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

7.2. IPv6 Traffic class field

If the DiffServ 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 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. In 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

7.3. 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. In the second, 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

7.4. 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.

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

7.5. 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.

7.6. 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.

7.7. IPv6 addresses fields

As in 6LoWPAN [RFC4944], IPv6 addresses are splitted 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.

7.7.1. IPv6 source and destination prefixes

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

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".

7.7.2. 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".

7.8. IPv6 extensions

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

7.9. 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 these 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".

7.10. 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".

7.11. 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 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".

8. Security considerations

8.1. 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.

8.2. 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 splitted 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 the 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.

9. Acknowledgements

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

10. References

10.1. 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>.

10.2. 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 22 presents the protocol stack for this Device. IPv6 and UDP are represented with dotted lines since these protocols are compressed on the radio link.

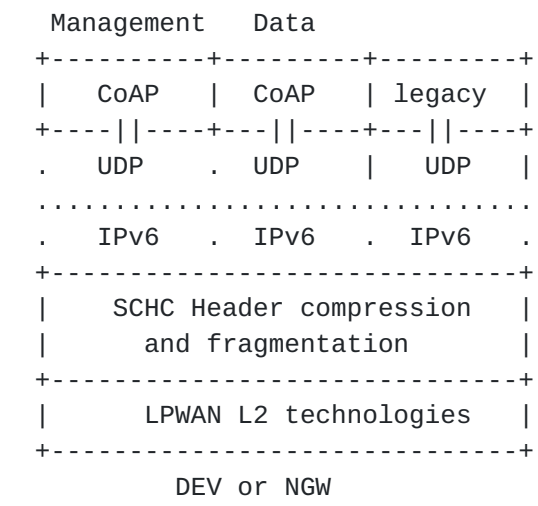


Figure 22: Simplified Protocol Stack for LP-WAN

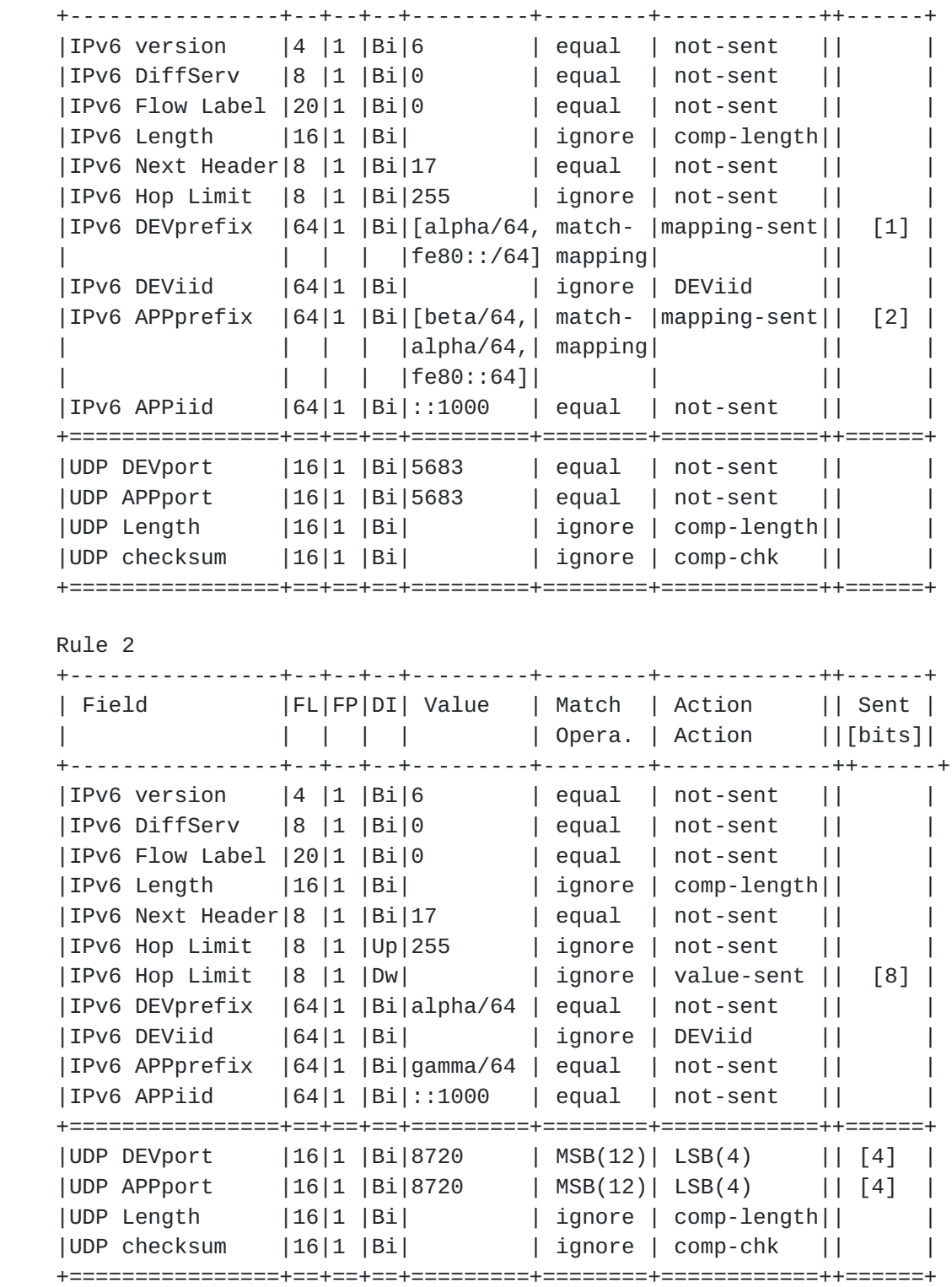
Note that in some LPWAN technologies, only the Devs have a device ID. Therefore, when such technologies are used, it is necessary to define statically an IID for the Link Local address for the SCHC C/D.

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, fe80::/64]	match-mapping	mapping-sent		[1]
IPv6 DEViid	64	1	Bi		ignore	DEViid		
IPv6 APPprefix	64	1	Bi	[beta/64, alpha/64, fe80::64]	match-mapping	mapping-sent		[2]
IPv6 APPiid	64	1	Bi	::1000	equal	not-sent		
UDP DEVport	16	1	Bi	5683	equal	not-sent		
UDP APPport	16	1	Bi	5683	equal	not-sent		
UDP Length	16	1	Bi		ignore	comp-length		
UDP checksum	16	1	Bi		ignore	comp-chk		

Rule 2

Field	FL	FP	DI	Value	Match Opera.	Action	Sent [bits]	
IPv6 version	4	1	Bi	6	equal	not-sent		
IPv6 DiffServ	8	1	Bi	0	equal	not-sent		
IPv6 Flow Label	20	1	Bi	0	equal	not-sent		
IPv6 Length	16	1	Bi		ignore	comp-length		
IPv6 Next Header	8	1	Bi	17	equal	not-sent		
IPv6 Hop Limit	8	1	Up	255	ignore	not-sent		
IPv6 Hop Limit	8	1	Dw		ignore	value-sent		[8]
IPv6 DEVprefix	64	1	Bi	alpha/64	equal	not-sent		
IPv6 DEViid	64	1	Bi		ignore	DEViid		
IPv6 APPprefix	64	1	Bi	gamma/64	equal	not-sent		
IPv6 APPiid	64	1	Bi	::1000	equal	not-sent		
UDP DEVport	16	1	Bi	8720	MSB(12)	LSB(4)		[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 23: Context rules

All the fields described in the three rules depicted on Figure 23 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 24 illustrates the transmission of an IPv6 packet that needs 11 fragments in the No ACK option. Where FCN is always 1 bit.

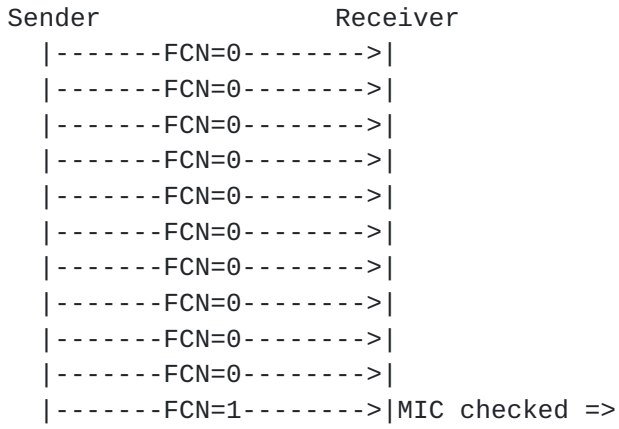


Figure 24: Transmission of an IPv6 packet carried by 11 fragments in the No ACK option

Figure 25 illustrates the transmission of an IPv6 packet that needs 11 fragments in ACK-on-error, for N=3, without losses.

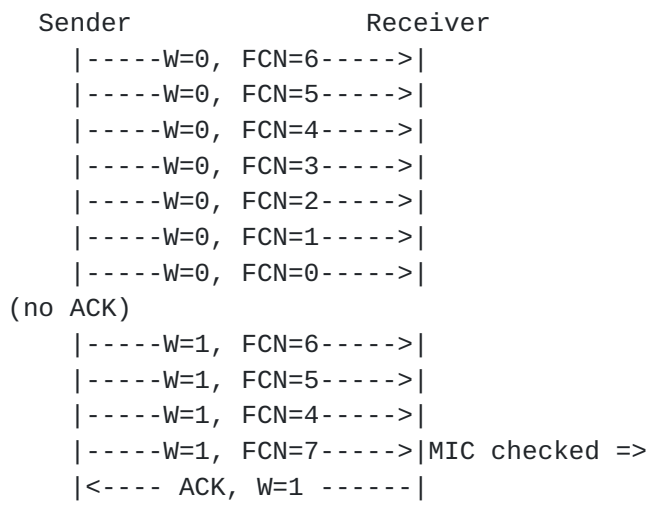


Figure 25: Transmission of an IPv6 packet carried by 11 fragments in ACK-on-error, for N=3 and MAX_WIND_FCN=6, without losses.

Figure 26 illustrates the transmission of an IPv6 packet that needs 11 fragments ACK-on-error, for N=3, with three losses.

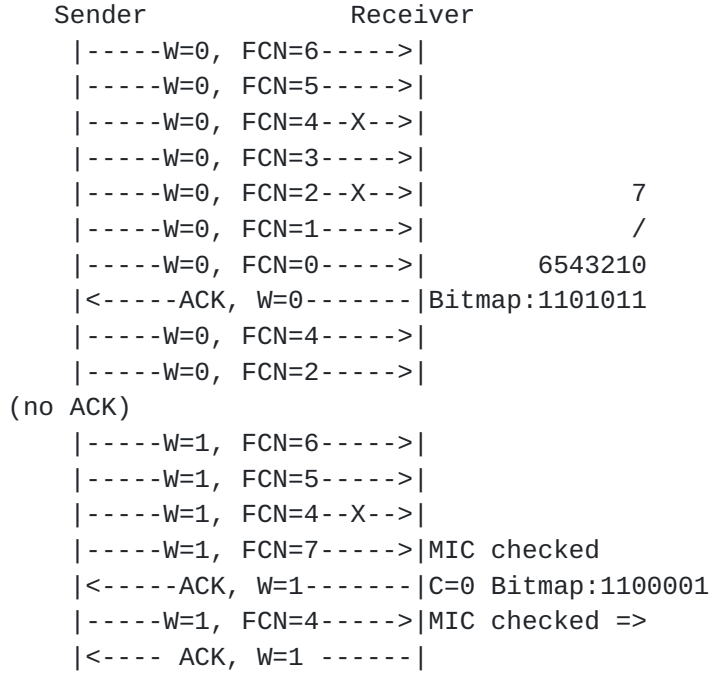


Figure 26: Transmission of an IPv6 packet carried by 11 fragments in ACK-on-error, for N=3 and MAX_WIND_FCN=6, three losses.

Figure 27 illustrates the transmission of an IPv6 packet that needs 11 fragments in 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.

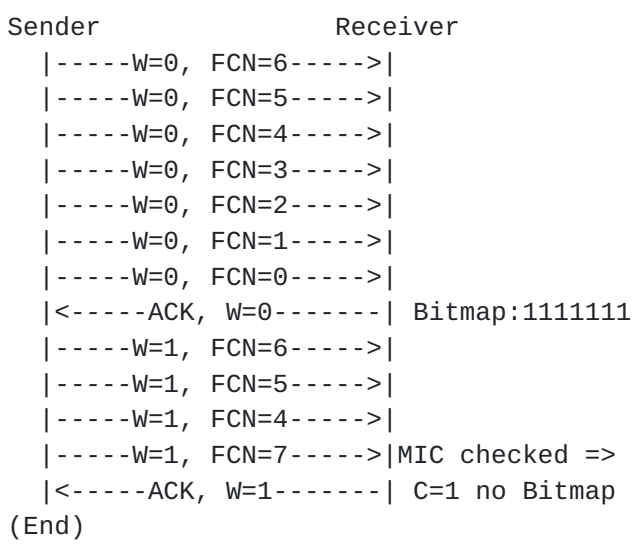


Figure 27: Transmission of an IPv6 packet carried by 11 fragments in ACK-Always, for N=3 and MAX_WIND_FCN=6, no losses.

Figure 28 illustrates the transmission of an IPv6 packet that needs 11 fragments in ACK-Always, for N=3 and MAX_WIND_FCN=6, with three losses.

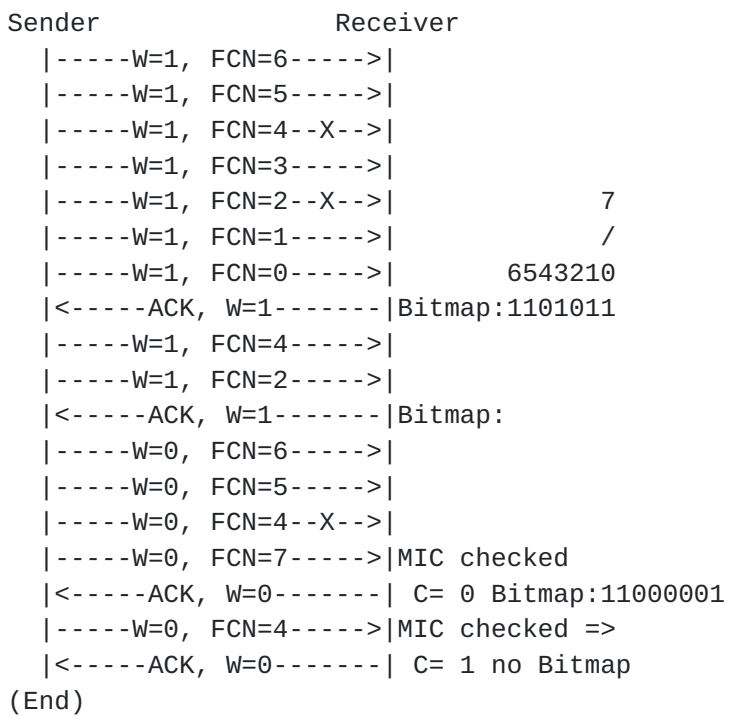


Figure 28: Transmission of an IPv6 packet carried by 11 fragments in ACK-Always, for N=3, and MAX_WIND_FCN=6, with three losses.

Figure 29 illustrates the transmission of an IPv6 packet that needs 6 fragments in 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.

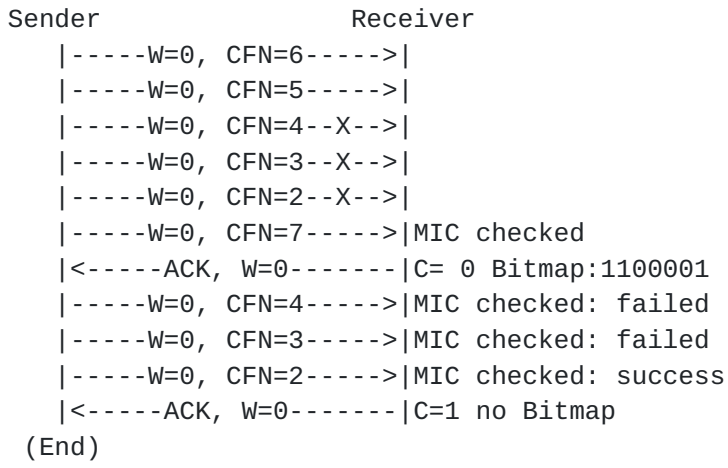


Figure 29: Transmission of an IPv6 packet carried by 11 fragments in ACK-Always, for N=3, and MAX_WIND_FCN=6, with three losses, and only one retry is needed for each lost fragment.

Figure 30 illustrates the transmission of an IPv6 packet that needs 6 fragments in 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.

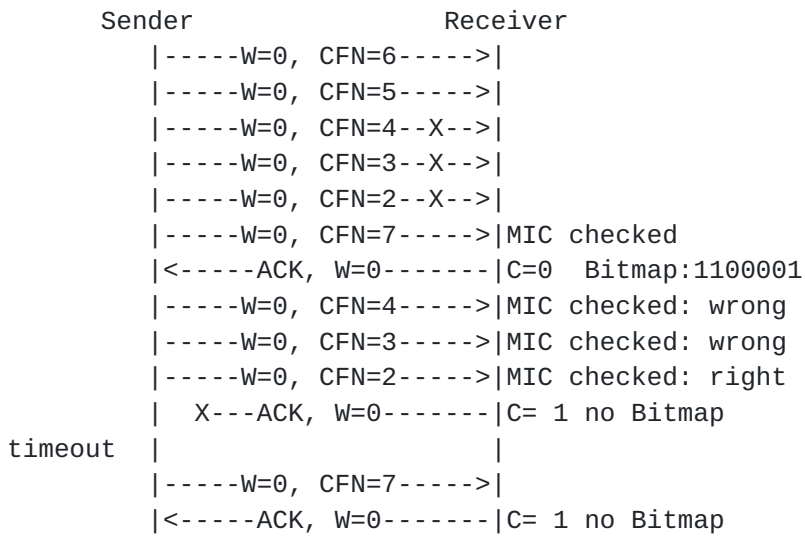


Figure 30: Transmission of an IPv6 packet carried by 11 fragments in ACK-Always, for N=3, and MAX_WIND_FCN=6, with three losses, and the second ACK is lost.

Figure 31 illustrates the transmission of an IPv6 packet that needs 6 fragments in 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.

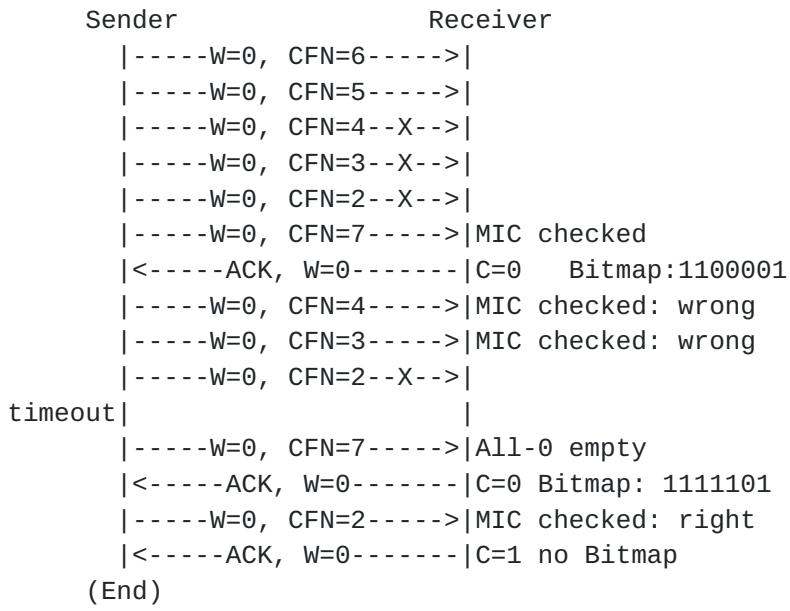
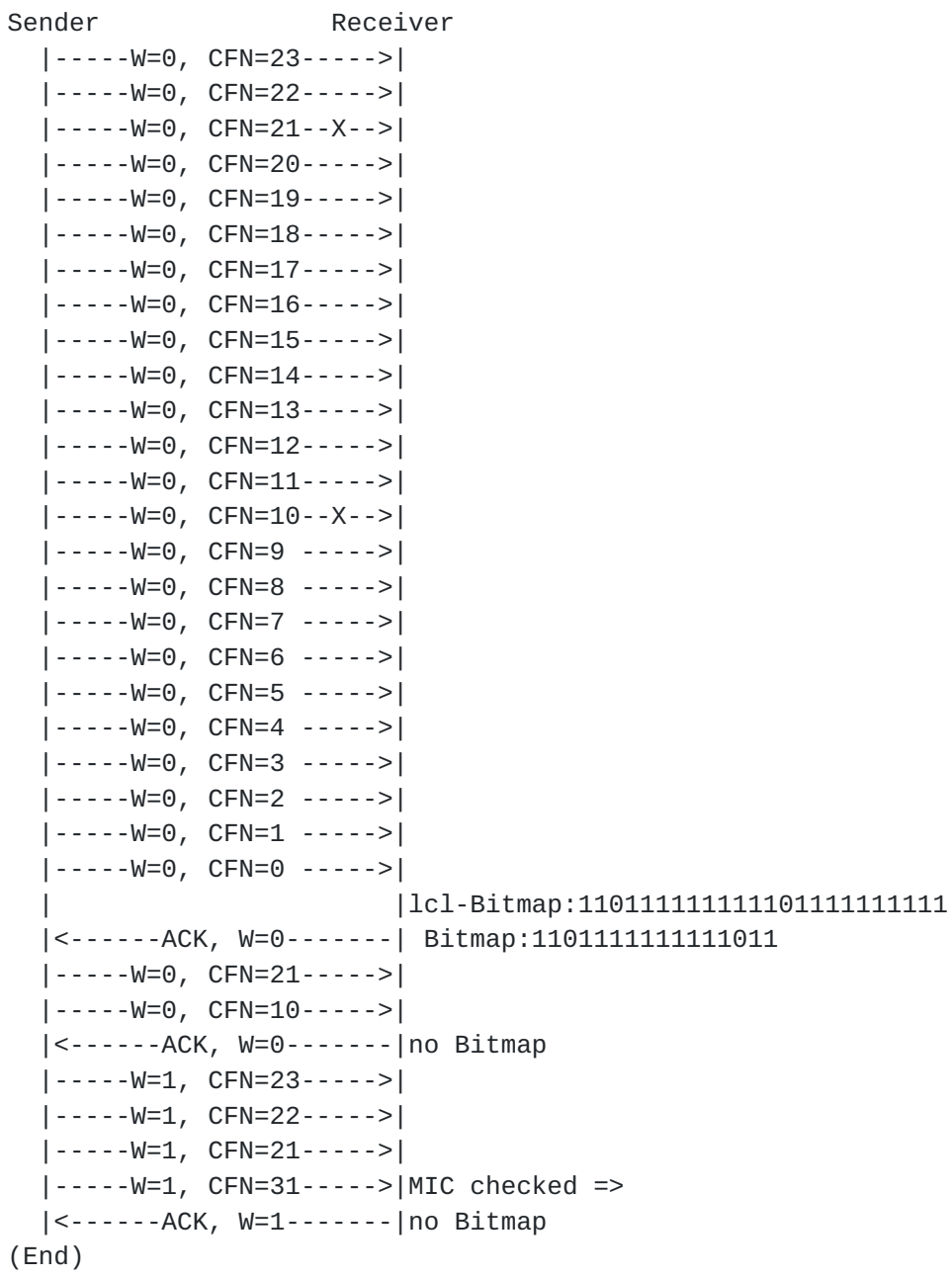


Figure 31: Transmission of an IPv6 packet carried by 11 fragments in ACK-Always, for N=3, and MAX_WIND_FCN=6, with three losses, and one retransmitted fragment is lost.

Appendix C illustrates the transmission of an IPv6 packet that needs 28 fragments in 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. Fragmentation State Machines

The fragmentation state machines of the sender and the receiver in the different reliability options are next in the following figures:

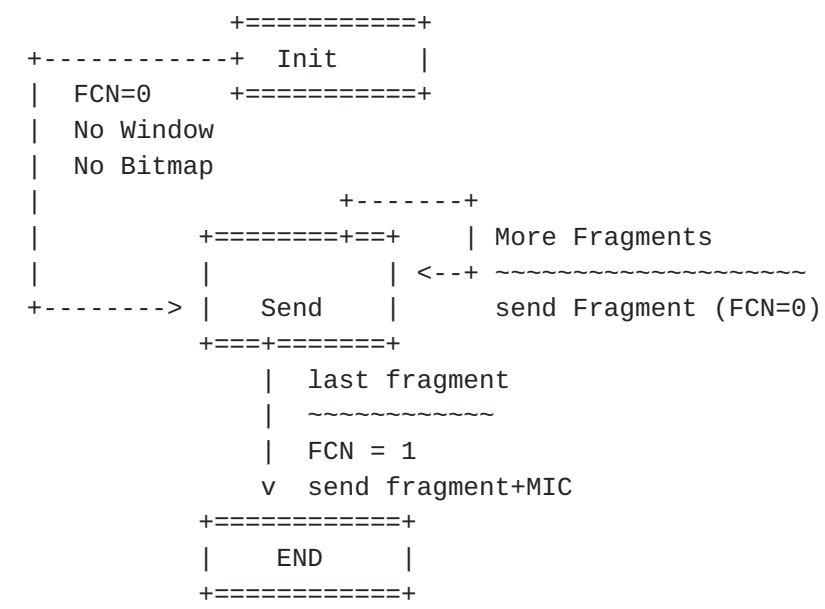


Figure 32: Sender State Machine for the No ACK Mode

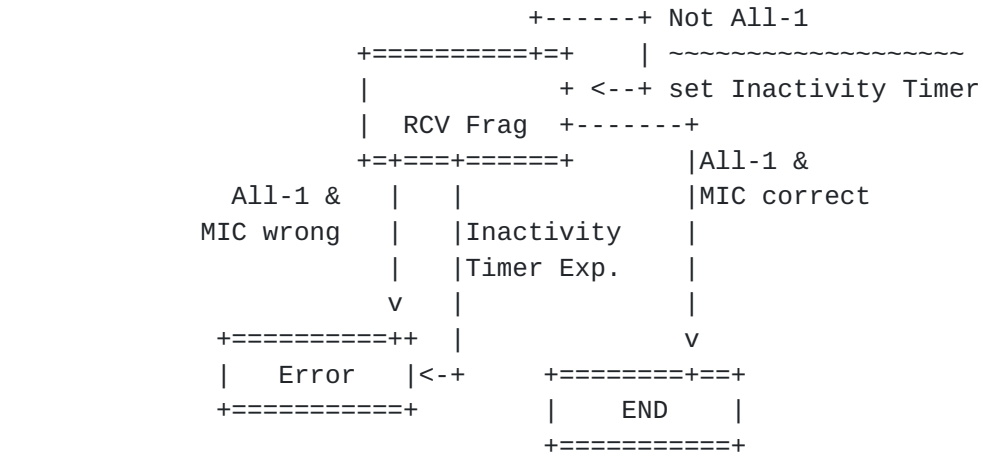


Figure 33: Receiver State Machine for the No ACK Mode

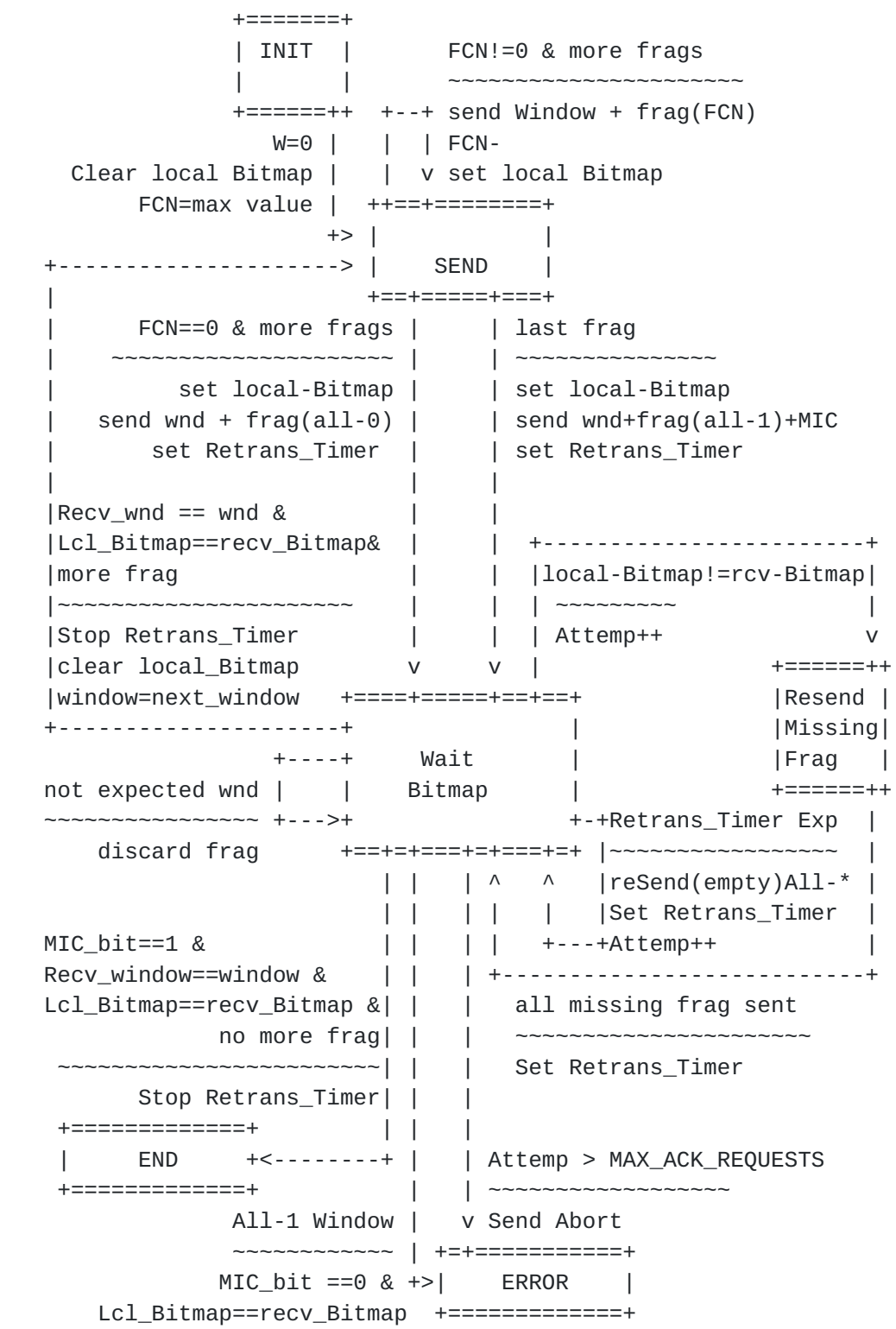
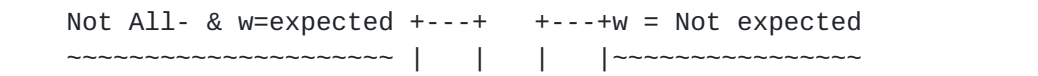


Figure 34: Sender State Machine for the ACK Always Mode




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

```
          ~~~~~~~
          Inactivity_Timer = expires
      When DWN_Link
        IF Inactivity_Timer expires
           Send DWL Request
           Attemp++
```

Figure 35: Receiver State Machine for the ACK Always Mode

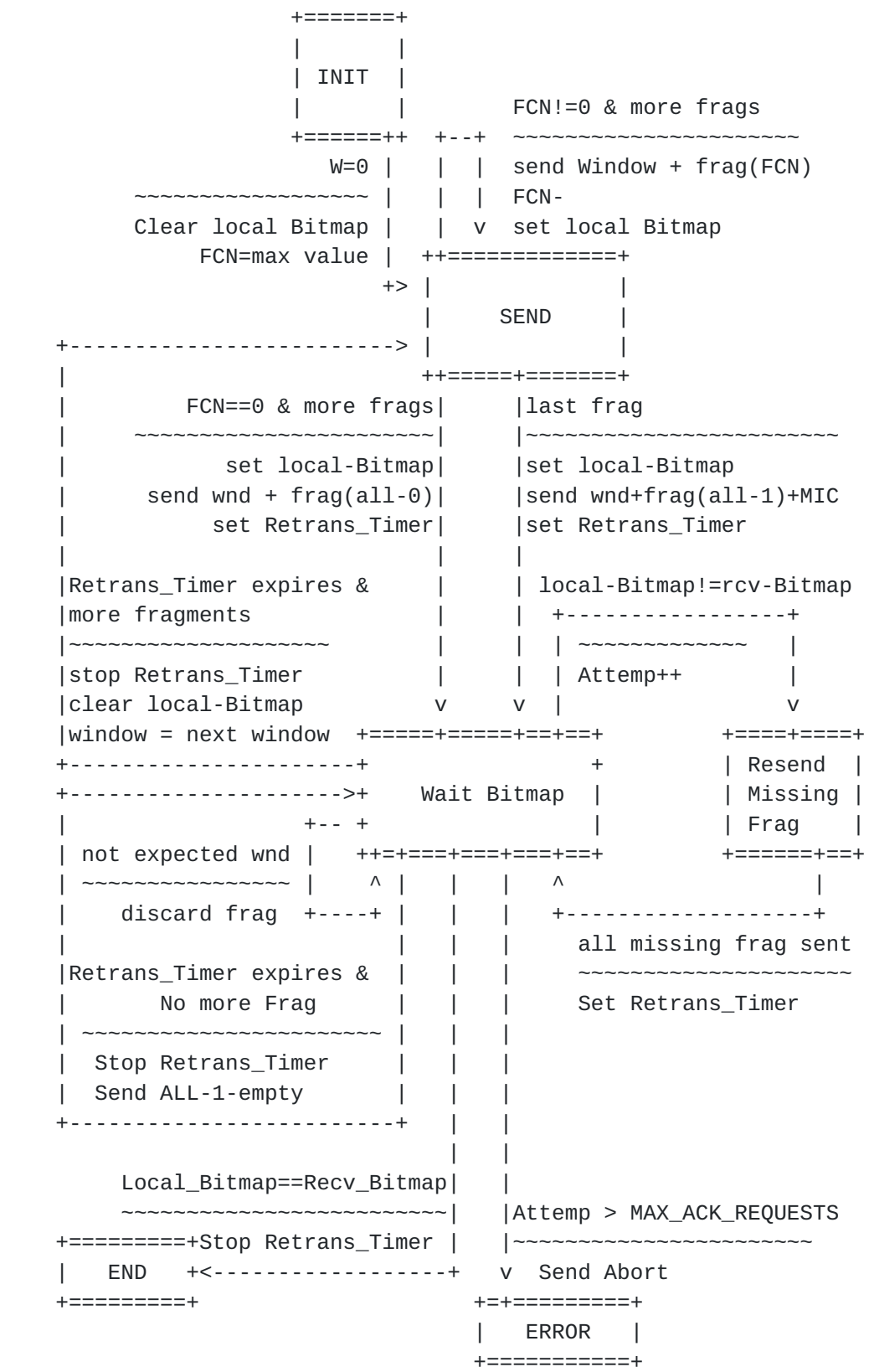


Figure 36: Sender State Machine for the ACK on error Mode

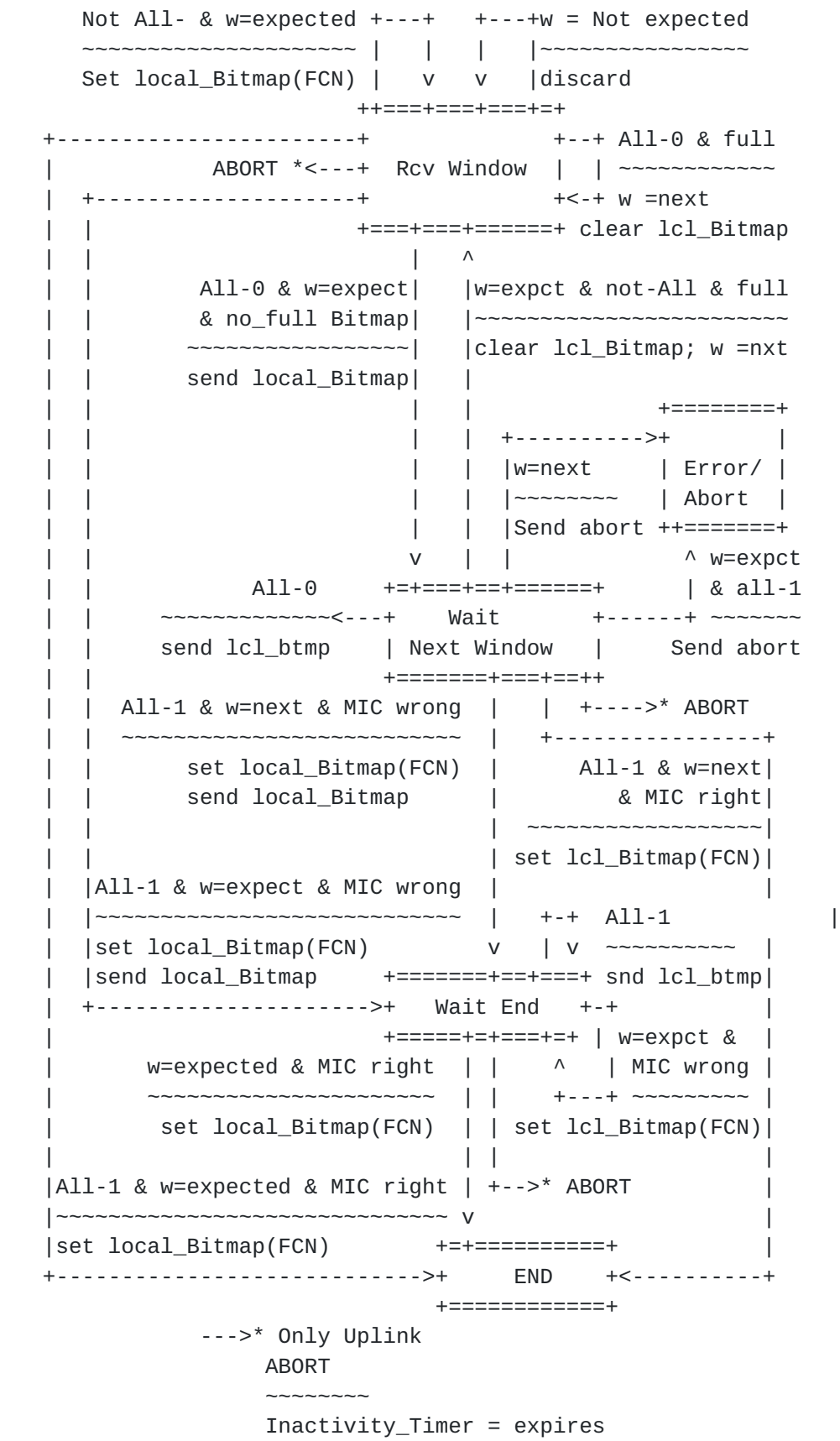

Not All- & w=expected
Set local_Bitmap(FCN)
w = Not expected
discard
ABORT
Rcv Window
All-0 & full
w =next
clear lcl_Bitmap
All-0 & w=expect
& no_full Bitmap
send local_Bitmap
w=expct & not-All & full
clear lcl_Bitmap; w =nxt
w=next
Error/
Abort
Send abort
^ w=expct
& all-1
All-0
Wait
send lcl_btmp
Next Window
Send abort
All-1 & w=next & MIC wrong
ABORT
set local_Bitmap(FCN)
send local_Bitmap
All-1 & w=next
& MIC right
set lcl_Bitmap(FCN)
All-1 & w=expect & MIC wrong
All-1
set local_Bitmap(FCN)
send local_Bitmap
snd lcl_btmp
Wait End
w=expct &
w=expected & MIC right
MIC wrong
set local_Bitmap(FCN)
set lcl_Bitmap(FCN)
All-1 & w=expected & MIC right
ABORT
set local_Bitmap(FCN)
END
---->* Only Uplink
ABORT
Inactivity_Timer = expires

Figure 37: Receiver State Machine for the ACK on error Mode

Appendix D. 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 E. 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