SCHC Working Group
Internet-Draft
Intended status: Informational
Expires: 2 October 2026

R. Munoz-Lara
Universidad de Chile
S. Cespedes
Concordia University
J. A. Fernandez
IMT Atlantique
31 March 2026

Static Context Header Compression and Fragmentation ARQ/FEC mode
draft-munoz-schc-over-dts-iot-02

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 2 October 2026.

Copyright Notice

Table of Contents

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. When losses are detected, the reassembler reports the missing tile indices to the fragmenter as part of the acknowledgement mechanism, according to the selected fragmentation mode.

With regard to detecting tiles with errors, the reassembler uses a Reassembly Check Sequence (RCS) computed over the entire SCHC packet. If the integrity check is unsuccessful, the reassembler discards the SCHC packet and the fragmentation process 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 reassembler may discard the SCHC packet and declare the fragmentation 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 on a Type II Hybrid ARQ/FEC mechanism and consists of two main processes: encoding and fragmentation. Figure 1 shows both processes in the sender and the receiver. The SCHC packet coming from the compression sublayer is first encoded into a coded data structure and then fragmented. At the receiver, the process is reversed. The reassembler first reconstructs the coded data structure from the received tiles and then decodes it to recover the SCHC packet. The format of the coded data structure depends on the

selected encoding geometry. This specification defines two encoding geometries: a matrix-based encoding geometry ("C-matrix") and a streaming encoding geometry ("C-Stream").

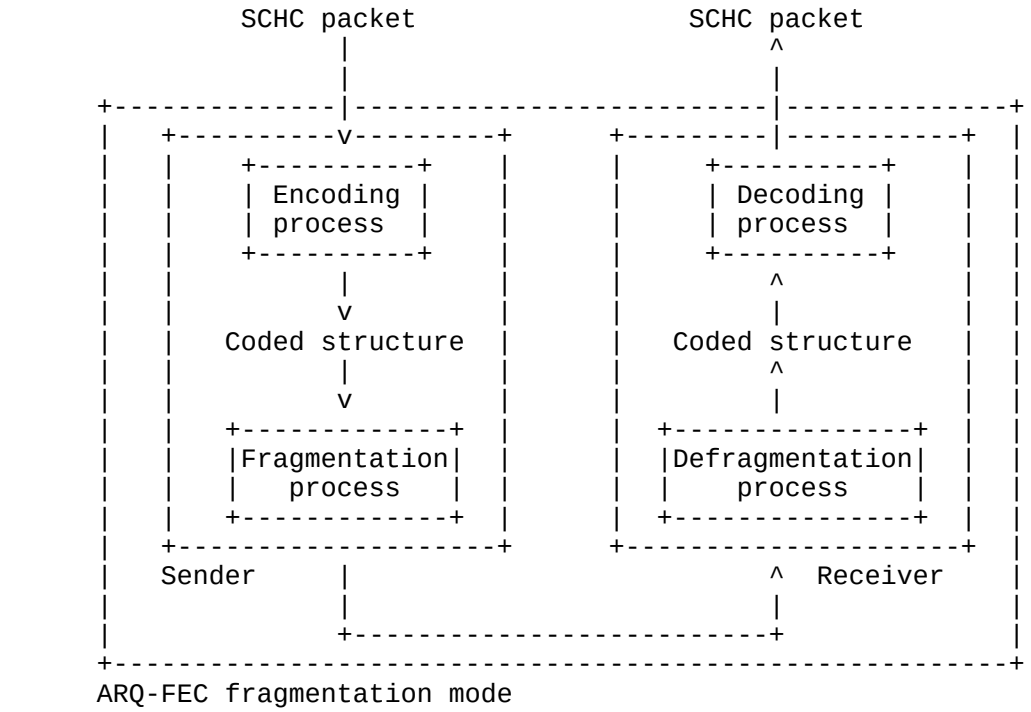


Figure 1: SCHC ARQ-FEC Fragmentation mode.

2.1. Protocol Elements

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

* SCHC Packet size: 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: A 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: A unit of data used during the encoding process. A source symbol contains information from the original SCHC packet.

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

* Encoding Geometry: The method used to arrange data from the SCHC Packet and to organize both source and encoded symbols into either a matrix or a stream data structure.

* Coded Data Structure: The arrangement of symbols resulting from the FEC encoding process, as defined by the selected encoding geometry. It serves as the input to the fragmentation process and is represented either as a C-Matrix or a C-Stream.

* D-matrix (Data Matrix): The conceptual two-dimensional arrangement of the source symbols derived from the SCHC Packet prior to FEC encoding. Each row of the D-matrix is a Source Block.

* D-Stream: The conceptual sequential arrangement of the source symbols derived from the SCHC Packet prior to FEC encoding. A Source Block is a group of sequential source 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.

* C-matrix (Coded Matrix): The conceptual two-dimensional arrangement of encoded symbols produced by applying the selected FEC scheme to each row of the D-matrix. Each row of the C-matrix is an Encoded Block corresponding to the associated D-matrix row.

* C-Stream: The conceptual sequential arrangement of encoded symbols produced by applying the selected FEC scheme to the D-Stream. Encoded symbols are generated and organized as a stream corresponding to the source blocks.

* 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 the coded data structure from a SCHC packet, according to the selected encoding geometry. At the receiver, the decoding process works in reverse, i.e., it creates a SCHC packet from the coded data structure.

2.2.1. Encoding Geometries

The encoding and decoding processes support two distinct encoding geometries, which define how symbols are to be organized, and how repair information is generated and applied. The selected encoding geometry impacts both the performance of the FEC mechanism and the resource requirements of the system.

This specification defines the following encoding geometries:

* Matrix-based geometry: Symbols are arranged in a two-dimensional structure, enabling flexible FEC strategies such as structured interleaving, configurable parity coverage, and repeated multi-pass decoding. This geometry supports more robust error correction capabilities.

* Stream geometry: Symbols are organized as a sequential stream, where repair symbols are generated over a sliding or sequential set of symbols (block). This geometry enables simpler encoding and decoding operations.

The choice of encoding geometry reflects a trade-off between reliability and resource consumption. The matrix-based geometry provides stronger recovery capabilities and are suited for use cases requiring high reliability under significant loss conditions. The stream geometry, on the other hand, is designed for environments with constrained processing or memory resources, providing lightweight protection against losses at a lower implementation cost.

The selection of the encoding geometry is configured within the SCHC fragmentation rule and determines the encoding and decoding procedures described in the following sections.

2.2.2. 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.

2.2.2.1. Matrix-Based Encoding

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.

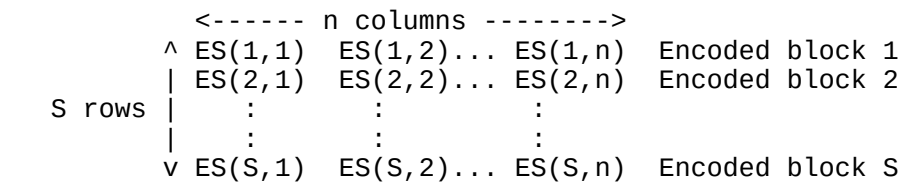
1. Apply the FEC algorithm to each source block. The FEC algorithm encodes k symbols from the source block and generates an encoded block with n symbols.
2. Arrange the encoded blocks in a grid pattern or matrix, this will be the C-matrix.

* 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.



ES: Encoded symbol

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

2.2.2.2. Stream Encoding

In the stream encoding geometry, the encoding process is performed as a sequence of steps applied to the SCHC Packet.

First, the SCHC Packet is partitioned into source symbols of size "m" bits. Each symbol is treated as an indivisible unit by the encoding process.

The resulting sequence of source symbols is then grouped into source blocks, each containing "k" source symbols. The number of source blocks depends on the size of the SCHC Packet and is not fixed a priori.

Next, the FEC encoding is applied independently to each source block. For each source block, the encoding process generates the corresponding encoded symbols, forming an encoded block of "n" symbols.

Finally, the encoded blocks are arranged sequentially to form the C-Stream. The encoded symbols are ordered according to the sequence of their corresponding source blocks, producing a continuous stream of symbols that constitutes the coded data structure.

2.2.3. Receiver Behavior

In the receiver, the decoding process MUST create a SCHC packet from the coded data structure 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 coded data structure to obtain a SCHC packet.

2.2.3.1. Matrix-Based Decoding

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. These will be the first (k * m) bits of the SCHC packet.

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

3. Continue selecting the k-symbols from each row until 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.

2.2.3.2. Stream Decoding

In the stream encoding geometry, the decoding process operates on the received C-Stream and is performed on a per-block basis.

The receiver processes the C-Stream as a sequence of encoded blocks, each containing "n" symbols. For each encoded block, the decoding process interprets the received symbols according to the selected FEC scheme. Based on this interpretation, the decoding process determines which source symbols are available, which are missing, and how the available symbols can be used to recover missing information.

The decoding process then applies the FEC decoding algorithm to recover any missing source symbols within the block.

Once all source symbols of an encoded block have been recovered, any symbols that are not required for reconstructing the original data are discarded. The recovered source symbols are then placed in their original order within the stream.

After all encoded blocks have been processed, the sequence of recovered source symbols is reassembled to reconstruct the SCHC Packet.

2.3. Fragmentation and Defragmentation Processes

The fragmentation and defragmentation processes are responsible for transporting the FEC-encoded data from the sender to the receiver. Unlike the other fragmentation processes defined in [RFC8724], the fragmentation and defragmentation processes of ARQ-FEC mode 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 the coded data structure 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 encoded 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 receives the tiles that are then passed to the assembler sub-process. The assembler subprocess constructs the coded data structure using the received tiles. Once the assembler

sub-process has collected sufficient symbols to enable decoding, the transporter sub-process terminates the fragmentation session and forwards the coded data structure to the decoding process.

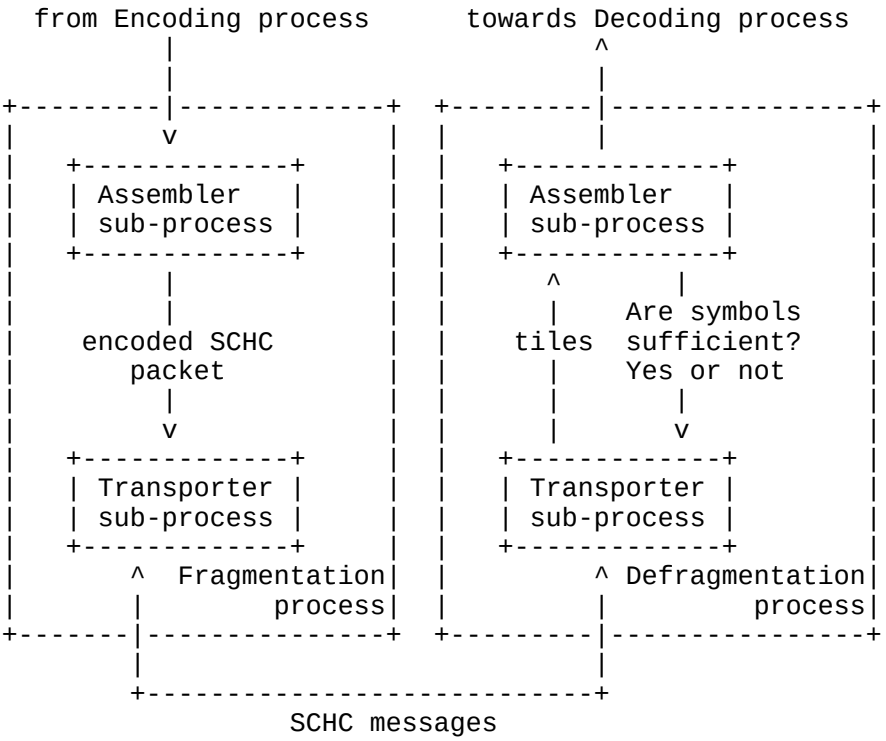


Figure 4: Fragmentation and Defragmentation process and its subprocesses.

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 coded data structure into an encoded SCHC packet. To do this, the assembler sub-process reads the encoded symbols from the coded data structure according to the selected encoding geometry and arranges them into a sequential buffer. This buffer forms the encoded SCHC packet that is passed to the transporter sub-process. The arrangement of symbols in the encoded SCHC packet is determined by the encoding geometry.

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

2.3.1.1.1. C-Matrix Assembly

In the matrix-based encoding geometry, the assembler sub-process constructs the encoded SCHC packet column by column. It first selects the S symbols of the first column of the C-matrix, which form the beginning of the encoded SCHC packet. It then selects the S symbols of the second column, which are appended to the encoded SCHC packet, and continues this process for all columns. Figure 5 shows how the encoded SCHC packet is constructed from the C-matrix.

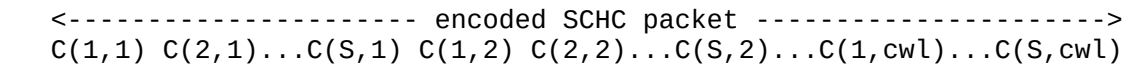


Figure 5: Symbols of a encoded SCHC packet.

2.3.1.1.2. C-Stream Assembly

In the stream encoding geometry, the assembler sub-process constructs the encoded SCHC packet from the C-Stream.

The assembler sub-process reads the encoded symbols sequentially from the C-Stream and arranges them into a continuous buffer. This buffer is then partitioned into tiles according to the tile size defined by the fragmentation rule. The tiles are grouped into windows of a priori known size. The mapping of symbols to tiles and windows is determined by the order in which symbols are read from the C-Stream.

The assembler sub-process MAY apply an interleaving pattern to the encoded symbols, as specified by the fragmentation rule, resulting in a reordering of the window and tile numbering sequence. This reordering is used to distribute symbols originating from different encoded blocks across multiple windows, improving robustness to losses. When such reordering is applied, the relative position of symbols within each encoded block is preserved, while their placement within the encoded SCHC packet is modified according to a deterministic pattern known to both sender and receiver.

The resulting sequence of tiles forms the encoded SCHC packet, which is passed to the transporter sub-process for transmission.

2.3.1.2. Receiver Behavior

In the receiver, the assembler sub-process is responsible for three tasks: placing each tile in the coded data structure, verifying whether there are enough symbols to successfully decode the SCHC Packet, and indicating to the transporter sub-process which tiles need to be retransmitted.

2.3.1.2.1. Placing Tiles in the Coded Data Structure

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 coded data structure.

2.3.1.2.1.1. C-Matrix Tile Placement

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. Each subsequent symbol of the tile is placed in the same column and the next row in increasing order. If the last row of the matrix is reached and unplaced symbols remain in the tile, placement shall continue at the first row of the next column. For example, assume a tile contains 4 symbols and the C-matrix has 7 rows (S = 7). If placement begins at row 5, column 1, the symbols are placed at positions (5,1), (6,1), (7,1), and (1,2), respectively.

2.3.1.2.1.2. C-Stream Tile Placement

When operating on a C-Stream, the assembler sub-process places received tiles into the sequential block structure of the C-Stream.

For each received tile, the assembler sub-process determines its position in the C-Stream based on the Window (W) and Tile (T) fields carried in the SCHC fragment. The tile is then stored at the corresponding position in the structure.

The placement of tiles in the C-Stream is independent of the order in which they are received. In particular, when interleaving has been applied at the sender, tiles may be received in an order that does not correspond to their original sequence in the C-Stream. By placing each tile according to its W and T values, the assembler sub-process reconstructs the original ordering of the encoded symbols within the C-Stream.

Missing tiles remain uninitialized or marked absent in the C-Stream; if the C-Stream as a whole becomes decodable, it is passed to the decoding process even if some tiles were never received.

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 Encoded Block to recover the original k symbols. Therefore, the assembler process only needs to have k encoded symbols in each Encoded Block to pass the coded data structure to the decoding process. If the coded data structure 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 coded data structure, as explained in section Section 2.3.1.2.1. Once the tiles are in place, the task of checking whether there are enough symbols in each Encoded Block can begin.

The verification task must review all encoded blocks in the coded data structure and check its present symbol count; if a block contains at least k symbols it is marked as decodeable, otherwise, not decodeable. If a block is already marked as decodeable, the task skips it and continues until all encoded blocks have been reviewed.

If at any point the verification task detects that all encoded blocks of the coded data structure are decodable, then the assembler sub-process MUST indicate to the transporter sub-process that the coded data structure 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 coded data structure. When the assembler sub-process adds the residual fragmentation bits, two outcomes are possible:

* If all blocks are marked as decodeable, the assembler sub-process must notify the signaling sub-process that the coded data structure contains enough symbols and the fragmentation session may end.

* If one or more encoded blocks within the coded data structure were marked as undecodable, the assembler sub-process must follow the retransmission selection steps for the chosen encoding geometry.

2.3.1.2.4. C-Matrix Tile Retransmission Selection

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.

2.3.1.2.5. C-Stream Tile Retransmission Selection

When the assembler process determines that an encoded block cannot be successfully decoded due to missing data, the assembler sub-process identifies the corresponding missing tiles within the C-Stream.

The identification of missing tiles is based on the position of tiles within the C-Stream, as determined by their Window (W) and Tile (T) values. This identification is independent of whether interleaving has been applied at the sender, as each tile retains a unique position within the C-Stream.

The assembler sub-process indicates the set of missing tiles to the transporter sub-process using the bitmap format defined for SCHC Compound ACK messages.

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 reassembler 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 in ARQ-FEC mode is associated with successful decoding of the coded data structure. If some tiles are lost during fragmentation and reassembly, the coded data 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 received data, or at the end of the transmission if there are insufficient symbols. For this reason, the SCHC Compound ACK MUST NOT be sent after each window. The sender may advance to subsequent windows even before confirming that all tiles from 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 (matrix-based geometry),

* if the receiver receives a SCHC regular fragment message and the assembler sub-process verify that there are enough symbols to decode the coded data structure,

* if the receiver receives a SCHC All-1 message and the assembler sub-process verify that there are enough symbols to decode the coded data structure,

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

The fragmented encoded SCHC Packet transmission concludes when:

* the receiver verify that there are enough symbols to decode the coded data structure.

* 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 encoded geometry choice, * the value of M (Symbol size in bits), * the value of N (Encoded block size), * 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.

The last tile 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 recover the SCHC packet. 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.

2.3.2.1.1. Matrix-Specific Behavior

In the matrix-based encoding geometry, 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.

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:

The behavior of the transporter sub-process depends on the type of SCHC fragment received and the decoding status as indicated by the assembler sub-process.

When a Regular SCHC Fragment is received, the transporter sub-process extracts the tiles based on the a priori known tile size and delivers them to the assembler sub-process. The transporter sub-process then awaits a response from the assembler sub-process. If the assembler sub-process indicates that sufficient symbols have been collected to enable decoding, 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 fragment. Otherwise, the transporter sub-process SHOULD take no action.

When an All-1 SCHC fragment is received, the transporter sub-process delivers the remaining tile(s) to the assembler sub-process and awaits a response. If the assembler sub-process indicates that sufficient symbols have been collected to enable decoding, 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 terminate the

fragmentation session. Otherwise, the transporter sub-process MUST send a SCHC Compound ACK with the C field set to 0, including bitmaps indicating the missing tiles.

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.

2.3.2.2.1. Matrix-Specific Behavior

When the first SCHC fragment of a fragmentation session is received, the first tile carries the S parameter. This parameter is extracted by the transporter sub-process and provided to the assembler sub-process.

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.

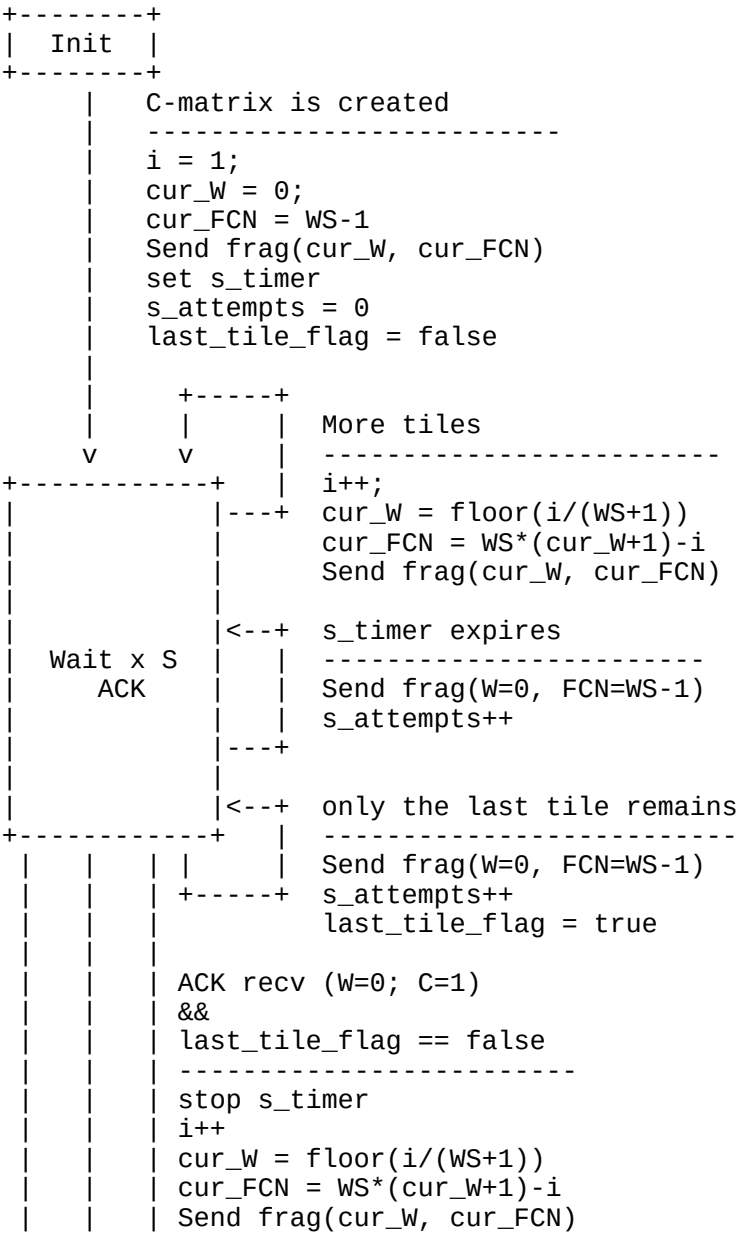
4. Security Considerations

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




Send
More tiles
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
Attempts = 0;
set rtx_timer
Wait x session ACK
rtx_timer expired
Send All-1
Attempts++;
set rtx_timer
ACK recv (C=0, bitmap)
Send missing tiles;
Resend Missing Tiles
ACK recv (W=3, C=1)
stop rtx_timer
ACK recv (W=3, C=1)
stop rtx_timer
End
(D) s_timer expired
&&
s_attempts > MAX_ACK_REQUEST

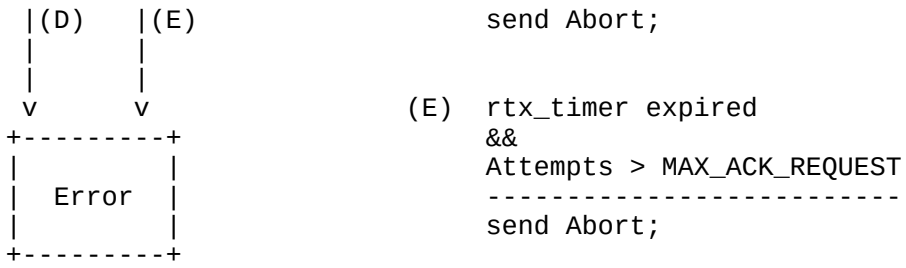
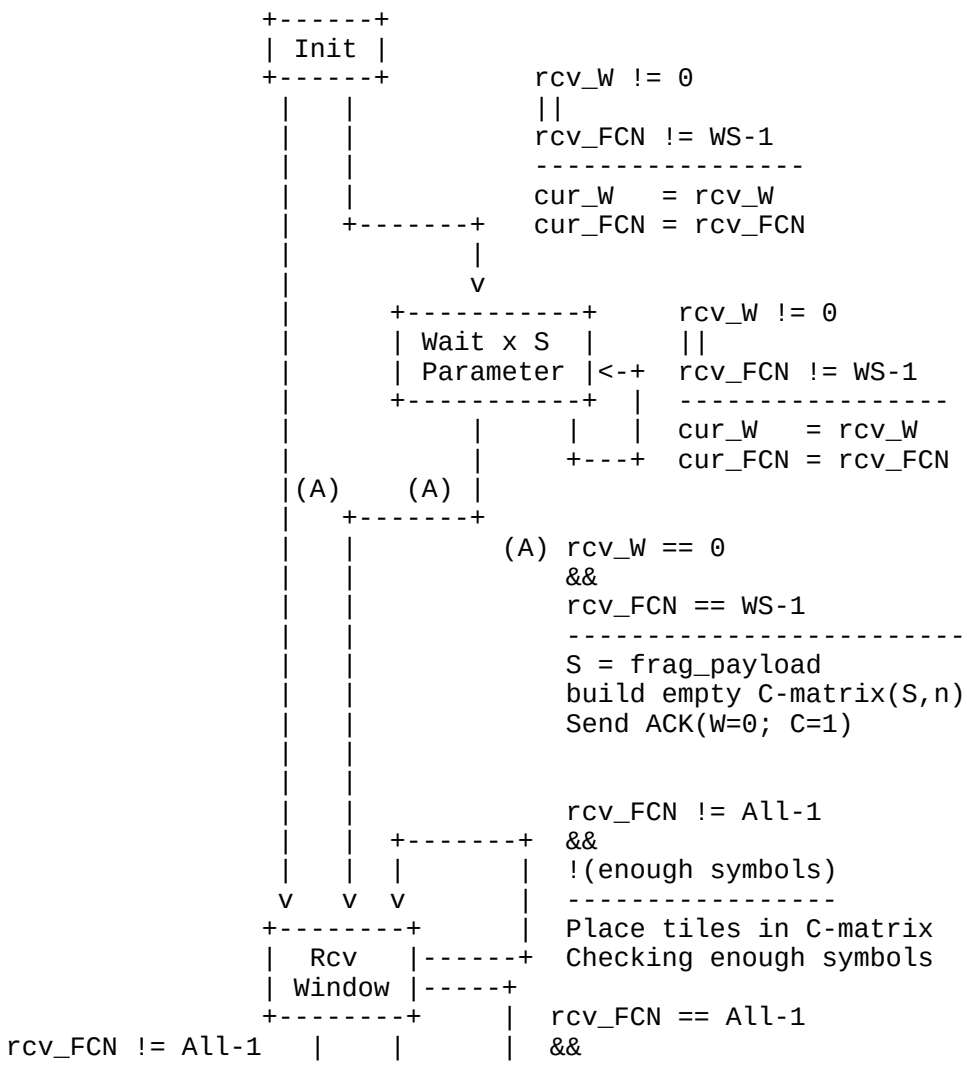


Figure 6: Sender State Machine for the ARQ-FEC Mode.



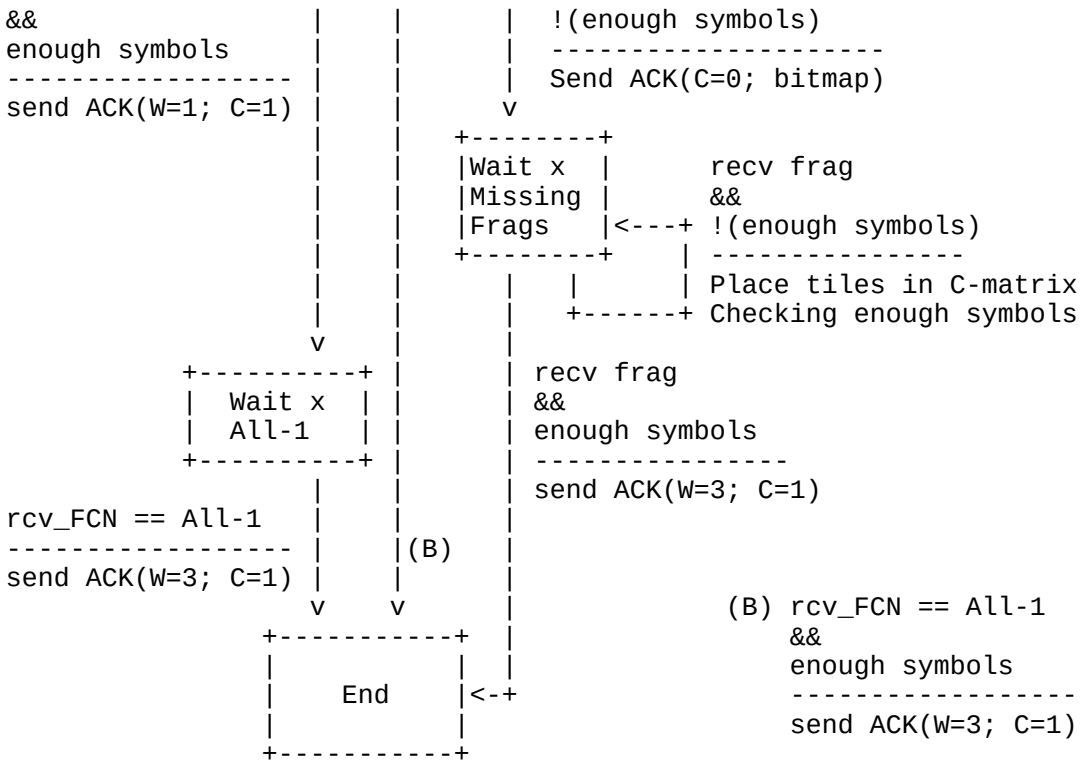


Figure 7: Receiver State Machine for the ARQ-FEC Mode.

7. Appendix B. Example of ARQ-FEC Fragmentation Mode with Matrix-Based Encoding Geometry

This example represents the transmission of a SCHC packet generated by the compression sublayer, for the matrix-based encoding geometry. 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.

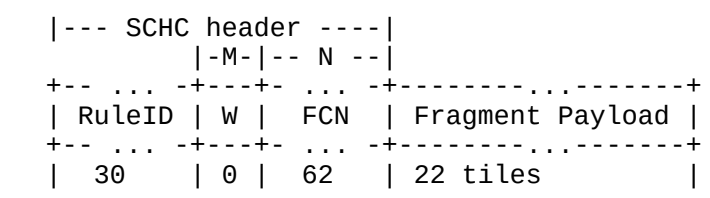


Figure 8: First regular SCHC fragment. Case 1: without loss and varible MTU.

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

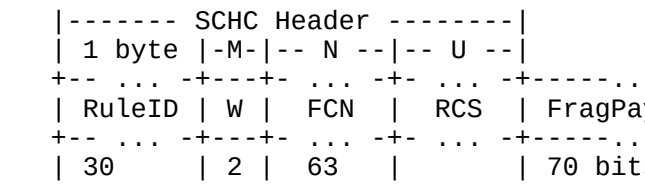


Figure 9: All-1 SCHC fragment. Case 1: without loss and varible MTU.

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

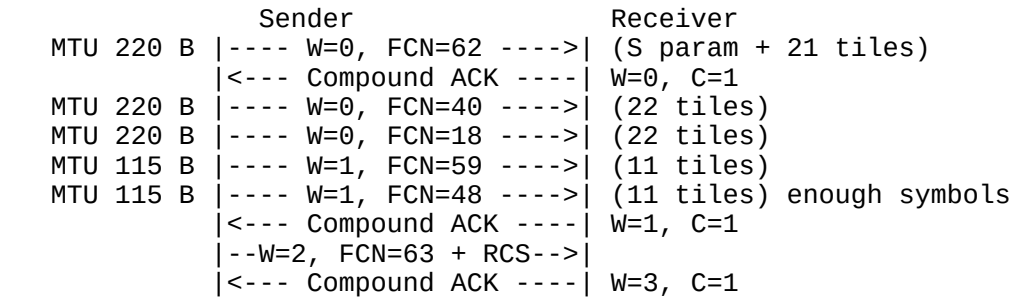


Figure 10: Message flow for ARQ-FEC mode (Case 1).

7.2.2. 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.

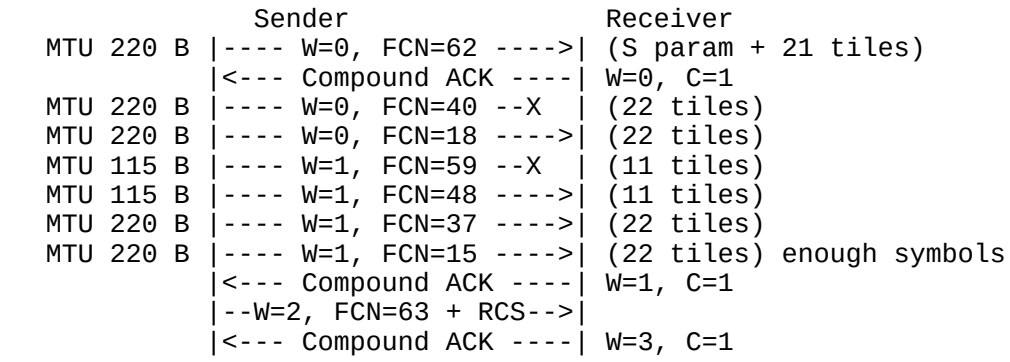


Figure 11: Message flow for ARQ-FEC mode (Case 2).

7.2.3. 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.

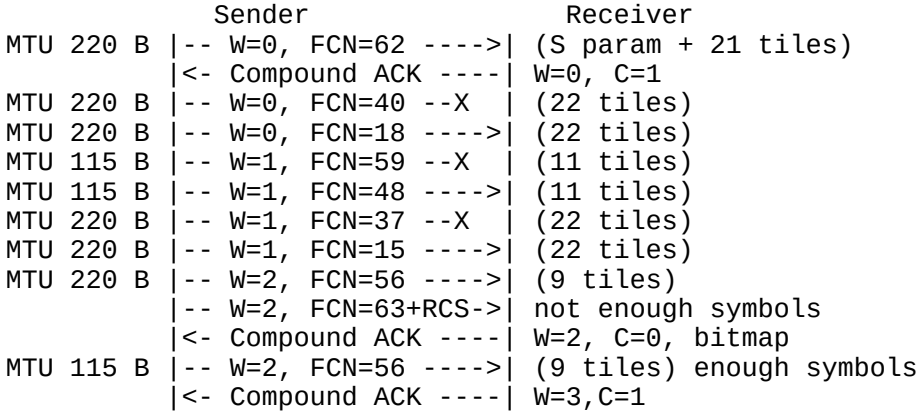


Figure 12: Message flow for ARQ-FEC mode (Case 3).

8. Appendix C. Example of ARQ-FEC Fragmentation Mode with Stream Encoding Geometry

This appendix gives a worked example of ARQ-FEC fragmentation using the C-Stream encoding geometry with XOR parity encoding.

8.1. Parameters

* Encoding geometry: C-Stream.
* Tile size: one symbol per tile.
* Window size: 7 tiles.
* FEC: simple XOR parity with source block size 2, encoded block size of 3 (2 data symbols + 1 parity).
* Interleaving: depth 3 (encoded block size).
* Disable All-1 payload: yes.

8.2. Sender

8.2.1. Encoding

1. Divide the SCHC Packet into symbols (each symbol is represented by a single letter):

abcdefghijklmnopqrstuvwxyzABCDEFGHIJ

Figure 13: SCHC Packet data partitioned in symbols.

1. Apply XOR sequentially every two source symbols (source blocks) to produce a third parity symbol. This will produce the C-Stream with encoded blocks of 3 symbols each.

```
ab+cd+ef+gh+ij+kl+mn+op+qr+st+uv+wx+yz+AB+CD+EF+GH+IJ+
```

Figure 14: C-Stream after XOR encoding of each block.

8.2.2. Fragmentation

8.2.2.1. Assembler Sub-Process

1. Divide the C-Stream into tiles and group them into windows. Tile size is set to match the symbol size.

```
C-Stream: ab+cd+e f+gh+ij +kl+mn+ op+qr+s t+uv+wx +yz+AB+ CD+EF+G H+IJ+
Tiles:    6543210 6543210 6543210 6543210 6543210 6543210 6543210 65432
Windows:  0000000 1111111 2222222 3333333 4444444 5555555 6666666 77777
```

Figure 15: Tiles are numbered per normal tile numeration.

1. Apply interleaving (depth 3). Because the interleaving depth equals the encoded block size, this produces a simple, fixed permutation: take the 1st symbol from each block in order, then 2nd symbols, then 3rd symbols, producing the reordered C-Stream for fragmentation (i.e., the encoded SCHC Packet).

```
Window: 000 112 233 344 556 667 001 112 233 444 556 677 001 122 233 445 556 677
Tile:   630 415 263 041 526 304 526 304 152 630 415 263 415 263 041 526 304 152
Data:   ace gik moq suw yAC EGI bdf hjl npr tvx zBD FHJ +++ +++ +++ +++ +++ +++
```

Figure 16: Encoded SCHC Packet after interleaving.

8.2.2.2. Transporter Sub-Process

1. Partition the encoded SCHC Packet into fragments that each fit within the L2 MTU; for example, assume each fragment carries 9 tiles.

```
Window:   000112233 344556667 001112233 444556677 001122233 445556677
Tile:     630415263 041526304 526304152 630415263 415263041 526304152
Data:     acegikmoq suwyACEGI bdfhjlnpr tvxzBDFHJ +++++++++ +++++++++
Fragment: |0:6____| |3:0____| |0:5____| |4:6____| |0:4____| |4:5____|
```

Figure 17: Fragmented encoded SCHC Packet, ready to transmit.

1. Transmit each fragment in sequence, along with an All-1 fragment carrying the RCS data.

8.3. Receiver

8.3.1. Defragmentation

8.3.1.1. Transporter Sub-Process

1. Process the received fragments according to the SCHC header (identify Rule ID, begin the fragmentation session).

2. Extract payload and pass the received tiles to the Assembler sub-process. For this example, assume the second fragment was lost during transmission.

```
Window:   000112233 001112233 444556677 001122233 445556677 7~~~~~
Tile:     630415263 526304152 630415263 415263041 526304152 7~~~~~
Data:     acegikmoq bdfhjlnpr tvxzBDFHJ +++++++++ +++++++++ RCS~~~
Fragment: |0:6____| |0:5____| |4:6____| |0:4____| |4:5____| |7:7~~
```

Figure 18: Received SCHC fragments.

1. Await the signal form the Assembler sub-process before transmitting the SCHC Compound Ack back to the sender.

8.3.1.2. Assembler Sub-Process

1. Place any received tiles into an empty C-Stream while the fragmentation session is active. Tiles are placed in their corresponding order according to the interleaving parameter.

```
   C-Stream wth Window:Tile numbered slots (observe as grid):
   W: 000000011111111222222233333334444444555555566666667777 7...
   T: 654321065432106543210654321065432106543210654321065432...
   =: ==================================================
   W: 0  0  0  1  1  2  2  3  3    <- 1st fragment received
   T: 6  3  0  4  1  5  2  6  3    <- W:T numbering
   D: a  c  e  g  i  k  m  o  q    <- Data

       0  0  1  1  1  2  2  3  3    <- 2nd fragment received
       5  2  6  3  0  4  1  5  2    <
       b  d  f  h  j  l  n  p  r    <

      3rd fragment received ->    4  4  4  5  5  6  6  7  7
                                 >  6  3  0  4  1  5  2  6  3
                                 >  t  v  x  z  B  D  F  H  J

        0  0  1  1  2  2  2  3  3    <- 4th fragment received
        4  1  5  2  6  3  0  4  1    <
        +  +  +  +  +  +  +  +  +    <

       5th fragment received ->    4  4  5  5  5  6  6  7  7
                                  >  5  2  6  3  0  4  1  5  7
                                  >  +  +  +  +  +  +  +  +  +
   =: ==================================================
   C-Stream (underscores mark missing data):
   =: ab+cd+ef+gh+ij+kl+mn+op+qr+_t+_v+_x+_z+_B+_D+_F+_H+_J+
```

Figure 19: Received tiles placed in the C-Stream.

1. Check if there are enough symbols to decode missing data within every encoded block. In this example, the XOR-encoded blocks require 2 out of 3 symbols.

```
C-Stream: ab+ cd+ ef+ gh+ ij+ kl+ mn+ op+ qr+ _t+ _v+ _x+ _z+ _B+ _D+ _F+ _H+ _J+
Bitmap:   111 111 111 111 111 111 111 111 111 011 011 011 011 011 011 011 011 011
2 of 3:    OK  OK  OK  OK  OK  OK  OK  OK  OK  OK  OK  OK  OK  OK  OK  OK  OK  OK
```

Figure 20: Encoded blocks are marked as decodeable or non-decodeable.

1. In this case, all blocks are marked as decodeable, even if there was missing data. The Assembler sub-process signals the Transporter sub-process that the encoded SCHC packet is fully decodeable, triggering the SCHC Compound Ack with no tiles to retransmit.

8.3.2. Decoding

The XOR FEC scheme may be applied on each encoded block of the C-Stream to recover any missing data. When no data is missing, parity symbols are ignored. If a symbol is missing, the available symbols are combined using XOR to recover the missing data.

```
C-Stream:    ab+ cd+ ef+ gh+ ij+ kl+ mn+ op+ qr+ _t+   _v+   _x+   _z+   _B+   _D+   _F+
  _H+   _J+
Action:        i   i   i   i   i   i   i   i   i =xx   =xx   =xx   =xx   =xx   =xx   =xx
  =xx   =xx
Operation:                                       t^+=s v^+=u x^+=w z^+=y B^+=A D^+=C F^+=
E H^+=G J^+=I
SCHC Packet: ab  cd  ef  gh  ij  kl  mk  op  qr  st    uv    wx    yz    AB    CD    EF
  GH    IJ
```

*Legend: 'i'=ignore; '='=calculate; 'x'=combine

Figure 21: Recovering the SCHC Packet using available symbols.

9. 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.
* Version 02
  - Introduces encoding geometries and coded data structure abstraction.
  - Adds C-Stream alternative for simpler FEC schemes (e.g. XOR).
  - Minor wording and phrase improvements.

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炭单iga, J., Gomez, C., Aguilar, S., Toutain, L., C迦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

Javier Alejandro Fernandez
IMT Atlantique
Cesson-Sevigne
France
Email: javier-alejandro.fernandez@imt-atlantique.fr