SCHC Working Group
Internet-Draft
Intended status: Informational
Expires: 26 July 2026

R. Munoz-Lara
Universidad de Chile
S. Cespedes
Concordia University

22 January 2026

Static Context Header Compression and Fragmentation ARQ/FEC mode draft-munoz-schc-over-dts-iot-01

Abstract

This document defines a new fragmentation mode for the SCHC standard defined in RFC8724. The new fragmentation mode is designed for communications that require additional reliability mechanisms. The reliability is based on a hybrid ARQ/FEC type II mechanism that combines forward error correction (FEC) with adaptive retransmissions to achieve high reliability.

Status of This Memo

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

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

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

This Internet-Draft will expire on 26 July 2026.

Copyright Notice

Table of Contents

1. Introduction 2
2. SCHC ARQ-FEC fragmentation mode 3
2.1. SCHC ARQ-FEC protocol elements 4
2.2. Encoding and Decoding processes 5
2.2.1. Sender Behavior 5
2.2.2. Receiver Behavior 7
2.3. Fragmentation and Defragmentation processes 8
2.3.1. Assembler sub-process 9
2.3.2. Transporter sub-process 13
3. Conventions and Definitions 19
4. Security Considerations 19
5. IANA Considerations 19
6. Appendix A. State Machines for the transport sub-process . . 19
7. Appendix B. Example of an ARQ-FEC fragmentation mode 23
7.1. Encoding/Decoding process 23
7.1.1. Step 1: Create D-matrix 23
7.1.2. Step 2: Create C-matrix 23
7.2. Fragmentation/Defragmentation process 23
7.2.1. Case 1: Without loss and variable MTU. Sufficient fragments in the decoding process. 24
7.3. Case 2: With loss of fragments. Sufficient fragments in the decoding process. 25
7.4. Case 3: With loss of fragments. Insufficient fragments in the decoding process. 26
8. Changelog 26
Acknowledgments 27
Normative References 27
Authors' Addresses 27

1. Introduction

The fragmentation sublayer of the SCHC standard was designed to fragment a SCHC packet into tiles and transport each tile from the fragmenter to the reassembler using a fragmentation mode defined in [RFC8724]. The detection of missing tiles in the reassembler is based on the enumeration of each tile. If tiles are missing, the reassembler uses a parameter called "bitmap" to indicate to the

fragmenter which tiles are missing.

With regard to detecting tiles with errors, the reassembler uses a cyclic redundancy check algorithm applied to the entire SCHC packet received. If the check is unsuccessful, the reassembler discards the SCHC packet and the fragmentation is declared a failure. If missing tiles are detected, the reassembler requests retransmission of the missing tiles. In the event of multiple failed retransmissions, the receiver may discard the SCHC packet and declare the fragmentation as failed.

The SCHC WG has expressed interest in providing additional reliability mechanisms, such as FEC for fragments recovery.. Thus, this document presents a new mode of fragmentation based on channel coding and selective retransmission of tiles. The new mode can be adapted to different scenarios or technologies, and each of these scenarios can be associated with a profile. The new fragmentation mode allows each profile to define its own reliability fragmentation parameters.

The new fragmentation mode defined in this document is not associated with any specific profile. Hereinafter, the new fragmentation mode will be referred to as ARQ-FEC mode.

2. SCHC ARQ-FEC fragmentation mode

The ARQ-FEC mode uses channel coding to protect fragments from errors and losses. ARQ-FEC mode performs data delivery without retransmitting the lost tiles if the number of lost tiles is not higher than a configurable threshold.

ARQ-FEC mode is based in a Type II Hybrid ARQ/FEC mechanism and consists of two processes: encoding and fragmentation. Figure 1 shows both processes in the sender and the receiver. The SCHC packet coming from the compression sublayer is encoded and fragmented. At the receiver, the process is reversed. The defragmenter first creates a C-matrix from the tiles and then decodes the C-matrix to generate a SCHC packet.

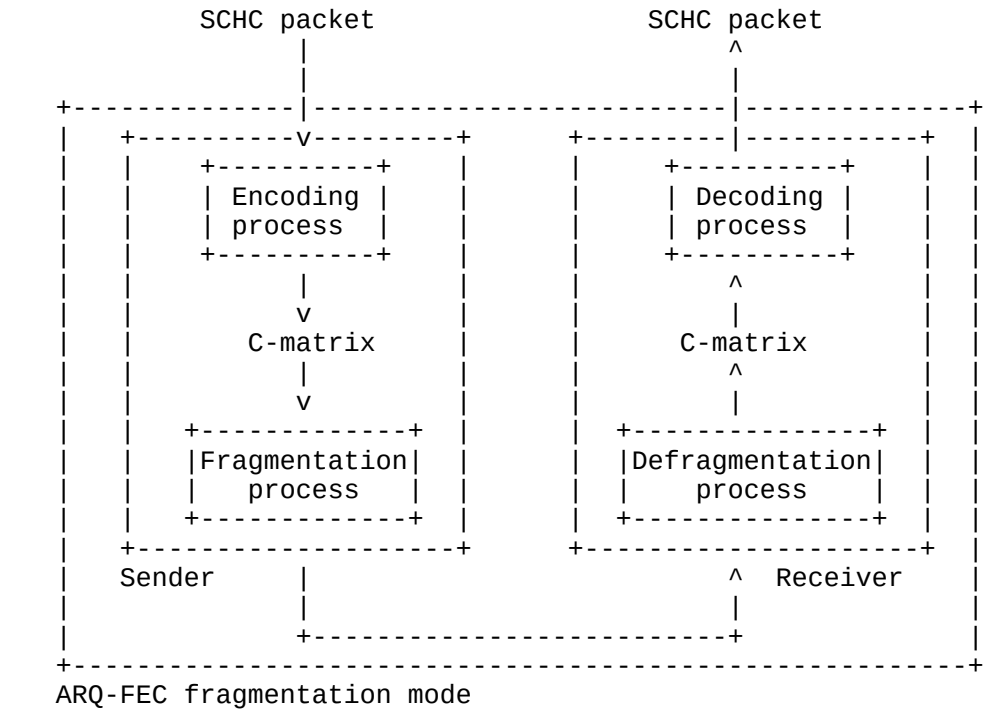


Figure 1: SCHC ARQ-FEC Fragmentation mode.

2.1. SCHC ARQ-FEC protocol elements

This subsection outlines the various components used to support the ARQ-FEC mode functionality defined in this document.

* SCHC Packet size: is the size in bits of the data packet received from the compression sublayer (see Figure 24 in [RFC8724]). In this document the SCHC Packet size is called "P". Each profile MUST define the maximum size of a SCHC packet that can be fragmented in ARQ-FEC mode. This maximum size is called "P_max".

* Symbol: data unit handled by the encoding process. Each symbol is treated as an indivisible unit, meaning it is either fully received or entirely lost. A symbol has "m" bits.

* Source Symbol: Unit of data used during the encoding process. A source symbol contains information from the original SCHC packet.

* Source Block: A subset of the source symbols. A source block has "k" symbols.

* Encoded Symbol: A symbol containing information generated by the Forward Error Correction (FEC) code which can be used to recover lost source symbols.

* Encoded Block: An encoded block consists of all the encoded symbols resulting from the FEC encoding of a single source block. An encoded block has "n" symbols.

* Residual coding bits: These are the left over bits resulting from dividing a SCHC packet into a D-matrix.

* Residual fragmentation bits: These are the bits left over from dividing a C-matrix into tiles of a given size.

2.2. Encoding and Decoding processes

At the sender, the encoding process creates a C-matrix from a SCHC packet. At the receiver, the decoding process works in reverse, i.e., it creates a SCHC packet from a C-matrix.

2.2.1. Sender Behavior

Upon reception of a SCHC packet from the compression sublayer, the sender divides the SCHC packet into source blocks. Each source block has "k" source symbols, and each source symbol has "m" bits. The encoding process consists of applying an FEC algorithm to each source block. Thus, the FEC (n,k) algorithm encodes a source block of k source symbols into an encoded block of n symbols.

The steps for dividing the SCHC packet and creating a C-matrix is as follows:

1. Divide the SCHC packet into S source blocks of k symbols. Each symbol has m bits.

    * S is calculated as floor(P/(k * m))

    * If (P mod (k * m)) equals zero, then there are no remaining bits.

    * If (P mod (k * m)) is not equal to zero, then there should be (P mod (k * m)) remaining bits. The remaining bits are called residual coding bits.

2. Arrange the source blocks in a grid pattern or matrix, this will be the D-matrix.

    * Each element of the D-matrix is a source symbol.

* Each row of the D-matrix will be a source block.

* The D-matrix has S rows and k columns.

* The residual coding bits MUST NOT be placed in the D-matrix.

* Figure 2 shows an example of the arrangement.

```
          <------- k columns ------->
        ^ SS(1,1)  SS(1,2)... SS(1,k)   Source block 1
        | SS(2,1)  SS(2,2)... SS(2,k)   Source block 2
S rows  |    :       :        :
        |    :       :        :
        v SS(S,1)  SS(S,2)... SS(S,k)   Source block S
```

SS: Source Symbol

Figure 2: Source blocks arranged in a D-matrix.

1. Apply the FEC algorithm to each source block. The FEC algorithm encodes k symbols from the source block and generates an encoded block with n symbols.

2. Arrange the encoded blocks in a grid pattern or matrix, this will be the C-matrix.

   * Each element of the C-matrix is an encoded symbol.

   * Each row of the C-matrix will be an encoded block.

   * The C-matrix has S rows and n columns.

   * Figure 3 shows an example of the arrangement.

```
          <------ n columns -------->
        ^ ES(1,1)  ES(1,2)... ES(1,n)  Encoded block 1
        | ES(2,1)  ES(2,2)... ES(2,n)  Encoded block 2
S rows  |    :       :         :
        |    :       :         :
        v ES(S,1)  ES(S,2)... ES(S,n)  Encoded block S
```

ES: Encoded symbol

Figure 3: Encoded blocks arranged in a C-matrix.

2.2.2. Receiver Behavior

In the receiver, the decoding process MUST create a SCHC packet from a C-matrix with enough encoded symbols. The number of encoded symbols required depends on the forward error correction code used.

The decoding process begins when the defragmentation process indicates that there are enough encoded symbols in the C-matrix to obtain a SCHC packet.

The steps for decoding a C-matrix and creating a SCHC packet is as follows:

1. Decode the encoded block in the first row of C-matrix. The generated source block contains the k symbols corresponding to the first row of matrix D.

2. Decode the encoded block in the second row of C-matrix. The generated source block contains the k symbols corresponding to the second row of matrix D.

3. Continue with the encoded block in the next row of the C-matrix. Decode all rows of the C-matrix.

4. The result is the D-matrix with all symbols decoded.

Once the D-matrix has been generated, reconstruct the SCHC packet by following these steps:

1. Select the k-symbols from the first row of the D-matrix. This symbols will be the first (k*m) bits of the SCHC packet.

2. Select the k-symbols from the second row of the D-matrix. This symbols will be the next (k*m) bits of the SCHC packet.

3. Continue selecting the k-symbols from each row until you reach the last row of the D-matrix.

4. Add the residual coding bits and padding bits to the end of the SCHC packet. The residual coding bits and padding bits are provided by the defragmentation process.

2.3. Fragmentation and Defragmentation processes

The fragmentation and defragmentation processes are responsible for transporting the C-matrix from the sender to the receiver. Unlike the others fragmentation processes defined in [RFC8724], the fragmentation and defragmentation processes are divided into two sub-processes: the assembler sub-process and the transporter sub-process. Figure 4 shows the architecture of the fragmentation and defragmentation processes at the sender and receiver.

At the sender, the fragmentation process receives from the Encoding process a C-matrix and the residual coding bits separately. The C-matrix is processed by the assembler sub-process. The assembler sub-process converts the C-matrix in an encoded SCHC packet that is passed to the transporter sub-process. The transporter sub-process divides the encode SCHC packet in tiles. The tiles are sent to the receiver via Regular SCHC fragment messages.

At the receiver, the defragmentation process works in reverse. The transporter sub-process receive the tiles that are passed to the assembler sub-process. The assembler subprocess constructs a C-matrix using the tiles. When the assembler subprocess has enough symbols for the decoder process to decode C-matrix, the transporter subprocess ends the fragmentation session and the C-matrix is passed on to the decoding process.

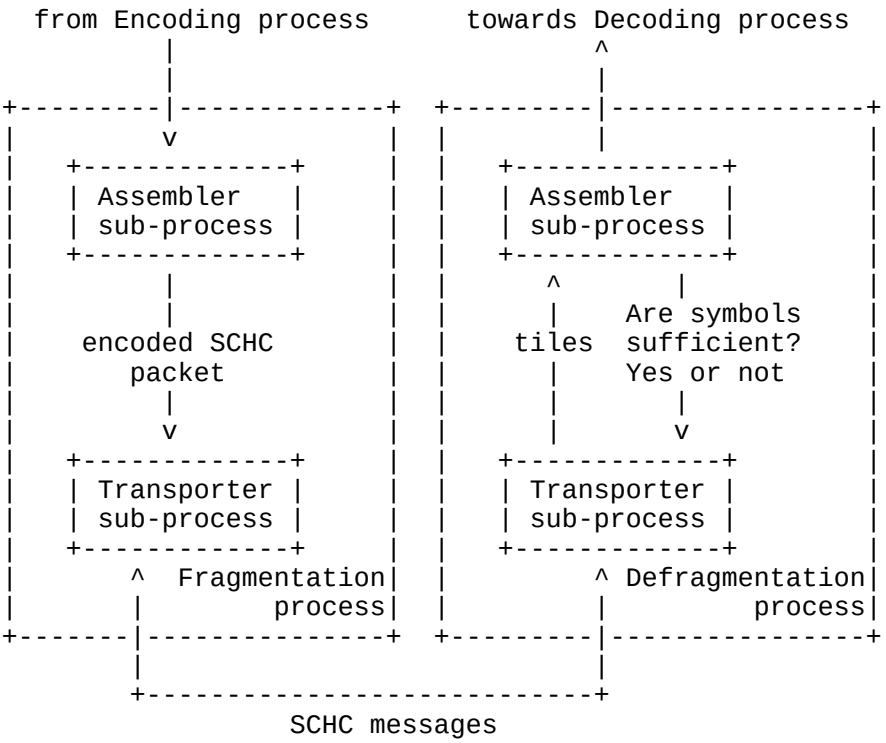


Figure 4: Fragmentation and Defragmentation process and its subprocesses.

2.3.1. Assembler sub-process

2.3.1.1. Sender Behavior

On the sender side, the assembler sub-process is responsible for transforming the C-matrix into an encoded SCHC packet. To do this, the assembler sub-process selects the S-bytes of the first column of the C-matrix. These S-bytes will be the first S-bytes of the encoded SCHC packet. Next, the sub-process selects the S-bytes of the second column of the C-matrix. These will follow in the encoded SCHC packet, and so on. Figure 5 shows how the encoded SCHC packet is constructed from the C-matrix.

> Important: Residual encoding bits MUST NOT be included in the encoded SCHC packet. Residual coding bits MUST be included in the last tile.

```
<---------------------- encoded SCHC packet ---------------------->
C(1,1) C(2,1)...C(S,1) C(1,2) C(2,2)...C(S,2)...C(1,cwl)...C(S,cwl)
```

Figure 5: Symbols of a encoded SCHC packet.

2.3.1.2. Receiver Behavior

In the receiver, the assembler sub-process is responsible for three tasks: placing each tile in the C-matrix, verifying whether there are enough symbols in each row to decode, and indicating to the transporter sub-process which tiles need to be retransmitted.

2.3.1.2.1. Placing tiles in C-matrix

When the transporter sub-process receives a regular SCHC fragment, the transporter sub-process delivers the tiles to the assembler sub-process to place them in the C-matrix.

> If the receiver has not received parameter S, then the following steps cannot be executed and the receiver must wait to receive parameter S before continuing with the verification process. Parameter S is transported in the first tile of the first fragment of the first window.

The procedure for placing a tile in the C-matrix is as follows:

1. Convert the FCN and W of each tile in a correlative tile number (ctn). The correlative tile number starts at zero. For example, tile 62 of window 0 (FCN=62, W=0) corresponds to ctn=0. Tile 61 of window 0 (FCN=61, W=0) corresponds to ctn=1, and so on. To determine the correlative tile number (ctn), use the following equation:

   ctn = WINDOW_SIZE*(W+1) - FCN - 1

where WINDOW_SIZE is defined by each profile. FCN is the Fragment Compressed Number of each tile. W is the field that indicates the window to which the tile belongs.

1. Convert the correlative tile number (ctn) into a position (row, column) within the matrix.

   * To determine the *row*, use the following equation:

     row = (ctn-1)*ts + 1 - floor((ctn-1)*ts/S)*S

   * To determine the *column*, use the following equation:

     column = floor((ctn-1)*ts/S) + 1

where ts is the tile size in symbols. S is the number for row in the C-matrix.

2. The tile must be placed vertically in the matrix. This means that the first symbol on the tile is placed in the position (row, column) determined in the previous point. The second symbol on the tile must be placed in the same column but in row (row+1). The third symbol must be placed in row (row+2) and so on. If the last row of the matrix is reached and there are still symbols on the tile to be placed in the C-matrix, continue in the first row of the next column. For example, let us consider that a tile is equivalent to 4 symbols, C-matrix has 7 rows (S=7) and the tile must start in row 5, column 1. Then, the first symbol will go in row 5, column 1. The second symbol will go in row 6, column 1. The third symbol will go in row 7, column 1, and the fourth symbol will go in row 1, column 2. Note that the last symbol cannot be placed in row 8 because there is no such row. Therefore, you must start in the first row of the next column.

2.3.1.2.2. Checking if there are enough symbols

In ARQ-FEC mode, the decoding process does not need to have the n encoded symbols in each row of the C-matrix to recover the original k symbols. Therefore, the assembler process only needs to have k encoded symbols in each row of the C-matrix to pass this matrix to the decoding process. If the C-matrix has enough symbols, the sender can stop transmitting tiles and the receiver can start the decoding process.

When the receiver receives a SCHC regular fragment, the transporter sub-process extracts the tiles from the message and sends them to the assembler sub-process. The assembler sub-process places the tiles in the C-matrix following the steps explained in the section Section 2.3.1.2.1. Once the tiles are in place, the task of checking whether there are enough symbols in each row can begin.

First, the verification task must check the number of symbols in the first row of C-matrix. If the number of symbols is greater than or equal to k, then that row is marked as decodeable; otherwise, it is marked as not decodeable. If the verification task checks a row already marked as decodeable, the task must skip that row and continue with the next one.

The verification task must review all rows of C-matrix to complete the verification task associated with the reception of a SCHC regular fragment.

If at any point the verification task detects that all rows of the C-matrix are decodable, then the assembler sub-process MUST indicate to the transporter sub-process that the C-matrix has sufficient symbols.

2.3.1.2.3. Selecting the tiles to retransmit

If the receiver receives a SCHC All-1 fragment message, the transporter subprocess must indicate to the assembler subprocess that it has received a SCHC All-1 fragment along with the last tile. The assembler subprocess obtains the residual fragmentation bits from the last tile and adds them to C-matrix. When the assembler sub-process adds the residual fragmentation bits to the matrix, two events may occur:

* If all rows are marked as decodeable, then the assembler sub-process must indicate to the signaling sub-process that there are enough symbols in the C-matrix and that it can end the fragmentation session, or

* If one or more rows of C-matrix were marked as undecodable, then the assembler sub-process must perform the following steps:

  1. Select a row marked as undecodable. Consider that the row has m received symbols, select the (k-m) lost symbols, where k is the number of symbols in each row of D-matrix (see Figure 2).

  2. Convert the position of the missing symbol into a correlative tile number (ctn). The position of a missing symbol is defined by the row and column to which it belongs. To obtain the correlative tile number (ctn), use the following equation:

     ctn = ceil((row + S*(col - 1))/ts)

     where ts is the tile size in symbols. S is the number for row in the C-matrix.

  3. To obtain the Fragment Compressed Number (FCN) of the correlative tile number (ctn), use the following equation:

     fcn = WINDOW_SIZE * (floor(ctn/WINDOW_SIZE) + 1) - ctn - 1

     where WINDOW_SIZE is defined by each profile. ctn is the correlative tile number.

  4. To obtain the window to which a tile (w) belongs from the correlative tile number (ctn), use the following equation:

```
w = floor(ctn/WINDOW_SIZE)
```

5. Pass the list of missing tiles to the transporter subprocess. A tile is defined by its Fragment Compressed Number (FCN) and the window to which it belongs.

2.3.2. Transporter sub-process

ARQ-FEC mode supports the following features:

* the mode works with L2 technologies that have a variable MTU and may deliver packets out of order.

* the mode requires the L2 layer to provide a feedback channel from the defragmenter back to the fragmenter.

* the mode uses window.

* all tiles except the last one MUST be of the same size. These tiles are called regular tiles.

* A SCHC Fragment message carries one or several contiguous regular tiles, which may span multiple windows.

Unlike ACK-on-Error and ACK-Always modes, the success of the transfer of tiles is associated with success in decoding the C-matrix. Thus, if the fragmentation/defragmentation process loses some tiles, the C-matrix can still be decoded, the SCHC packet obtained, and the fragmentation declared successful. Therefore, the receiver responds with a SCHC Compound ACK when there are enough symbols to decode the C-matrix or when, at the end of the transmission of all tiles, there are not enough symbols to decode the C-matrix. For this reason, the SCHC Compound ACK MUST NOT be sent after the reception of each window. The sender can advance to next windows even before it has ascertained that all tiles belonging to previous windows have been correctly received. The SCHC Compound ACK message is defined in [RFC9441].

The ARQ-FEC mode in the receiver MUST send a SCHC Compound ACK only in these four situations:

* If the receiver receives a SCHC regular fragment message with the S parameter,

* if the receiver receives a SCHC regular fragment message and the assembler sub-process verify that there are enought symbols to decode the C-matrix,

* if the receiver receives a SCHC All-1 message and the assembler sub-process verify that there are enought symbols to decode the C-matrix,

* if the receiver receives a SCHC All-1 message and the assembler subprocess verifies that there are not enough symbols to decode the C-matrix.

The fragmented encoded SCHC Packet transmission concludes when:

* the receiver verify that there are enough symbols to decode the C-matrix.

* too many retransmission attempts were made, or

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

Each Profile MUST specify which RuleID value(s) corresponds to SCHC F/R messages operating in this mode.

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

Each Profile, for each RuleID value, MUST define: * the tile size (a tile must be a multiple of a symbol and MUST be at least the size of an L2 word.), * the value of M, * the value of N, * the value of WINDOW_SIZE, which MUST be strictly less than 2^N, * the size and algorithm for the RCS field, * the value of T, * the value of MAX_ACK_REQUESTS, * the expiration time of the Retransmission Timer, * the expiration time of the Inactivity Timer, * the expiration time of the S Timer,

For each active pair of RuleID and DTag values, the sender MUST maintain: * one Attempts counter, * one Retransmission Timer, * one S Attempts counter, and * one S Timer

For each active pair of RuleID and DTag values, the receiver MUST maintain: * one Inactivity Timer, and * one Attempts counter.

2.3.2.1. Sender Behavior

At the beginning of the fragmentation of a encoded SCHC packet

* The sender MUST select the values for RuleID and DTag for this encoded SCHC packet. In addition, the sender MUST initialise the Attempts counter and the S Attempts counter to 0 for these RuleID and DTag values.

To fragment an encoded SCHC packet, the sender MUST split the encoded SCHC packet into tiles of the same size. These tiles are called "regular tiles". The remaining part of the division will be the residual fragmentation bits. A Regular SCHC Fragment message carries in its payload one or several contiguous tiles. If more than one tile is carried in one Regular SCHC Fragment:

* The selected tiles MUST be contiguous in the original encoded SCHC Packet, and

* the selected tiles MUST be placed in the SCHC fragment payload next to each other, in the same order as they appear in the encoded SCHC packet, and

* the FCN field MUST contain the tile index of the first tile sent in that SCHC Fragment, and

* the sender MUST fill the W field with the window number of the first tile sent in that SCHC Fragment.

In ARQ-FEC mode, the receiver needs to obtain the S parameter to know how many rows the C-matrix should have. With the C-matrix constructed, the receiver can know when there are enough symbols to decode an encoded SCHC packet. The sender sends the S parameter in the first tile of the first window. This means that the encoded SCHC packet will go from the second tile of the first window onwards. In summary: * the first tile (FCN=WINDOW_SIZE-1) of the first window (W=0) contains the parameter S. The parameter S indicates the number of rows in the C-matrix, and * the first tile of the encoded SCHC packet MUST be transported in the fragment with FCN = WINDOW_SIZE-2 of the first window (W=0).

The sender must wait for the reception of a SCHC Compound ACK to the tile carrying parameter S. The SCHC Compound ACK must contain the field W=00, C=1, and without bitmap.

The last tail consists of the residual coding bits and the residual fragmentation bits. The last tile MUST be sent in an All-1 SCHC message. In an All-1 SCHC Fragment message, the sender MUST fill the W field with the window number of the last tile of the encoded SCHC Packet. The sender must wait for the reception of a SCHC ACK after sending an All-1 SCHC. In this case, two situations may occur:

* If the sender receives a SCHC ACK containing the field W=11 and C=1, then the All-1 SCHC was successfully received, the receiver has enough symbols, and the sender can terminate the fragmentation session.

* If the sender receives a SCHC ACK containing the C=0 field, then the All-1 was successfully received, the receiver does not have enough symbols, and the bitmap contains the tiles that must be retransmitted. At this point, the sender must retransmit all lost tiles and wait for a SCHC ACK confirming the end of the session.

At any time, after receiving confirmation of successful receipt of parameter S, if the sender receives a SCHC ACK with parameters W=01 and C=1, this means that the receiver has enough symbols to decode matrix C. The sender must respond with an All-1 SCHC and wait for confirmation.

In brief, the fragment sender MUST listen for SCHC Compound ACK messages after having sent:

* a SCHC Regular Fragment, or

* an All-1 SCHC Fragment, or

* a SCHC ACK REQ.

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

* it MUST increment the Attempts counter, and

* it MUST reset the Retransmission Timer.

On Retransmission Timer expiration:

* if the Attempts counter is strictly less than MAX_ACK_REQUESTS, the fragment sender MUST send either the All-1 SCHC Fragment or a SCHC ACK REQ with the W field equal to zero.

* otherwise, the fragment sender MUST send a SCHC Sender-Abort, and it MAY exit with an error condition.

Each time a fragment sender sends a Regular SCHC Fragment with the S parameter:

* it MUST increment the S Attempts counter, and

* it MUST reset the S Timer.

On S Timer expiration:

* if the S Attempts counter is strictly less than MAX_ACK_REQUESTS, the fragment sender MUST send a Regular SCHC Fragment with the S parameter in the first tile.

* otherwise, the fragment sender MUST send a SCHC Sender-Abort, and it MAY exit with an error condition.

On receiving a SCHC Compound ACK:

* if the W field is set to 00 and the C bit is set to 1, the sender MUST assume that the receiver already knows the S parameter.

* If field W is set to 01 and bit C is set to 1, the sender MUST assume that the receiver has enough symbols to decode the C-matrix. The sender MUST send an All-1 SCHC and wait for a SCHC ACK confirming receipt of the All-1 SCHC.

* If the W field is set to 11 and the C bit is set to 1, the sender MUST terminate the fragmentation session.

* if the C bit is set to 0, the fragment sender MUST send SCHC Fragment messages containing all the tiles that are reported missing in the SCHC Compound ACK.

2.3.2.2. Receiver Behavior

When the reassembler receives a SCHC fragment with a RuleID that is not currently being processed, then:

* The receiver MUST start a process to assemble a new encoded SCHC Packet with that RuleID,

* the receiver MUST start an Inactivity Timer and an Attempts counter to 0 for that RuleID, and

* if the receiver is under-resourced to do this, it MUST respond to the sender with a SCHC Receiver-Abort.

Whenever a SCHC F/R message arrives for the current RuleID, the receiver MUST reset the corresponding Inactivity Timer.

On receiving a SCHC Fragment message, the receiver determines what tiles were received, based on the payload length and on the FCN field of the SCHC Fragment. When the receiver receives a SCHC Fragment message, the following situations may occur:

* If the FCN field is set to WINDOW_SIZE - 1 and the W field is set to 0, then the first tile of the SCHC fragment has the S parameter. In this case, the receiver MUST respond with a SCHC ACK with the W field set to 0 and the C field set to 1.

* Otherwise:

  - If the N bits of the FCN field are not all set to one, then the tiles MUST be assembled based on the a priori known tile size. In this case, the transporter sub-process must deliver the tiles to the assembler sub-process and wait a response from assembler sub-process:

    o if there are enough symbols, then the transporter sub-process MUST send a SCHC Compound ACK with the W field set to 1 and the C field set to 1 and wait for an All-1 SCHC message.

    o if there are not enough symbols, then the transporter sub-process SHOULD do nothing.

  - Otherwise:

    o If the N bits of the FCN field are all set to one, then the transporter sub-process must deliver the last tile to the assembler sub-process and wait a response from assembler sub-process:

      + if there are enough symbols, then the transporter sub-process MUST send a SCHC Compound ACK with the W field set to 3 and the C field set to 1, and end the fragmentation session.

      + If there are not enough symbols, then the transporter sub-process MUST send a SCHC Compound ACK with the C field set to 0 and with the bitmaps indicating which tiles are missing.

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

Upon receiving a SCHC Sender-Abort, the receiver MAY exit with an error condition.

Upon expiration of the Inactivity Timer, the receiver MUST send a SCHC Receiver-Abort, and it MAY exit with an error condition.

On the Attempts counter exceeding MAX_ACK_REQUESTS, the receiver MUST send a SCHC Receiver-Abort, and it MAY exit with an error condition.

Fragmentation session concludes when:

* a Sender-Abort has been received, or

* the Inactivity Timer has expired, or

* the Attempts counter has exceeded MAX_ACK_REQUESTS, or

* an All-1 SCHC message is sent with the W field set to 3 and the C field set to 1.

3. Conventions and Definitions

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

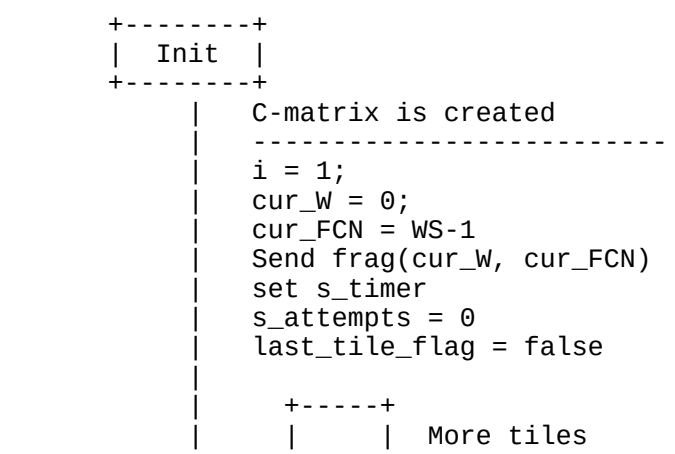
4. Security Considerations

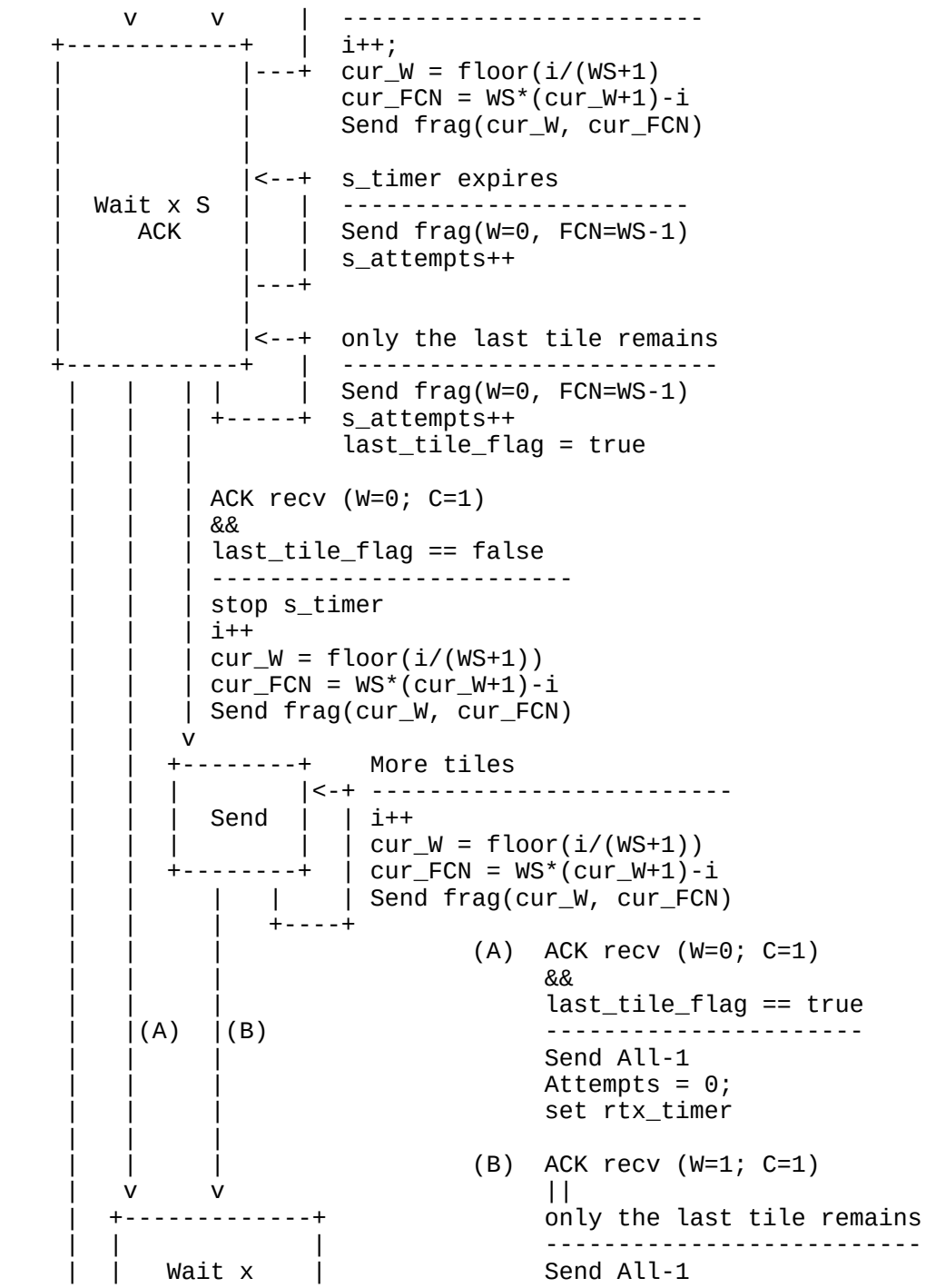
This document does not add any security considerations and follows the [RFC8724].

5. IANA Considerations

This document has no IANA actions.

6. Appendix A. State Machines for the transport sub-process




i++;
cur_W = floor(i/(WS+1)
cur_FCN = WS*(cur_W+1)-i
Send frag(cur_W, cur_FCN)
s_timer expires
Wait x S
ACK
Send frag(W=0, FCN=WS-1)
s_attempts++
only the last tile remains
Send frag(W=0, FCN=WS-1)
s_attempts++
last_tile_flag = true
ACK recv (W=0; C=1)
&&
last_tile_flag == false
stop s_timer
i++
cur_W = floor(i/(WS+1))
cur_FCN = WS*(cur_W+1)-i
Send frag(cur_W, cur_FCN)
More tiles
Send
i++
cur_W = floor(i/(WS+1))
cur_FCN = WS*(cur_W+1)-i
Send frag(cur_W, cur_FCN)
(A)
ACK recv (W=0; C=1)
&&
last_tile_flag == true
Send All-1
Attempts = 0;
set rtx_timer
(B)
ACK recv (W=1; C=1)
||
only the last tile remains
Send All-1
Wait x

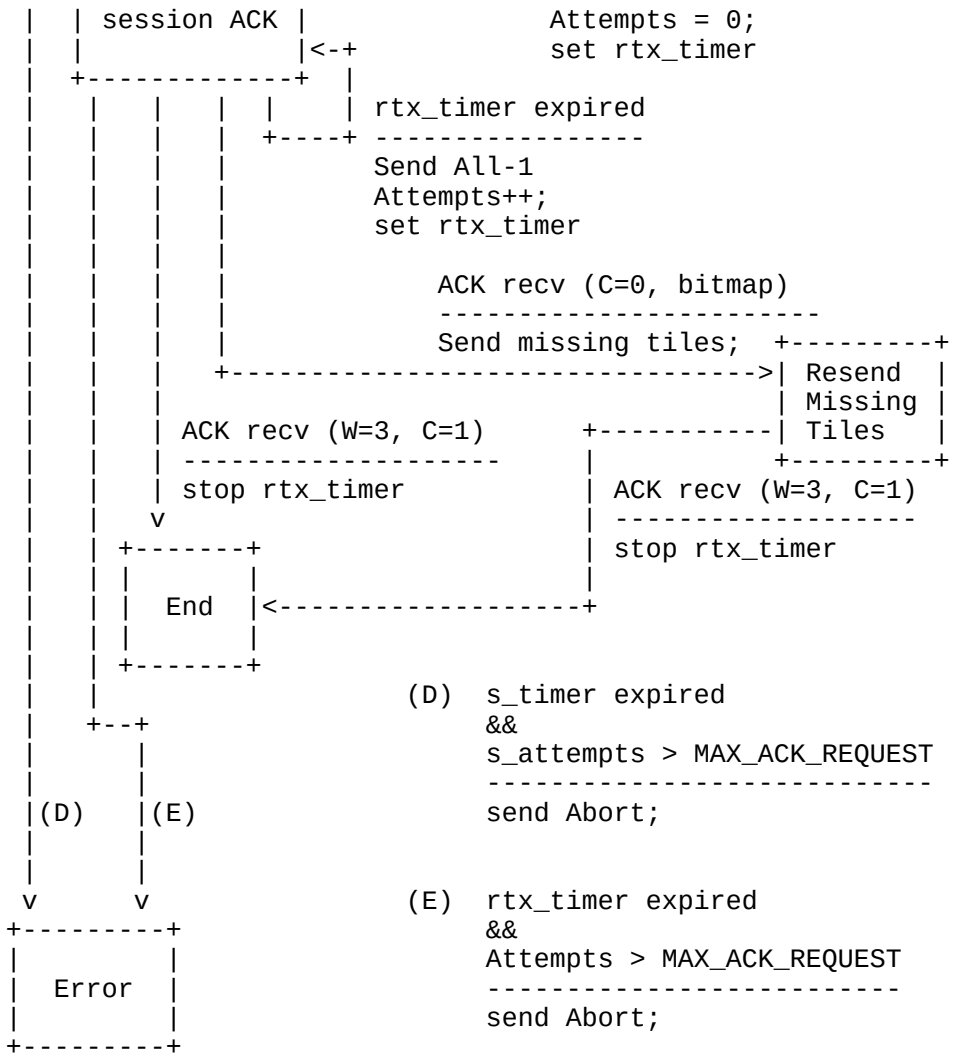
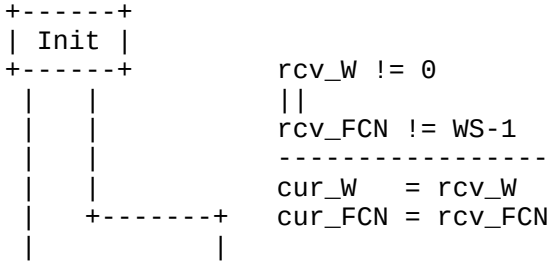


Figure 6: Sender State Machine for the ARQ-FEC Mode.



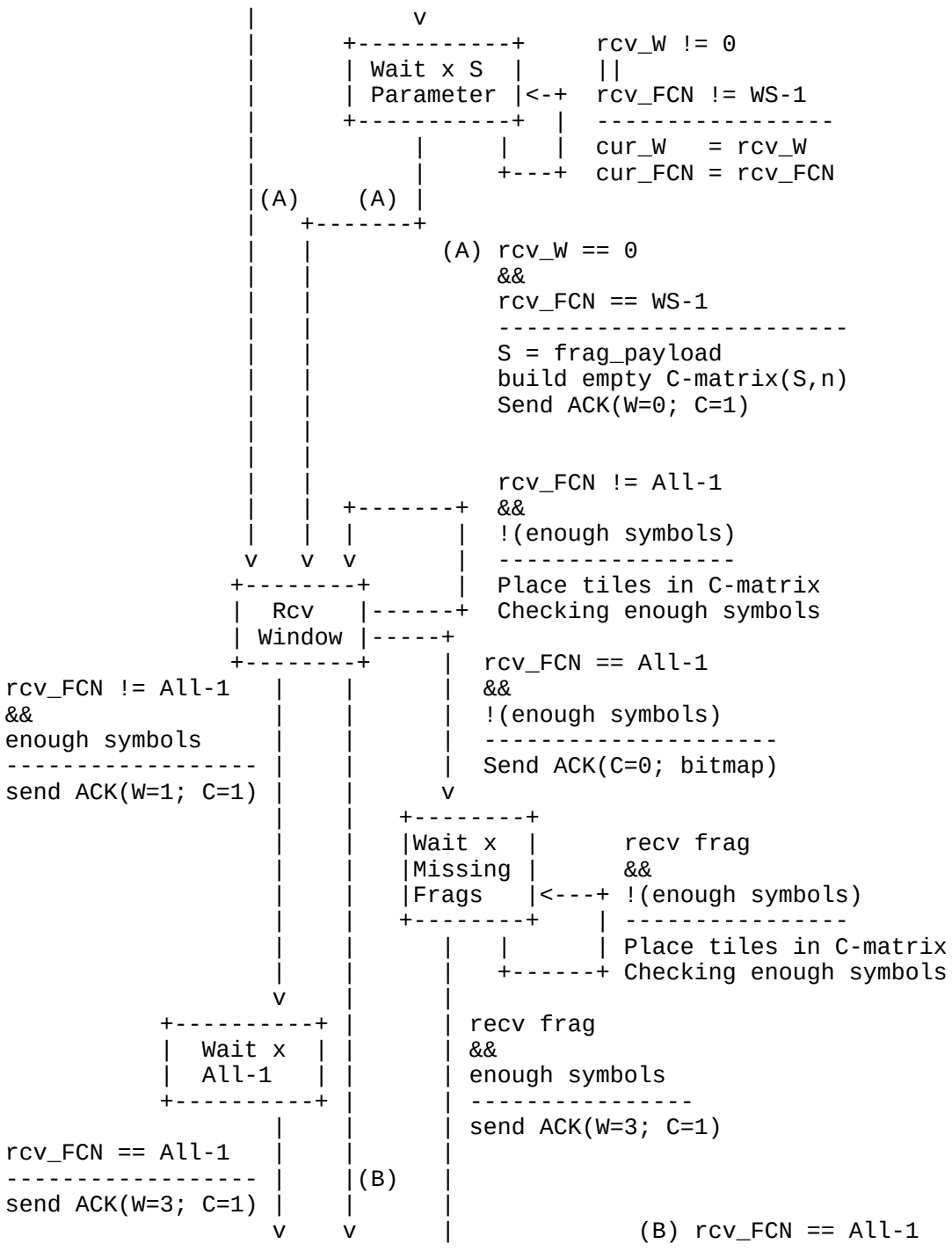

v
Wait x S Parameter
rcv_W != 0
rcv_FCN != WS-1
cur_W = rcv_W
cur_FCN = rcv_FCN
(A)
(A)
(A) rcv_W == 0
&&
rcv_FCN == WS-1
S = frag_payload
build empty C-matrix(S,n)
Send ACK(W=0; C=1)
rcv_FCN != All-1
&&
!(enough symbols)
Place tiles in C-matrix
Checking enough symbols
Rcv Window
rcv_FCN == All-1
&&
!(enough symbols)
Send ACK(C=0; bitmap)
rcv_FCN != All-1
&&
enough symbols
send ACK(W=1; C=1)
Wait x Missing Frags
recv frag
&&
!(enough symbols)
Place tiles in C-matrix
Checking enough symbols
Wait x All-1
recv frag
&&
enough symbols
send ACK(W=3; C=1)
rcv_FCN == All-1
send ACK(W=3; C=1)
(B)
(B) rcv_FCN == All-1

Figure 7: Receiver State Machine for the ARQ-FEC Mode.

7. Appendix B. Example of an ARQ-FEC fragmentation mode

This example represents the transmission of a SCHC packet generated by the compression sublayer. The transmission uses the ARQ-FEC fragmentation sublayer over LoRaWAN technology:

ARQ-FEC fragmentation mode parameters: * Tile size (ts): 10 symbols * Bits per symbol (m): 8 bits/symbol * Source block size (k): 4 symbols * Encoded Block Size (n): 7 symbols * SCHC packet size (P): 6445 bits * Maximum Transmittion Unit (MTU) in AU915-928 LoRaWAN frequency band with DR5 (222 bytes) and DR3 (115 bytes)

7.1. Encoding/Decoding process

7.1.1. Step 1: Create D-matrix

* S = floor(P/(k * m)) = 201 rows
* D-matrix size = (S * k) * m = 6432 bits
* Residual coding bits = (P mod (k * m)) = 13 bits

7.1.2. Step 2: Create C-matrix

* C-matrix size = (S * n) * m = 11256 bits
* Encoding overhead size = S * (n - k) * m = 4824 bits
* Encoded SCHC packet size = C-matrix size = 11256 bits

7.2. Fragmentation/Defragmentation process

* Full regular tiles = floor(Encoded SCHC packet size/(ts * m) = 140 tiles
* Residual fragmentation bits = (Encoded SCHC packet size) mod (ts * m) = 56 bits
* All-1 SCHC payload = Residual fragmentation bits + Residual coding bits + padding = 56 + 13 + 3 = 72 bits

* W: 2 bits

* N: 6 bits

* WINDOW_SIZE: (2^N)-1 = 63

7.2.1. Case 1: Without loss and variable MTU. Sufficient fragments in the decoding process.

Each regular SCHC fragment carries a SCHC packet with a 1-byte SCHC header and a SCHC payload of 22 tiles (DR5) or 11 tiles (DR3). Figure 8 shows the first regular SCHC fragment.

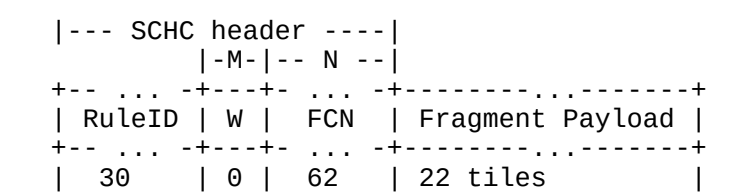


Figure 8: First regular SCHC fragment. Case 1: without loss and varible MTU.

The All-1 SCHC fragment carries the 72 bits (including the padding bits). Figure 9 shows the All-1 SCHC fragment of this example.

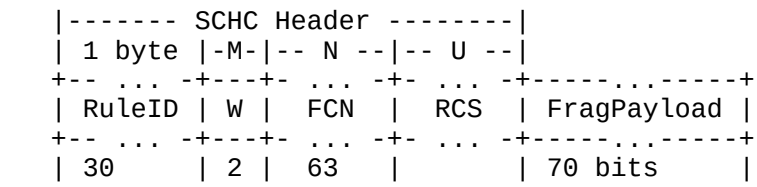


Figure 9: All-1 SCHC fragment. Case 1: without loss and varible MTU.

Figure 10 illustrates the transmission of a SCHC Packet using the ARQ-FEC fragmentation proccess. During the transmission of the encoded SCHC packet, the MTU is modified according to the LoRaWAN data rate.

If there are no losses, the number of tiles needed to decode a C-matrix is given by:

enough number of tiles = ceil((S * k)/ ts) = 81 tiles

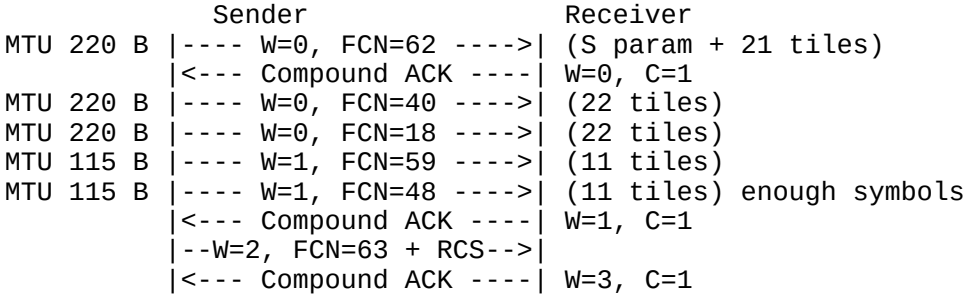


Figure 10: Message flow for ARQ-FEC mode (Case 1).

7.3. Case 2: With loss of fragments. Sufficient fragments in the decoding process.

In this case, there is a loss of fragments, but the decoding process indicate to defragmentation process that there are enougth tiles to decode the encode SCHC packet. Figure 11 illustrates the transmission of a encode SCHC Packet using the ARQ-FEC fragmentation mode. During transmission of the encoded SCHC packet, fragments 2 and 4 are lost.

To determine the number of tiles sufficient to decode a matrix, it is necessary to construct C-matrix. The process for determining the location of each tile in C-matrix and how to determine if there are enough symbols is explained in section {receiver_behavior}. In this case, if fragments 2 and 4 are lost, the receiver needs to receive up to the tile with FCN=8 and W=1.

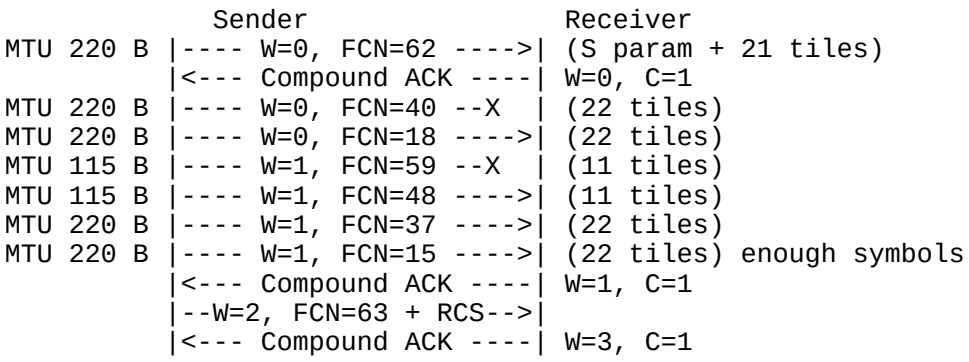


Figure 11: Message flow for ARQ-FEC mode (Case 2).

7.4. Case 3: With loss of fragments. Insufficient fragments in the decoding process.

In this case, fragment loss occurs and the decoding process signals to defragmentation process that there are not enough tiles to decode the encoded SCHC packet. Figure 12 illustrates the transmission of an encoded SCHC packet using the ARQ-FEC fragmentation mode. During the transmission of the encoded SCHC packet, fragments 2, 4, and 6 are lost.

To determine the number of tiles sufficient to decode a matrix, it is necessary to construct C-matrix. The process for determining the location of each tile in C-matrix and how to determine if there are enough symbols is explained in section {receiver_behavior}. In this case, if fragments 2, 4, and 6 are lost, the receiver only needs to receive the 9 tiles from fragment 6.

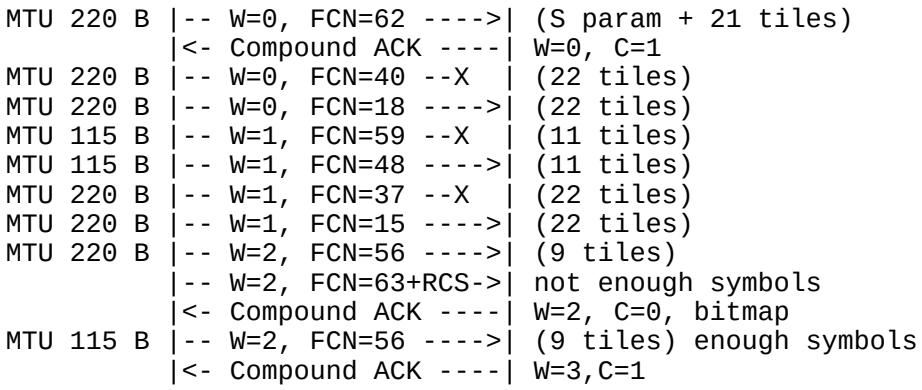


Figure 12: Message flow for ARQ-FEC mode (Case 3).

8. Changelog

* Version 00
  - Initial version
* Version 01
  - This version removes references to DtS-IoT.
  - This version adds state machines for the transporter sub-process.
  - This version features a new design for the ARQ-FEC mode.

Acknowledgments

TODO acknowledge.

Normative References

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

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

[RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <https://www.rfc-editor.org/info/rfc8724>.

[RFC9441] Z炭単iga, J., Gomez, C., Aguilar, S., Toutain, L., C辿spedes, S., and D. Wistuba, "Static Context Header Compression (SCHC) Compound Acknowledgement (ACK)", RFC 9441, DOI 10.17487/RFC9441, July 2023, <https://www.rfc-editor.org/info/rfc9441>.

Authors' Addresses

Rodrigo Munoz-Lara
Universidad de Chile
Santiago
Chile
Email: rmunozlara@ing.uchile.cl

Sandra Cespedes
Concordia University
Montreal
Canada
Email: sandra.cespedes@concordia.ca