INTERNATIONAL STANDARD

ISO/IEC 23008-10

First edition
2015-04-15

Information technology — High efficiency coding and media delivery in heterogeneous environments —

Part 10: MPEG Media Transport Forward Error Correction (FEC) codes

Technologies de l'information — Codage à haute efficacité et livraison des medias dans des environnements hétérogènes —

Partie 10: Codes de correction d'erreur anticipée pour le transport des medias MPEG

Reference number
ISO/IEC 23008-10:2015(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword ..... **iv**

Introduction ..... **v**

1 Scope **1**

2 Normative references **1**

3 Terms, definitions, symbols, and abbreviated terms **1**
3.1 Terms and definitions 1
3.2 Symbols and abbreviated terms 2
3.3 Conventions 2

4 Overview **2**

5 FEC Code Points **3**

6 Specification for Reed-Solomon Codes **3**
6.1 Introduction 3
6.2 Generator matrix 4

7 Specification for Structured Low-Density Parity-Check (S-LDPC) Codes **4**
7.1 Introduction 4
7.2 Structured LDPC Codes 5
7.3 Creating Parity-Check Matrix 5
7.4 Encoding Algorithm 6
7.5 Decoding Algorithm 7
7.6 Base matrix 7

8 Specification 6330 code and RaptorQ LA code **11**
8.1 Introduction 11
8.2 6330 code 12
8.3 RaptorQ LA 12
8.3.1 RaptorQ LA First Encoding Step Intermediate symbol generation 13
8.3.2 Layer-Aware RaptorQ Second Encoding Step 15
8.3.3 Layer-Aware RaptorQ Decoding 15

9 Specification for FireFort Low Density Generate Matrix (FireFort-LDGM) codes **16**
9.1 Introduction 16
9.2 FireFort Low Density Generator Matrix (FireFort-LDGM) Codes 16
9.2.1 Definition 16
9.2.2 FF-LDGM-Specific Elements 17
9.2.3 Parity Check Matrix of FF-LDGM Scheme 18
9.2.4 Creating a Sparse Matrix 18
9.2.5 Creating a Punctured Sparse Matrix 22
9.2.6 Source symbol division scheme 24
9.2.7 Structured interleaving and de-interleaving scheme 25
9.2.8 FF-LDGM code Encoding Algorithm 25
9.2.9 FF-LDGM code Decoding Algorithm 26
9.3 Layer-Aware FireFort-LDGM (LA FireFort-LDGM) Codes 26
9.3.1 Specification of the LA FF-LDGM Scheme 26
9.3.2 Encoding Algorithm 27
9.3.3 Decoding Algorithm 27

10 Specification for FEC code algorithms in SMPTE 2022-1 ..... **27**

Bibliography **28**

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 23008 consists of the following parts, under the general title *Information technology — High efficiency coding and media delivery in heterogeneous environments*:

— *Part 1: MPEG media transport (MMT)*

— *Part 2: High efficiency video coding (HEVC)*

— *Part 3: 3D Audio*

— *Part 10: MPEG Media Transport Forward Error Correction (FEC) codes*

— *Part 11: MPEG Media Transport Composition Information*

Introduction

This part of ISO/IEC 23008 specifies application level forward error correction (FEC) codes which can be used with application level-forward error correction (AL-FEC) framework of ISO/IEC 23008-1 MPEG Media Transport (MMT) to provide reliable delivery in IP network and non IP network environments that are prone to packet losses.

Information technology — High efficiency coding and media delivery in heterogeneous environments —

Part 10: MPEG Media Transport Forward Error Correction (FEC) codes

1 Scope

This part of ISO/IEC 23008 specifies application level forward error correction (FEC) codes which can be used with AL-FEC framework of ISO/IEC 23008-1 MPEG Media Transport to provide reliable delivery in IP network and non IP network environments that are prone to packet losses.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 23008-1, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 1: MPEG media transport (MMT)*

IETF RFC 5170, *Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes, June 2008*

IETF RFC 5510, *Reed-Solomon Forward Error Correction (FEC) Schemes, April 2009*

IETF RFC 6330, *RaptorQ Forward Error Correction Scheme for Object Delivery, August 2011*

SMPTE2022-1, *Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks*

3 Terms, definitions, symbols, and abbreviated terms

For the purposes of this document, the following terms and definitions apply.

3.1 Terms and definitions

3.1.1
code rate
ratio between the number of source symbols and the number of encoding symbols

3.1.2
encoding symbol
unit of data generated by the encoding process

3.1.3
encoding symbol block
set of encoding symbols from the encoding process of a source symbol block

3.1.4
3FEC code
algorithm for encoding data such that the encoded data flow is resilient to data loss

3.1.5
FEC payload ID
identifier that identifies the contents of a MMT packet with respect to the MMT FEC scheme

3.1.6
repair symbol
encoding symbol that is not a source symbol

3.1.7
repair symbol block
set of repair symbols which can be used to recover lost source symbols

3.1.8
source symbol
unit of data used during the encoding process

3.1.9
source symbol block
set of source symbols which is used to generate repair symbol block by FEC code

3.1.10
systematic code
any error correction code in which the source symbols are part of output encoded symbols

3.2 Symbols and abbreviated terms

For the purpose of this document, the symbols and abbreviated terms given below apply.

AL-FEC application layer (level) forward error correction

FEC forward error correction

LA layer aware

LA-FEC layer aware forward error correction

LDGM low density generator matrix

LDPC low density parity check

MMT MPEG media transport

RS Reed-Solomon

S-LDPC structured low density parity check

3.3 Conventions

The following conventions apply in this document:

— The Big Endian number representation scheme is used.

4 Overview

This part of ISO/IEC 23008 specifies application level forward error correction (FEC) codes. All codes specified in this part are systematic codes.

This specification defines six FEC code algorithms which each FEC code algorithm shall generate a repair symbol block from a source symbol block as shown in Figure 1. The source symbol block consists

of K source symbols of size T (in bytes) and the repair symbol block consists of P repair symbols of size T (in bytes).

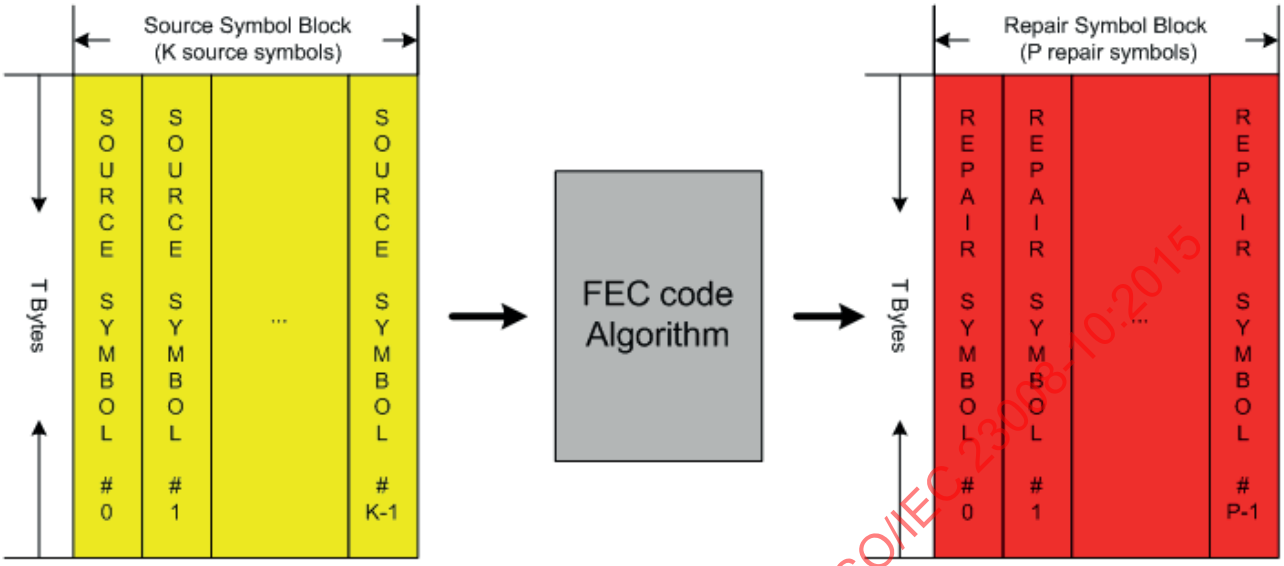

Figure 1 — Input and Output of FEC code

5 FEC Code Points

Table 1 specifies the code points for the FEC code algorithms specified in this part of ISO/IEC 23008. The FEC code algorithms themselves are specified in Clauses 6 to 10.

Table 1 — FEC Code Algorithms and Its Code Point

Code Point	FEC Code Algorithm
0	Reserved for ISO use
1	RS code (sub-Clause 6)
2	S_LDPC code (sub-Clause 7)
3	6330 code (sub-Clause 8.2)
4	RaptorQ LA code (sub-Clause 8.3)
5	FireFort-LDGM code (sub-Clause 9)
6	FEC code algorithm in SMPTE 2022-1 (sub-Clause 10)
7 ~ 255	Reserved for ISO use

NOTE When one of the FEC code algorithms specified in this specification is used for MMT AL-FEC framework, fec_code_id_for_repair_flow field as defined in ISO/IEC 23008-1:2014 Annex C.6 is set to its corresponding code point as specified in Table 1.

6 Specification for Reed-Solomon Codes

6.1 Introduction

In this clause, the following notations are used.

— *K*: number of source symbols in a source symbol block

— *P*: number of repair symbols in a repair symbol block

— $\boldsymbol{G} = [\boldsymbol{I}; \boldsymbol{A}]$: a systematic generator matrix for $[K+P, K]$-RS code where $\boldsymbol{I}$ is the identity matrix of order K and $\boldsymbol{A}$ is a $K \times P$ matrix.

A (N, K) Reed–Solomon code is a linear block code of length N (over Galois Field F) with dimension K and minimum Hamming distance $N - K + 1$. The Reed–Solomon code is optimal in the sense that the minimum distance has the maximum value possible for a linear code of size (N, K); this is known as the Singleton bound. Such a code is also called a maximum distance separable (MDS) code.

The clause 8 of IETF RFC5510 gives full specification of the RS code for the erasure channel and especially, the clause 8.2.1 of IETF RFC5510 gives encoding principle for RS encoding algorithm. The generate matrix $\boldsymbol{G}$ perfectly characterizes the RS code. In this specification, it specifies only the case when $m = 8$ (over GF(2^8)) with the generator matrix given in the sub-clause 6.2. Therefore, an encoding symbol block shall be generated from a source symbol block by the given generator matrix in the sub-clause 6.2 and this FEC code shall output the repair symbol block of the encoding symbol block.

6.2 Generator matrix

The generator matrix $\boldsymbol{G}$ has the form $\boldsymbol{G} = [\boldsymbol{I}; \boldsymbol{A}]$ where $\boldsymbol{I}$ is an identity matrix of size K and $\boldsymbol{A}$ is a $K \times P$ matrix, $(K + P) \leq 255$. Let α be the root of the polynomial $1 + x^2 + x^3 + x^4 + x^8$ which is the primitive polynomial of degree 8 given in sub-clause 8.1 of IETF RFC5510. The non-zero elements of the finite filed GF(2^8) are generated by a primitive element α and the elements of GF(2^8) are represented by bytes (group of 8 bits), using the polynomial base representation, with (α^7, α^6, α^5, α^4, α^3, α^2, α, 1) as a basis. The root α is thus represented as: $\alpha = 00000010$. For RS code specified in this specification, the matrix $\boldsymbol{A}$ is a Cauchy matrix which shall have entries

$$A_{i,j} = 1/(x_i + y_j) \text{ for } 0 \leq i < K \text{ and } 0 \leq j < P$$

where x_i and y_j are elements in GF(2^8) and are defined as:

$$x_i = \alpha^{254 - i}, \text{ and } y_j = \alpha j.$$

Therefore, the matrix $\boldsymbol{A}$ is given by

$$A = \begin{bmatrix} \frac{1}{x_0 + y_0} & \frac{1}{x_0 + y_1} & \cdots & \frac{1}{x_0 + y_{P-2}} & \frac{1}{x_0 + y_{P-1}} \\ \frac{1}{x_1 + y_0} & \frac{1}{x_1 + y_1} & \cdots & \frac{1}{x_1 + y_{P-2}} & \frac{1}{x_1 + y_{P-1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{x_{K-2} + y_0} & \frac{1}{x_{K-2} + y_1} & \cdots & \frac{1}{x_{K-2} + y_{P-2}} & \frac{1}{x_{K-2} + y_{P-1}} \\ \frac{1}{x_{K-1} + y_0} & \frac{1}{x_{K-1} + y_1} & \cdots & \frac{1}{x_{K-1} + y_{P-2}} & \frac{1}{x_{K-1} + y_{P-1}} \end{bmatrix}$$

NOTE Any submatrix of the Cauchy matrix is invertible.

7 Specification for Structured Low-Density Parity-Check (S-LDPC) Codes

7.1 Introduction

A Low-Density-Parity-Check (LDPC) code is a linear block code defined by its parity-check matrix. In this specification, we use a special case of LDPC codes, called structured LDPC (S-LDPC) codes, which have an efficient encoding algorithm and adopted as an FEC code in standardizations such as IEEE 802.16e and 801.11n.

In this document, we use the following notations.

— K: number of source symbols in a source symbol block

— K': number of source symbols in an extended source symbol block

— P: number of repair symbols in a repair symbol block

— P': number of repair symbols in an extended repair symbol block

— $S(i)$: the i-th source symbol in a source symbol block ($0 \le i < K$). It can be represented as a binary column vector of length $8T$ or a column vector of length T over GF(2^8)

— $R(i)$: the i-th repair symbols in a repair symbol block ($0 \le i < P$). It can be represented as a binary column vector of length $8T$ or a column vector of length T over GF(2^8)

— $\boldsymbol{H}$: a sparse parity-check matrix of an S-LDPC code.

7.2 Structured LDPC Codes

In this clause, S-LDPC codes are described by parity-check matrices which consist of circulant permutation matrices or the zero matrix.

Let $\boldsymbol{Q}$ be the $L \times L$ permutation matrix given by

$$\boldsymbol{Q} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Note that $\boldsymbol{Q}^i$ is just the circulant permutation matrix which shifts the identity matrix $\boldsymbol{I}$ to the right by i times for any integer i, $0 \le i < L$. For simple notation, $\boldsymbol{Q}^\infty$ denotes the zero matrix.

Let $\boldsymbol{H}$ be the P' by $K' + P'$ matrix defined by

$$\boldsymbol{H} = \begin{bmatrix} Q^{a_{0,0}} & Q^{a_{0,1}} & \cdots & Q^{a_{0,k+p-2}} & Q^{a_{0,k+p-1}} \\ Q^{a_{1,0}} & Q^{a_{1,1}} & \cdots & Q^{a_{1,k+p-2}} & Q^{a_{1,k+p-1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Q^{a_{p-1,0}} & Q^{a_{p-1,1}} & \cdots & Q^{a_{p-1,k+p-2}} & Q^{a_{p-1,k+p-1}} \end{bmatrix}$$

where p and k are given by $p = P' / L$ and $k = K' / L$, respectively and $a_{i,j} \in \{0, 1, \ldots, L-1, \infty\}$. If the locations of 1's in the first row of the i-th row block

$$\boldsymbol{H}_i = \begin{bmatrix} Q^{a_{i,0}} \cdots Q^{a_{i,k+p-1}} \end{bmatrix}$$

are fixed, then the locations of other 1's in $\boldsymbol{H}_i$ are uniquely determined.

7.3 Creating Parity-Check Matrix

For efficient encoding, it restricts the parity part of $\boldsymbol{H}$ to an almost lower triangular matrix with additional constraints as follows:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H_I} & \boldsymbol{H_P} \end{bmatrix} = \begin{bmatrix} & \boldsymbol{Q} & \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ & \vdots & \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{H_I} & \boldsymbol{I} & \vdots & \vdots & \cdots & \vdots & \vdots \\ & \vdots & \vdots & \vdots & \cdots & \boldsymbol{I} & \boldsymbol{0} \\ & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} & \boldsymbol{I} \\ & \boldsymbol{Q} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$$

In the first column block of $\boldsymbol{H_P}$, $\boldsymbol{I}$ is placed only at the ceil(p/2)-th row block , $\boldsymbol{H}_{\text{ceil}(p/2)-1}$, where ceil(x) is the smallest integer not less than x.

Let ***BM*** be a mother matrix having 400 column blocks and 20 row blocks with L = 16, i.e. the matrix ***BM*** has 6400 columns and 320 rows. Each column block and row block of ***BM*** has exactly 7 and 140 circulant permutation matrices of size 16, respectively. The remaining part of ***BM*** is filled with zero matrices of size 16. The i-th row block of ***BM*** can be represented as a sequence of pairs $(t_{i,j}, e_{i,j})$ where $t_{i,j}$ is the index of column block corresponding to the j-th circulant permutation matrix and $e_{i,j}$ is its exponent. The ***BM*** matrix supports various values of K and P with techniques called scaling down and row splitting. The resulting matrix is used as $\boldsymbol{H_I}$ for encoding process.

In order to support short source symbol blocks efficiently, the matrix ***BM*** is scaled down by a scaling factor S_1. The resulting matrix is composed of circulant permutation matrices and zero matrices of size $16/S_1$, i.e. it has $6400/S_1$ columns and $320/S_1$ rows. The scaling factor can be obtained as $S_1 = 2^a$ where a is the largest integer satisfying $K \le (400 \cdot 16) / 2^a$. Note that the resulting matrix can be represented as the sequence of pairs $(t_{i,j}, e_{i,j} \bmod (16/S_1))$ since the size of circulant permutation matrices and zero matrices are reduced from 16 to $16/S_1$.

As mentioned above, the matrix ***BM*** has $320/S_1$ rows after downscaling. It means that the number of repair symbols P is $320/S_1$ at maximum. To support larger values of P, we extend the matrix ***BM*** by splitting its rows. In this process, each row block is splitted into S_2 row blocks. For given repair symbol block length P and the scaling factor S_1, the row splitting factor S_2 can be obtained as $S_2 = \text{ceil}(P/(320/S_1))$.

The matrix $\boldsymbol{H_I}$ is obtained from the matrix ***BM*** as follows. Let $\boldsymbol{BM}_i = \{(t_{i,0}, e_{i,0}), (t_{i,1}, e_{i,1}),..., (t_{i,139}, e_{i,139})\}$ be the ordered sequence of pairs $(t_{i,j}, e_{i,j})$ representing the i-th row block of ***BM***. Let S_1 and S_2 be the scaling factor and the row splitting factor, respectively. They are determined uniquely by K and P. Then the $(S_2 \times i + j)$-th row block of $\boldsymbol{H_I}$ can be represented as follows:

$$\boldsymbol{T}_{(S2\times i)+j} = \{(t_{i,k}, e_{i,k} \bmod (16/S_1)) \mid k \bmod (S_2) = (S_2 - 1 - j),\ 0 \le k < 140\}$$

Note that the matrix $\boldsymbol{H_I}$ has 400 column blocks and S_2×20 row blocks with $L = 16/S_1$, i.e., it has $(6400/S_1)$ columns and $(S_2 \times 320/S_1)$ rows.

Finally, the parity-check matrix ***H*** is obtained by augmenting the matrix $\boldsymbol{H_P}$ with appropriate size to the matrix $\boldsymbol{H_I}$. H has 400 + S_2×20 column blocks and S_2×20 row blocks with $L = 16/S_1$, i.e. H consists of $(6400 + S_2 \cdot 320)/S_1$ columns and $(S_2 \cdot 320/S_1)$ rows.

7.4 Encoding Algorithm

The encoding of an S-LDPC code is performed based on the following $p' \times L'$ by $(k'+p') \times L'$ parity-check matrix:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H_I} & \boldsymbol{H_P} \end{bmatrix} = \begin{bmatrix} & \boldsymbol{Q} & \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ & \vdots & \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{H_I} & \boldsymbol{I} & \vdots & \vdots & \cdots & \vdots & \vdots \\ & \vdots & \vdots & \vdots & \cdots & \boldsymbol{I} & \boldsymbol{0} \\ & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} & \boldsymbol{I} \\ & \boldsymbol{Q} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$$

where $L' = 16/S_1$, $p' = S_2 \times 20$ and $k' = 400$. It has $P' = S_2 \times 320/S_1$ rows.

The ***H*** is divided into the form

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{E} \end{bmatrix}$$

where ***A*** is $(p' - 1) \times L$ by $k' \times L$, ***B*** is $(p' - 1) \times L$ by L, ***T*** is $(p' - 1) \times L$ by $(p' - 1) \times L$, ***C*** is L by $k' \times L$, ***D*** = ***Q*** is L by L and ***E*** is L by $(p' - 1) \times L$. Let $\boldsymbol{c} = (\boldsymbol{s}, \boldsymbol{r_1}, \boldsymbol{r_2})$ be a codeword specified by ***H***, that is $\boldsymbol{Hc}^T = \boldsymbol{0}^T$ where $\boldsymbol{s}$ is the systematic part, $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ are the parity parts which have length L and $(p' - 1) \times L$, respectively. That is $\boldsymbol{s}$

= [$S(0),\ldots, S(K'-1)$], $\boldsymbol{r_1}$ = [$R(0),\ldots, R(L-1)$] and $\boldsymbol{r_2}$ = [$R(L),\ldots, R(P'-1)$]. Note that $S(K), S(K+1), \ldots, S(K'-1)$ denote $K'-K$ padded source symbols, which are set to all zero bits and not delivered. Furthermore, $R(P),\ldots, R(P'-1)$ denote the repair symbols to be punctured, i.e. their values are calculated but not contained in the corresponding repair symbol block.

Then, the detailed operations for encoding of an S-LDPC code defined by $\boldsymbol{H}$ are as follows:

Therefore, when a source symbol block $\boldsymbol{s}$ is input, this FEC code shall output the repair symbol block [R(0),...,R(P-1)] of $\boldsymbol{c}$ which satisfies the equation $\boldsymbol{Hc}^T = \boldsymbol{0}^T$.

7.5 Decoding Algorithm

S-LDPC codes are one family of LDPC codes. Therefore any decoding algorithm for conventional LDPC codes can be applied without any modification. Consideration on the decoding algorithm of LDPC codes can be found in IETF RFC5170.

7.6 Base matrix

The S-LDPC codes can be fully described by the base matrix $\boldsymbol{BM}$ and algorithms to calculate the scaling factor and the row splitting factor. The i-th row block of $\boldsymbol{BM}$ can be represented as a sequence of pairs $T_i = \{(t_{i,0}, e_{i,0}), (t_{i,1}, e_{i,1}),\ldots, (t_{i,139}, e_{i,139})\}$ where $t_{i,j}$ is the index of column block corresponding to the j-th circulant permutation matrix and $e_{i,j}$ is its exponent for $0 \le j < 140$.

T_0 = {(1,8), (2,8), (3,10), (5,12), (6,8), (7,12), (9,8), (12,4), (14,12), (15,0), (19,0), (20,9), (26,4), (34,8), (35,1), (38,0), (45,13), (46,0), (48,13), (56,9), (57,3), (62,1), (63,8), (71,12), (75,8), (77,3), (78,2), (82,13), (83,13), (85,9), (88,1), (90,15), (92,4), (93,12), (97,0), (99,15), (104,5), (107,14), (110,13), (111,15), (116,9), (117,7), (120,9), (121,8), (125,15), (127,14), (128,15), (129,9), (131,5), (134,12), (150,12), (152,13), (156,1), (158,9), (159,13), (161,7), (163,5), (164,4), (165,13), (168,11), (171,9), (172,12), (175,12), (177,13), (180,5), (185,9), (193,1), (194,8), (195,9), (199,3), (200,9), (202,9), (204,3), (207,13), (212,13), (213,1), (214,13), (217,3), (223,13), (224,14), (226,10), (227,5), (228,7), (232,5), (236,14), (240,7), (241,7), (245,9), (250,12), (251,5), (255,5), (260,5), (267,4), (268,4), (272,4), (273,4), (275,9), (276,6), (278,5), (284,8), (288,1), (289,6), (291,2), (292,4), (297,12), (299,4), (302,4), (309,5), (310,4), (311,3), (312,5), (326,0), (330,10), (334,9), (335,4), (337,1), (338,6), (340,14), (342,10), (343,1), (347,4), (349,9), (350,1), (351,4), (357,14), (361,8), (364,0), (365,12), (367,6), (369,2), (373,4), (375,12), (376,12), (377,0), (379,0), (383,0), (384,1), (388,4), (389,0), (391,3)}

T_1 = {(2,4), (5,0), (8,9), (10,8), (12,8), (13,4), (14,8), (17,1), (23,12), (24,13), (29,9), (30,12), (33,9), (37,4), (45,0), (46,12), (47,8), (56,4), (60,0), (65,13), (73,13), (77,13), (78,12), (81,13), (89,12), (94,5), (99,5), (100,9), (102,9), (107,4), (111,5), (112,13), (117,4), (125,0), (127,12), (128,5), (133,12), (134,1), (136,1), (137,0), (138,12), (141,12), (143,4), (155,0), (157,12), (158,9), (160,0), (161,0), (163,8), (169,9), (170,12), (174,8), (176,4), (177,4), (178,8), (180,8), (182,8), (186,12), (187,8), (188,8), (189,8), (191,8), (192,1), (196,0), (198,8), (199,8), (200,0), (202,1), (204,4), (207,8), (210,1), (214,4), (217,2), (221,9), (224,0), (226,12), (228,9), (233,1), (236,8), (239,0), (241,0), (246,1), (249,1), (251,5), (256,8), (257,9), (259,9), (263,0), (264,1), (266,0), (267,1), (270,5), (271,9), (280,8), (282,0), (285,12), (286,1), (291,0), (292,8), (295,5), (302,12), (305,12), (306,9), (308,5), (309,4), (311,1), (312,13), (315,13), (316,1), (321,12), (322,5), (323,1), (324,5), (327,13), (328,5), (338,1), (342,12), (343,5), (346,13), (347,5), (349,0), (356,13), (361,1), (363,5), (367,5), (369,5), (372,5), (373,13), (374,4), (376,1), (380,4), (382,1), (387,5), (389,13), (390,5), (392,5), (393,1), (394,8), (395,9), (397,5)}

T_2 = {(1,2), (2,14), (3,3), (5,3), (6,10), (12,11), (13,3), (15,0), (19,13), (20,13), (21,6), (22,11), (33,10), (36,13), (38,6), (39,5), (40,12), (43,0), (44,3), (46,15), (47,3), (48,13), (51,7), (53,14), (57,15), (58,8), (59,6), (61,11), (70,7), (71,10), (73,10), (74,14), (79,1), (85,7), (86,14), (88,5), (89,5), (90,2), (92,10), (95,6), (99,14), (103,4), (104,6), (105,6), (111,10), (115,8), (128,3), (130,12), (133,4), (136,2), (139,6), (142,11), (145,14), (155,0), (156,14), (160,7), (168,5), (171,10), (176,12), (182,12), (183,8), (185,12), (186,9), (191,6), (193,15), (199,10), (204,0), (205,8), (207,5), (213,1), (217,4), (224,0), (226,0), (228,0), (230,4), (233,0), (234,12), (236,1), (238,8), (239,1), (240,3), (246,13), (247,12), (248,1), (249,8), (250,4), (251,1), (253,9), (254,0), (258,5), (262,3), (264,7), (268,6), (270,4), (271,1), (274,8), (277,5), (280,2), (286,11), (287,1), (293,3), (294,12), (296,7), (297,5), (299,7), (301,7), (302,11), (304,11), (306,3), (308,11), (309,15), (314,15), (315,8), (316,14), (317,10), (322,10), (325,11), (326,8), (327,15), (331,11), (333,1), (334,5),

(335,9), (345,0), (348,13), (351,9), (358,9), (360,1), (362,15), (363,3), (371,3), (375,9), (376,7), (378,2), (379,1), (387,1), (389,11), (394,13), (396,15), (399,13)}

T_3 = {(2,4), (3,15), (4,7), (7,13), (9,8), (14,6), (15,5), (18,3), (22,4), (29,12), (30,7), (32,4), (36,14), (40,5), (50,5), (53,1), (54,11), (55,8), (60,8), (64,15), (68,12), (70,5), (71,9), (75,14), (77,2), (81,5), (85,10), (90,8), (91,11), (95,9), (96,11), (100,8), (101,5), (103,0), (104,4), (105,15), (107,13), (108,13), (109,13), (110,5), (111,9), (113,9), (116,0), (121,12), (123,12), (124,13), (131,9), (132,11), (133,1), (136,12), (137,11), (140,9), (144,12), (145,9), (149,0), (152,6), (155,7), (159,5), (162,0), (164,12), (166,6), (167,1), (168,9), (171,12), (174,5), (176,0), (180,5), (181,6), (182,0), (183,10), (184,11), (188,1), (189,1), (190,9), (197,6), (199,15), (203,6), (209,11), (213,15), (215,6), (223,3), (232,0), (234,6), (235,2), (240,10), (244,2), (247,6), (248,0), (253,6), (254,2), (255,2), (256,0), (257,10), (260,7), (265,2), (266,4), (274,0), (278,0), (280,4), (281,2), (286,2), (288,10), (291,2), (292,10), (294,1), (295,4), (297,0), (299,15), (301,2), (304,2), (306,3), (314,7), (321,1), (322,11), (325,10), (327,0), (334,11), (336,6), (338,8), (346,3), (348,3), (351,9), (352,8), (353,1), (355,11), (364,10), (366,10), (369,15), (370,14), (371,15), (373,13), (374,6), (376,15), (377,14), (382,5), (386,3), (387,2), (389,7), (396,8), (399,10)}

T_4 = {(0,0), (1,5), (5,9), (16,9), (17,1), (18,0), (19,0), (21,1), (26,0), (28,9), (29,8), (31,1), (34,7), (36,1), (37,5), (39,0), (40,15), (44,13), (50,9), (55,0), (56,0), (58,7), (67,15), (72,11), (73,1), (74,11), (76,5), (78,3), (80,3), (81,2), (84,0), (87,8), (89,5), (93,7), (96,2), (99,7), (107,14), (108,2), (110,0), (114,3), (117,1), (118,10), (119,10), (120,15), (123,11), (125,11), (128,10), (132,0), (138,10), (140,3), (143,2), (147,2), (152,14), (155,13), (164,13), (167,6), (171,2), (173,11), (174,8), (175,10), (180,7), (181,4), (182,14), (188,10), (191,2), (195,10), (199,9), (200,3), (204,14), (206,15), (207,0), (208,12), (211,6), (212,7), (213,7), (217,0), (218,15), (222,4), (223,2), (228,12), (230,9), (231,12), (241,6), (242,14), (249,15), (253,6), (254,1), (255,8), (256,1), (257,6), (259,11), (261,7), (262,12), (264,7), (265,15), (268,8), (269,12), (275,15), (277,11), (278,15), (280,11), (285,3), (286,13), (289,11), (293,13), (295,0), (297,3), (298,4), (300,12), (307,11), (311,1), (315,1), (316,6), (318,5), (320,4), (324,5), (326,9), (338,8), (339,13), (342,13), (346,5), (347,13), (348,7), (356,4), (357,2), (358,13), (359,2), (360,5), (361,0), (363,1), (364,8), (365,0), (368,13), (371,3), (375,8), (385,1), (388,3), (389,0), (393,5), (398,11)}

T_5 = {(1,14), (6,9), (8,11), (12,3), (18,10), (19,14), (21,9), (24,15), (25,10), (31,14), (34,11), (35,10), (37,8), (38,6), (40,7), (41,9), (43,3), (44,4), (45,14), (49,7), (53,7), (59,5), (61,13), (65,0), (72,11), (76,10), (83,15), (84,5), (86,11), (88,12), (101,5), (105,14), (108,5), (109,6), (113,6), (114,14), (115,13), (117,3), (119,0), (121,8), (122,7), (124,4), (129,1), (131,14), (132,1), (135,9), (136,8), (137,10), (139,3), (141,8), (142,15), (144,0), (148,12), (151,7), (154,11), (159,12), (166,1), (169,15), (175,3), (176,0), (183,13), (184,1), (186,3), (193,13), (194,3), (195,4), (196,3), (197,4), (205,4), (209,15), (210,0), (214,5), (217,10), (218,14), (219,4), (222,3), (223,0), (225,5), (227,9), (228,9), (229,9), (232,5), (235,5), (236,3), (241,1), (243,8), (245,0), (247,4), (248,5), (250,0), (262,8), (263,8), (266,5), (267,1), (272,1), (274,10), (275,1), (278,15), (281,0), (282,8), (287,15), (288,14), (291,14), (294,10), (298,1), (300,11), (303,10), (308,2), (309,10), (313,8), (317,0), (319,0), (325,2), (327,1), (329,2), (332,4), (333,2), (335,2), (336,0), (338,11), (350,10), (351,4), (353,3), (354,5), (355,5), (356,10), (359,6), (360,1), (362,15), (363,2), (366,2), (368,14), (373,6), (376,10), (378,12), (379,6), (383,10), (385,7), (388,14), (398,15)}

T_6 = {(4,8), (6,3), (7,10), (11,15), (13,11), (15,3), (16,11), (22,5), (27,3), (28,12), (31,12), (33,1), (34,13), (36,13), (38,13), (42,8), (43,12), (44,1), (54,14), (55,11), (57,12), (69,9), (70,1), (71,9), (73,14), (75,9), (76,8), (78,10), (80,8), (81,1), (84,9), (86,10), (87,13), (88,13), (93,0), (95,11), (101,0), (102,1), (103,1), (104,9), (107,1), (110,4), (111,1), (112,9), (114,9), (117,2), (119,1), (120,1), (122,8), (130,1), (131,1), (134,4), (135,10), (138,9), (139,15), (147,0), (149,3), (150,11), (153,2), (155,10), (165,13), (168,7), (170,11), (171,3), (173,8), (180,11), (185,3), (188,0), (193,10), (196,3), (198,1), (201,10), (203,11), (205,7), (207,14), (208,13), (211,4), (215,2), (216,7), (217,8), (220,3), (227,14), (229,5), (231,5), (233,14), (234,0), (237,6), (241,6), (242,6), (248,0), (258,6), (261,3), (262,14), (263,5), (264,2), (266,1), (268,12), (269,6), (273,14), (279,2), (283,14), (287,15), (288,0), (290,6), (292,2), (294,1), (295,10), (296,4), (300,4), (302,11), (308,12), (313,10), (317,6), (321,14), (325,1), (326,0), (332,10), (336,4), (341,10), (342,7), (343,7), (355,13), (356,1), (358,11), (365,7), (369,9), (370,3), (371,3), (377,2), (385,2), (387,4), (388,2), (390,7), (391,8), (392,5), (395,2), (396,1), (397,5), (398,7), (399,3)}

T_7 = {(0,15), (1,1), (3,6), (4,8), (10,9), (15,15), (17,10), (18,14), (20,9), (22,8), (25,14), (27,14), (31,4), (33,10), (36,3), (38,14), (41,10), (45,11), (48,0), (52,14), (55,10), (59,3), (62,12), (63,14), (64,3), (66,14), (69,0), (74,6), (77,5), (80,2), (82,4), (84,0), (87,6), (99,8), (103,2), (107,13), (108,10), (110,1), (118,12), (121,2), (122,7), (125,3), (129,6), (131,7), (132,7), (137,10), (138,6), (139,1), (142,15), (145,7), (146,3),

(147,5), (148,11), (151,2), (153,9), (154,10), (157,0), (165,3), (168,0), (173,11), (175,15), (179,9), (183,8), (184,3), (185,1), (186,2), (187,2), (189,6), (194,9), (196,4), (203,11), (204,3), (205,11), (206,5), (209,1), (215,6), (219,1), (222,1), (224,0), (225,1), (226,1), (229,3), (231,7), (232,13), (234,0), (237,2), (238,2), (242,10), (243,11), (245,5), (247,1), (248,9), (250,2), (251,3), (253,0), (255,9), (256,4), (259,1), (262,9), (263,0), (267,3), (268,1), (269,0), (270,9), (271,8), (276,9), (281,9), (285,9), (286,10), (289,11), (290,8), (292,5), (293,10), (297,1), (306,0), (308,1), (309,1), (314,0), (317,3), (318,12), (320,13), (322,12), (324,12), (328,9), (336,12), (339,13), (340,7), (341,0), (346,15), (347,15), (348,11), (360,10), (367,0), (368,5), (372,11), (375,3), (378,6), (382,4), (390,6), (396,11)}

T_8 = {(2,8), (3,13), (8,11), (11,9), (16,14), (24,3), (25,1), (26,4), (28,12), (31,4), (33,5), (34,2), (39,4), (44,0), (50,3), (51,8), (52,14), (54,8), (56,7), (57,2), (58,8), (60,1), (63,12), (64,2), (66,3), (67,1), (68,12), (69,5), (70,7), (72,7), (80,9), (83,12), (90,9), (95,13), (97,3), (98,4), (99,12), (100,8), (101,2), (103,8), (105,2), (108,8), (109,5), (118,4), (119,5), (126,11), (127,5), (130,2), (133,0), (138,9), (140,4), (145,0), (147,1), (149,6), (151,6), (152,14), (156,1), (158,14), (161,4), (162,2), (163,1), (166,1), (169,1), (170,10), (177,2), (178,8), (179,0), (181,2), (186,4), (190,0), (193,3), (196,14), (201,2), (202,1), (203,0), (205,0), (207,11), (209,2), (211,2), (214,3), (227,3), (231,9), (233,9), (239,10), (241,2), (242,6), (243,8), (244,14), (245,13), (247,9), (251,0), (254,8), (257,0), (268,7), (276,10), (277,8), (282,10), (284,10), (285,3), (288,7), (292,4), (295,0), (304,12), (305,14), (307,13), (308,10), (310,7), (312,6), (314,12), (317,14), (318,13), (323,11), (324,11), (328,6), (329,10), (330,11), (335,11), (340,13), (344,14), (345,5), (346,7), (347,6), (350,13), (352,7), (354,3), (355,4), (357,3), (361,14), (365,5), (367,3), (370,12), (372,9), (376,4), (377,7), (380,12), (382,6), (386,2), (389,9), (392,0), (394,7)}

T_9 = {(1,2), (3,11), (4,1), (5,15), (9,7), (10,0), (20,2), (21,2), (25,8), (26,1), (32,6), (33,10), (37,4), (46,2), (53,7), (54,8), (58,3), (62,5), (68,0), (70,0), (72,0), (73,9), (74,3), (80,10), (81,2), (82,10), (84,13), (86,6), (89,0), (91,4), (92,5), (96,10), (97,5), (99,2), (102,13), (111,1), (115,5), (116,5), (118,4), (123,10), (132,6), (138,8), (139,2), (141,7), (143,7), (146,14), (149,2), (152,4), (154,4), (157,3), (158,1), (159,13), (162,0), (167,0), (170,6), (171,4), (172,4), (174,1), (175,1), (177,9), (178,14), (181,4), (184,12), (190,9), (195,8), (196,0), (198,12), (199,0), (200,0), (201,0), (202,12), (204,14), (207,3), (208,5), (216,15), (220,8), (226,0), (229,15), (230,12), (233,6), (235,8), (237,8), (246,15), (247,1), (248,7), (252,10), (255,12), (257,7), (258,3), (263,2), (264,14), (265,11), (267,2), (269,0), (270,8), (271,11), (272,8), (275,10), (279,6), (286,2), (293,0), (296,10), (298,14), (302,2), (307,7), (311,3), (313,15), (316,8), (319,14), (320,2), (330,10), (331,13), (332,8), (333,2), (335,6), (339,9), (340,2), (344,6), (345,10), (346,6), (350,10), (351,10), (362,14), (363,4), (366,14), (367,4), (368,1), (369,2), (373,0), (375,1), (377,3), (378,11), (380,0), (384,11), (387,4), (390,10), (392,3), (394,1), (395,0), (397,11)}

T_{10} = {(0,10), (3,2), (4,6), (6,11), (8,0), (9,10), (10,0), (11,4), (13,12), (16,14), (17,0), (18,2), (30,10), (31,6), (34,8), (38,2), (42,7), (43,0), (45,2), (49,0), (50,8), (51,12), (52,6), (53,6), (58,10), (60,2), (61,8), (62,6), (67,0), (69,8), (74,2), (79,2), (87,4), (88,0), (91,6), (92,8), (93,2), (97,0), (98,10), (101,14), (104,2), (108,10), (111,10), (112,2), (115,2), (116,10), (122,2), (124,14), (126,8), (129,8), (130,0), (133,10), (135,2), (143,0), (144,8), (145,8), (150,8), (151,10), (156,10), (157,0), (158,10), (159,10), (161,0), (165,8), (170,2), (171,8), (173,4), (174,2), (182,14), (184,2), (191,14), (192,14), (198,4), (204,0), (206,15), (209,2), (216,12), (219,6), (220,6), (222,2), (226,6), (228,2), (232,8), (237,2), (240,14), (244,12), (245,0), (252,14), (253,10), (255,8), (257,10), (258,14), (259,4), (262,8), (267,2), (273,14), (278,6), (279,8), (290,10), (292,2), (296,6), (299,10), (301,2), (303,14), (310,6), (315,0), (319,6), (320,6), (323,8), (324,2), (326,6), (330,4), (331,4), (332,0), (333,8), (336,8), (337,10), (339,0), (340,0), (343,4), (344,4), (349,0), (352,8), (358,8), (366,8), (367,14), (372,0), (376,4), (377,14), (378,12), (379,4), (381,10), (383,12), (388,0), (389,8), (391,8), (393,6), (394,0), (397,12), (399,4)}

T_{11} = {(4,2), (6,8), (10,10), (13,10), (15,10), (16,2), (18,10), (20,10), (23,6), (30,0), (32,10), (37,2), (39,14), (42,14), (45,0), (48,12), (49,2), (51,7), (52,6), (57,11), (59,11), (61,1), (66,0), (73,3), (75,14), (78,11), (82,3), (83,2), (87,13), (90,10), (92,11), (93,11), (95,1), (96,15), (97,3), (98,5), (106,7), (108,0), (109,3), (113,4), (115,7), (116,1), (119,1), (126,7), (130,14), (131,3), (135,5), (138,1), (142,7), (144,5), (146,9), (147,11), (148,0), (149,1), (150,3), (151,3), (153,3), (154,3), (159,1), (164,9), (166,11), (167,3), (168,3), (169,7), (170,9), (178,3), (180,1), (181,5), (183,1), (186,5), (192,3), (193,3), (197,3), (198,5), (202,3), (203,13), (210,9), (212,9), (213,3), (215,1), (217,1), (218,15), (219,5), (220,3), (221,1), (225,1), (226,3), (227,6), (229,12), (230,8), (231,2), (233,2), (235,10), (238,2), (239,1), (244,6), (246,5), (249,11), (250,9), (256,1), (262,7), (265,2), (275,2), (276,8), (279,14), (280,0), (283,8), (286,1), (291,0), (296,2), (300,2), (303,2), (309,0), (311,0), (312,0), (317,0), (318,12), (319,0), (323,0), (327,0), (329,0), (336,0), (345,8), (348,8),

(354,8), (358,8), (359,12), (360,8), (364,0), (367,10), (371,4), (373,0), (374,0), (375,8), (388,1), (390,10), (391,10), (394,2), (397,0), (398,10)}

T_{12} = {(1,3), (7,3), (8,1), (11,13), (14,2), (16,1), (22,11), (23,0), (24,15), (27,8), (28,9), (30,11), (44,9), (45,7), (46,3), (48,7), (50,3), (59,1), (62,3), (63,3), (64,3), (65,7), (66,7), (70,2), (76,1), (80,3), (82,5), (83,7), (84,9), (85,11), (86,5), (94,11), (97,5), (98,9), (100,15), (101,1), (102,3), (109,0), (112,9), (113,9), (114,7), (117,5), (120,1), (121,1), (123,1), (124,3), (127,11), (128,1), (134,3), (140,9), (141,9), (142,5), (152,1), (160,5), (162,3), (163,9), (165,1), (167,11), (172,5), (176,1), (177,0), (178,1), (179,6), (180,2), (181,1), (185,0), (187,7), (189,0), (190,2), (192,15), (198,6), (202,8), (206,2), (208,10), (218,0), (219,2), (221,4), (224,3), (225,0), (227,6), (233,6), (239,3), (240,2), (241,0), (242,0), (243,6), (249,4), (250,4), (251,8), (252,8), (254,8), (256,0), (259,1), (260,1), (263,4), (265,0), (270,1), (272,0), (273,4), (277,2), (278,2), (284,4), (287,2), (290,2), (293,6), (301,10), (307,10), (311,2), (312,6), (313,12), (316,10), (321,10), (322,12), (323,12), (324,0), (325,0), (330,8), (331,2), (334,10), (335,10), (336,8), (337,10), (341,14), (344,10), (347,10), (349,6), (350,4), (353,8), (355,0), (358,2), (359,12), (360,1), (369,2), (380,4), (383,14), (384,11), (391,3), (392,2), (393,13), (397,3)}

T_{13} = {(5,3), (7,0), (9,1), (11,11), (12,3), (13,1), (15,1), (23,3), (24,3), (25,3), (27,1), (31,9), (33,9), (35,9), (36,9), (41,11), (42,3), (43,3), (47,1), (48,7), (49,11), (51,13), (52,1), (53,3), (58,5), (65,15), (69,13), (71,1), (81,3), (85,9), (86,9), (88,7), (94,11), (96,3), (97,2), (102,7), (104,15), (106,7), (113,10), (114,2), (115,3), (120,2), (122,1), (123,3), (125,5), (127,1), (129,1), (130,4), (140,3), (142,11), (144,1), (146,3), (147,5), (148,15), (152,3), (154,0), (156,2), (157,9), (160,10), (162,6), (165,12), (166,14), (173,8), (175,8), (176,2), (179,11), (183,6), (190,9), (192,10), (193,4), (195,7), (199,0), (205,2), (206,1), (208,6), (209,0), (210,10), (212,2), (213,8), (214,0), (216,10), (218,1), (221,10), (222,0), (223,8), (224,10), (229,10), (230,0), (231,2), (236,2), (243,9), (246,0), (249,13), (261,10), (266,4), (271,0), (273,6), (276,10), (281,6), (282,0), (283,0), (284,8), (289,12), (290,0), (293,2), (298,10), (300,0), (304,8), (305,8), (307,4), (312,0), (314,4), (315,0), (316,2), (318,10), (319,12), (320,8), (325,12), (329,0), (334,1), (338,3), (342,4), (347,9), (350,1), (351,2), (352,2), (353,0), (354,11), (361,9), (364,13), (365,11), (368,9), (371,3), (372,0), (380,3), (382,4), (384,0), (385,1), (386,1), (399,4)}

T_{14} = {(0,8), (6,0), (12,0), (19,2), (20,0), (25,8), (26,10), (29,0), (30,2), (32,0), (34,0), (40,8), (41,0), (47,0), (49,0), (51,2), (52,2), (54,8), (55,0), (58,2), (60,0), (63,0), (64,0), (66,3), (69,1), (72,8), (75,9), (77,0), (78,0), (79,5), (81,9), (82,12), (87,4), (90,1), (91,15), (92,11), (94,1), (95,0), (100,11), (105,10), (106,10), (109,11), (110,2), (112,15), (115,1), (118,9), (120,3), (124,13), (125,0), (126,15), (132,3), (133,11), (135,3), (140,11), (141,2), (143,9), (145,9), (146,11), (153,7), (155,1), (161,11), (163,7), (167,8), (172,1), (175,15), (178,1), (183,0), (184,8), (186,1), (187,11), (194,3), (197,11), (201,3), (206,3), (208,13), (211,3), (216,3), (219,1), (223,3), (225,11), (235,9), (237,9), (238,3), (240,3), (246,11), (252,9), (259,1), (264,3), (273,2), (274,9), (275,9), (276,5), (277,11), (280,9), (282,9), (287,1), (289,5), (293,9), (294,0), (298,2), (300,3), (301,1), (310,1), (313,0), (315,10), (316,12), (318,0), (319,12), (320,8), (321,10), (322,1), (327,1), (328,2), (330,0), (331,0), (334,2), (337,4), (339,0), (340,2), (359,1), (362,8), (365,5), (368,8), (370,10), (371,1), (372,2), (374,2), (379,1), (381,8), (382,8), (384,4), (386,4), (387,0), (388,4), (391,3), (393,0), (395,2), (396,2), (398,10), (399,0)}

T_{15} = {(2,2), (4,10), (8,8), (11,0), (13,0), (16,0), (21,8), (26,2), (27,10), (28,2), (29,0), (32,3), (36,0), (38,2), (39,11), (42,1), (51,2), (54,10), (56,0), (59,0), (61,0), (63,1), (65,3), (67,11), (68,0), (71,3), (72,3), (74,10), (79,1), (82,0), (84,8), (87,1), (88,3), (94,8), (102,1), (106,9), (112,11), (116,8), (118,0), (121,11), (124,3), (126,9), (127,1), (131,9), (132,0), (134,1), (135,1), (136,1), (139,1), (140,1), (142,1), (149,1), (150,1), (156,1), (157,1), (158,3), (159,1), (162,11), (163,1), (165,9), (166,2), (172,0), (173,1), (174,3), (179,1), (181,1), (184,1), (185,1), (189,1), (191,3), (194,9), (195,9), (197,0), (201,1), (203,3), (208,1), (209,8), (212,8), (214,3), (215,11), (218,11), (219,8), (221,1), (227,11), (230,3), (232,3), (234,3), (243,9), (244,11), (252,2), (253,1), (260,0), (261,11), (265,9), (271,11), (272,1), (273,1), (281,10), (282,2), (283,9), (285,8), (288,10), (289,8), (291,11), (294,2), (303,3), (306,10), (310,2), (317,2), (318,11), (319,9), (323,3), (329,1), (330,1), (333,2), (337,8), (338,2), (339,2), (341,10), (342,0), (343,2), (344,0), (345,0), (349,10), (350,0), (353,8), (356,0), (359,0), (362,3), (366,8), (370,0), (374,10), (377,0), (379,10), (381,8), (383,8), (384,0), (390,2), (391,8), (395,0)}

T_{16} = {(7,4), (11,5), (17,3), (19,6), (22,5), (27,6), (29,6), (30,7), (35,14), (42,6), (47,14), (54,4), (55,4), (56,2), (63,1), (64,6), (66,0), (67,2), (71,13), (72,0), (73,0), (75,12), (76,10), (77,4), (79,0), (80,0), (85,7), (89,6), (90,4), (93,6), (94,3), (96,5), (98,4), (100,0), (106,2), (113,14), (116,4), (118,2), (122,2), (126,12), (128,6), (133,6), (136,10), (137,14), (141,2), (144,14), (147,4), (148,2), (150,0), (151,0), (153,14), (155,2),

(157,2), (162,2), (163,7), (164,2), (167,1), (169,3), (176,1), (182,12), (185,4), (187,3), (189,0), (190,1), (191,11), (197,1), (203,5), (210,12), (211,0), (212,0), (216,0), (218,3), (220,11), (224,4), (225,7), (228,10), (232,3), (235,8), (236,0), (237,1), (238,9), (244,7), (245,1), (252,0), (256,1), (258,3), (260,7), (261,1), (265,7), (266,1), (269,5), (270,1), (275,3), (276,2), (277,5), (279,0), (280,7), (284,7), (285,9), (288,1), (290,1), (295,7), (297,9), (298,1), (299,3), (300,3), (301,0), (302,1), (303,1), (304,3), (305,1), (308,1), (311,0), (313,1), (315,1), (321,5), (325,2), (326,5), (328,5), (331,10), (332,13), (334,1), (335,7), (337,5), (341,5), (344,4), (355,4), (356,5), (357,5), (358,4), (362,6), (366,5), (373,4), (374,4), (381,1), (385,1), (386,13), (395,4), (397,5), (398,6)}

T_{17} = {(0,1), (2,5), (7,2), (12,1), (14,5), (19,1), (21,5), (22,5), (23,7), (24,3), (27,2), (28,9), (32,1), (35,13), (37,1), (39,1), (40,1), (41,0), (43,1), (44,10), (46,1), (47,0), (53,5), (57,1), (62,11), (64,5), (65,5), (68,5), (77,7), (79,1), (83,2), (85,1), (86,4), (91,1), (92,3), (94,4), (95,4), (103,3), (104,0), (105,4), (106,8), (107,5), (109,7), (112,7), (113,2), (119,5), (120,7), (126,1), (135,1), (143,2), (144,4), (145,12), (146,13), (153,5), (154,5), (160,4), (164,6), (168,0), (169,2), (172,0), (177,13), (178,4), (179,7), (188,0), (190,4), (192,8), (194,8), (195,4), (196,0), (200,0), (201,5), (213,11), (215,4), (229,1), (230,4), (231,0), (235,0), (242,4), (243,0), (244,4), (246,6), (247,4), (249,4), (252,4), (253,4), (254,4), (258,0), (264,0), (266,4), (269,7), (271,0), (272,0), (274,4), (281,2), (283,0), (284,0), (287,6), (289,2), (298,7), (299,10), (304,1), (305,3), (306,4), (307,7), (309,6), (320,14), (321,1), (322,7), (327,7), (328,6), (329,2), (331,10), (333,6), (343,0), (345,3), (346,0), (351,1), (352,11), (353,5), (354,2), (357,2), (359,1), (362,0), (363,3), (364,7), (365,6), (368,5), (369,5), (374,0), (378,5), (381,2), (382,7), (383,3), (384,12), (385,6), (386,0), (390,3), (393,3), (396,1), (399,1)}

T_{18} = {(0,2), (9,3), (10,6), (14,2), (17,6), (23,7), (24,7), (26,2), (28,5), (29,12), (35,5), (39,4), (41,5), (43,3), (47,7), (48,3), (49,4), (50,0), (60,2), (61,2), (62,4), (65,0), (66,5), (67,14), (68,2), (69,4), (70,4), (74,4), (75,6), (76,2), (79,2), (89,6), (91,4), (93,6), (98,7), (100,4), (103,4), (106,1), (114,0), (122,4), (123,6), (124,14), (125,14), (128,0), (129,4), (130,4), (134,6), (136,4), (137,4), (143,5), (146,5), (148,4), (150,5), (154,0), (160,4), (161,4), (164,0), (170,4), (173,0), (174,1), (177,4), (182,5), (187,1), (188,6), (191,0), (192,1), (200,7), (202,0), (210,3), (211,9), (215,1), (216,1), (220,0), (221,1), (222,5), (223,3), (225,0), (234,5), (238,0), (239,5), (254,0), (255,5), (257,1), (260,13), (261,7), (267,7), (268,1), (272,5), (274,3), (277,3), (278,5), (279,1), (281,1), (283,6), (285,3), (287,0), (290,1), (294,1), (296,1), (297,1), (303,1), (304,0), (305,5), (307,5), (310,0), (313,2), (314,1), (323,3), (326,1), (328,2), (329,0), (332,1), (337,0), (341,0), (342,3), (343,3), (344,1), (348,1), (349,1), (352,1), (353,2), (354,1), (356,9), (357,0), (360,1), (361,13), (363,1), (366,0), (370,5), (375,0), (378,5), (380,1), (381,4), (385,0), (386,5), (387,4), (392,6), (393,7), (396,5), (398,5)}

T_{19} = {(0,4), (5,12), (8,4), (9,8), (10,12), (14,8), (17,0), (18,10), (20,4), (21,9), (23,13), (25,5), (32,9), (35,12), (37,0), (40,12), (41,8), (42,0), (46,1), (49,1), (50,13), (52,12), (55,9), (56,0), (57,6), (59,14), (60,4), (61,10), (67,0), (68,0), (76,4), (78,6), (83,0), (89,6), (91,8), (96,0), (98,0), (101,8), (102,6), (105,1), (110,2), (114,5), (117,15), (119,1), (121,10), (123,5), (127,15), (129,0), (134,7), (137,9), (139,10), (141,1), (148,13), (149,4), (151,9), (153,4), (156,10), (158,11), (160,1), (161,11), (166,1), (169,12), (172,2), (179,11), (187,6), (188,9), (189,2), (194,5), (197,0), (198,1), (200,7), (201,13), (205,2), (206,13), (210,9), (211,1), (212,3), (214,11), (220,1), (221,2), (222,1), (234,1), (236,7), (237,8), (238,1), (239,3), (240,1), (242,7), (245,13), (248,13), (250,5), (251,7), (258,5), (259,3), (260,7), (261,14), (263,13), (269,5), (270,9), (274,0), (279,5), (282,6), (283,4), (284,0), (291,5), (295,3), (296,11), (299,5), (301,6), (302,2), (303,5), (305,4), (306,5), (310,9), (312,4), (314,13), (324,5), (332,12), (333,6), (339,7), (340,13), (341,14), (345,13), (348,12), (349,0), (352,12), (354,0), (355,1), (357,8), (361,0), (364,13), (370,13), (372,14), (379,12), (380,4), (381,11), (383,5), (392,10), (394,10), (395,8)}

8 Specification 6330 code and RaptorQ LA code

8.1 Introduction

This clause specifies FEC code point 3, the 6330 code, and FEC code point 4, the Layer-Aware extension of 6330 code, RaptorQ LA.

If FEC code point 4 (RaptorQ LA) is used with `fec_coding_structure==0011` in 23008-1, for FEC flows protecting the base layer the FEC code point 3 (6330 code) shall be signaled and for FEC flows protecting enhancement layers, the FEC code point 4 (RaptorQ LA) shall be signaled.

8.2 6330 code

The 6330 code is defined in IETF RFC 6330.

The relevant definitions that document the interfaces to the 6330 code encoding and decoding are provided

— Source block: a block of K source symbols that are considered together for encoding and decoding purposes.

— Symbol: a unit of data. The size, in octets, of a symbol is known as the symbol size T. The symbol size is always a positive integer.

— Source symbol: the smallest unit of data used during the encoding process. All source symbols within a source block have the same size T.

— Encoding symbol: a symbol that can be sent as part of the encoding of a source block. The encoding symbols of a source block consist of the source symbols of the source block and the repair symbols generated from the source block. Repair symbols generated from a source block have the same size T as the source symbols of that source block.

— Repair symbol: the encoding symbols of a source block that are not source symbols. The repair symbols are generated based on the source symbols of a source block.

— Encoding Symbol ID (ESI): information that uniquely identifies each encoding symbol associated with a source block for sending and receiving purposes. For a source block with K source symbols, the ESIs for the source symbols are 0, 1, 2, ..., K-1, and the ESIs for the repair symbols are K, K+1, K+2, ...

If FEC code point 3 is signaled, then for a source block of size K, a repair symbol with any ESI >= K shall be generated as defined in clause 5.3 of IETF RFC 6330.

8.3 RaptorQ LA

RaptorQ LA extends the 6330 code for efficient support of LA-FEC as defined in ISO/IEC 23008-1.

This sub-clause extends “5.3.3 First Encoding Step: Intermediate Symbol Generation” and sub-clause “5.3.4 Second Encoding Step: Encoding” of IETF RFC 6330. Relevant terminology and algorithms are specified in IETF RFC 6330 with the following additional definitions:

K_x is the number of source symbols in the source block of media layer x (cf. K in sub-clause 5.1.2 in IETF RFC 6330).

K'_x is the number of source symbols in the extended source block of media layer x (cf. K’ in sub-clause 5.1.2 in IETF RFC 6330).

P_x is the number of parity symbols of media layer x.

L_x denotes the number of intermediate symbols for a single extended source block of media layer x (cf. L in sub-clause 5.1.2 in IETF RFC 6330).

C_x denotes an array of intermediate symbols, $C_x[0]$, ... $C_x[L_x-1]$, of media layer x

Enc[K’, C ,(d, a, b, d1, a1, b1)] denotes an encoding symbol generator (cf. sub-clause 5.3.5.3 in IETF RFC 6330).

ISI is the internal symbol ID (cf. sub-clause 5.3.1 in IETF RFC 6330).

ESI is the encoded symbol ID (cf. sub-clause 5.3.1 in IETF RFC 6330).

$G\_ENC_{m\_LA\_z}$ is the LA-FEC z-th extension matrix of the G_ENC matrix of media layer m.

Matrix A is the constraint matrix in calculation process of intermediate symbols in the first encoding step of the 6330 code process as used for the code point in sub-clause 8.2 (cf. Figure 5 in sub-clause 5.3.3 in IETF RFC 6330). Matrix A is illustrated in Figure 2.

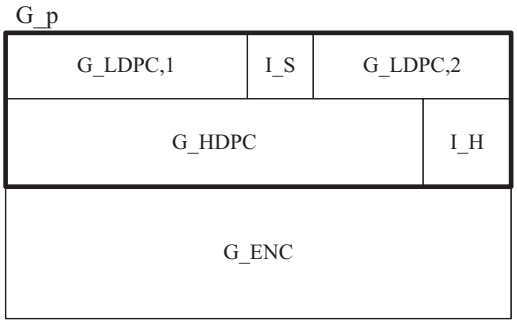

Figure 2 — Constraint Matrix A (cf. Figure 5 in sub-clause 5.3.3 in IETF RFC 6330)

Matrix A consists of six submatrices, 'G_LDPC,1', 'I_S', 'G_LDPC,2', 'G_HDPC', 'I_H' and 'G_ENC'. The construction of those submatrices is described in sub-clause 5.3.3 in IETF RFC 6330.

We define the submatrix G_p as the submatrix which contains all submatrices for precoding.

$$G\_p = \begin{bmatrix} G\_LDPC,1 & I\_S & G\_LDPC,2 \\ G\_HDPC & & I\_H \end{bmatrix}$$

Hence, matrix A consists of G_p and G_ENC:

$$A = \begin{bmatrix} G\_p \\ G\_ENC \end{bmatrix}$$

The intermediate symbols C can then be calculated from the source symbols D as:

$$C = (A^{-1})\ D$$

If FEC code point 4 is signaled, then for all source blocks of the LA-FEC process of a target media layer x_target of size $K_1, K_2, \ldots, K_{x\_target}$, a repair symbol with any ESI >= K_{x_target} shall be generated as defined in sub-clause 8.3.1 and 8.3.2.

8.3.1 RaptorQ LA First Encoding Step – Intermediate symbol generation

This sub-clause specifies the intermediate symbol generation for RaptorQ LA for a given media layer 2 that depends on media layer 1. Media layer 1 represents the base layer with dependency_id = 0 and media layer 2 represents the enhancement layer with dependency_id = 1 as shown in Figure 14 in the AL-FEC Framework in ISO/IEC 23008-1.

With RaptorQ LA, matrix A is extended for integration of all media layers in a constraint matrix $A_{\_LA\_L}$ in a way to preserve the systematic behavior of the code as shown in the two layered example in Figure 3.

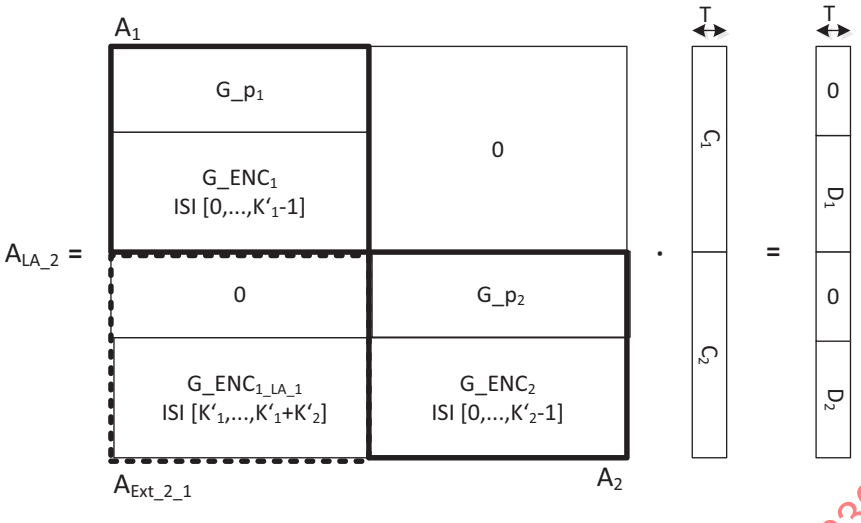

Figure 3 — Encoding process with Layer-Aware RaptorQ constraint matrix ALA_2 for 2 media layers

In the given example in Figure 3, matrix A_{LA_2} consists of the matrices A_1, A_2, the LA-FEC extension matrix $A_{Ext\_2\_1}$ and a zero matrix.

A_1 and A_2 are generated as described in sub-clause 5.3.3 in IETF RFC 6330 with the constraint that the symbol size of all source and intermediate symbols T is equal in all cases which may result in different number of source symbols K_1, K_2. the matrices denoted by '0' are zero matrices of respective sizes.

The LA-FEC extension matrix B_{1_LA} comprises the continuation matrix $G\_ENC_{1\_LA\_1}$ and a zero matrix. $G\_ENC_{1\_LA\_1}$ is the continuation of the G_ENC_1 matrix of media layer l=1, which is part of the A_1 matrix. The continuation matrix $G\_ENC_{1\_LA\_1}$ is generated in a way that each row is unique in G_ENC_1 and $G\_ENC_{1\_LA\_1}$. It is generated in the same way as G_ENC_1 by function Enc, which is described in sub-clause 5.3.3.4.2 in IETF RFC 6330. While G_ENC_1 is generated from ISIs [0: K'_1-1], $G\_ENC_{1\_LA\_1}$ is generated from ISIs [K'_1,...,K'_1+K'_2-1].

We describe the LA-FEC continuation matrix as $B_{1\_LA\_1}$=[0; $G\_ENC_{1\_LA\_1}$], the matrix A_{LA_2} of Layer-Aware RaptorQ is described for media layer l=2 by:

$$A_{LA\_2} = \begin{bmatrix} G\_p_1 & 0 \\ G\_ENC_1 & 0 \\ 0 & G\_p_2 \\ G\_ENC_{1\_LA\_1} & G\_ENC_2 \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ B_{1\_LA\_1} & A_2 \end{bmatrix}$$

The intermediate symbols C_1 and C_2 can then be calculated from source symbols D_1 and D_2 as:

$$[C_1C_2] = (A_{LA\_2}\text{^^-1}) \cdot [D_1D_2]$$

This concept can further be extended for the x-th media layer as shown by matrix A_{LA_x}:

$$A_{LA\_x} = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ B_{1\_LA\_1} & A_2 & & 0 \\ \vdots & & \ddots & \vdots \\ B_{1\_LA\_n-1} & B_{2\_LA\_n-2} & \cdots & A_n \end{bmatrix}$$

$A_{Ext\_i\_j}$ is the j-th continuation matrix of G_ENC_i generated from ISIs $[\sum_{x=j}^{i-1} K'_x : \sum_{x=j}^{i} K'_x - 1]$.

The intermediate symbols C_1,C_2,...,C_x can then be calculated from source symbols D_1,D_2,...,D_x as:

$$[C_1C_2...C_x] = (A_{LA\_x}\text{^^-1})*[D_1D_2...D_x]$$

For solving the equation an efficient decoding process as described by the example FEC Decoder in sub-clause 5.4 in IETF RFC 6330 can be applied.

8.3.2 Layer-Aware RaptorQ Second Encoding Step

The second encoding steps generates the repair symbols E_x of media layer x from the intermediate symbols C.

In the following we consider the case with *n* media layers with the intermediate symbols $C=[C_1C_{2...}\ C_n]$ and describe the encoding process for generation of repair symbols E_{x_target} of media layer x_target with x_target being the target layer for which repair symbols should be encoded.

The generation of the repair symbols *E* from intermediate symbols *C* is described in sub-clause 5.3.4 in IETF RFC 6330. The repair symbols E_{x_target} are generated from all intermediate symbols $C^{(x\_target)}=[C_1C_{2...}\ C_{x\_target}]$. The process for generation of the non-systematic encoding symbols is described in the following pseudo code:

For $\left(i=\sum_{x=1}^{x\_target} K'_x+\sum_{x=1}^{x\_target-1} P_x; i<\sum_{x=1}^{x\_target} K'_x+\sum_{x=1}^{x\_target} P_x; i++\right)$

E_{x_target} [i] = 0;

for layer 1:x_target

E_{x_target} [i]= E_{x_target} [i] XOR Enc(K'_{layer}, C_{layer}, [d,a,b,d1,a1,b1])

end

end

8.3.3 Layer-Aware RaptorQ Decoding

The decoding process is performed analogously to the encoding process of a set of received encoding symbols represented by tuples R_x with the related ISIs $[R_x]$ of the involved FEC flows of media layers x=1,...,n. The decoding process for decoding all layers together is similar to the encoding process (sub-clause 8.2.1). The difference is that for the first encoding process the matrix G_ENC_1 and G_ENC_2 are generated from ISI $[R_1]$ and ISI $[R_2]$ and $G\_ENC_{1\_LA\_1}$ is generated from ISI $[R_2(+K'_1)]$ of all received encoded symbols R_2, where $(+K'_1)$ denotes that each element in R_2 is incremented by $(+K'_1)$.

Note that layers used for prediction, e.g. base layer, can also be decoded independently of predicting layers, e.g. enhancement layer, and that the FEC flow of media layer x=1 is generated with FEC code point 3.

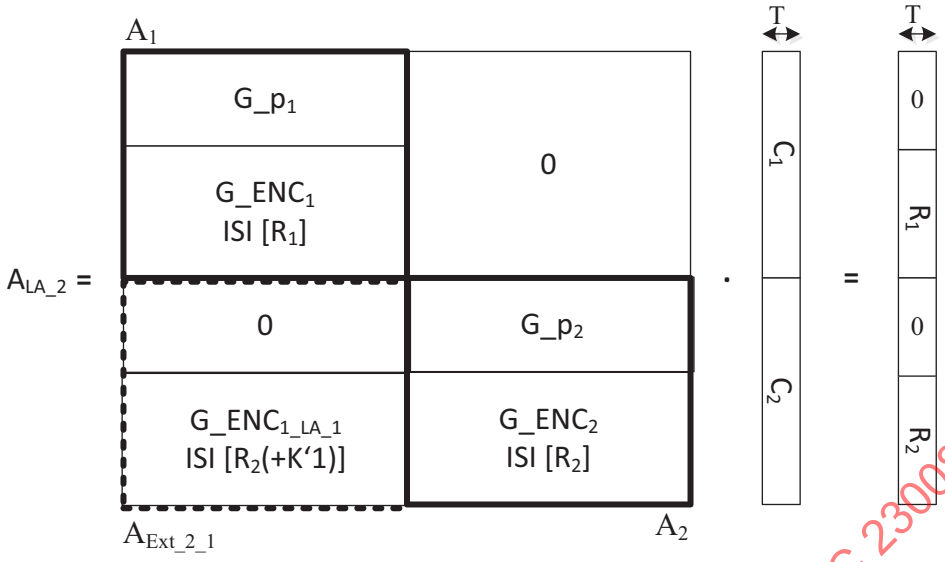

Figure 4 — Decoding process with Layer-Aware RaptorQ constraint matrix ALA_2 for 2 media layers

An exemplary decoder for solving the decoding process is specified in sub-clause 5.4 of IETF RFC 6330.

The second encoding process is analogously to the process described in 8.3.2 with C=[C_1C_2] as input from the first encoding process.

9 Specification for FireFort Low Density Generate Matrix (FireFort-LDGM) codes

9.1 Introduction

FireFort-LDGM (FF-LDGM) code is a special case of irregular repeat–accumulate codes that are well known as LDPC staircase in IETF RFC 5170. The specified FF-LDGM codes can support packet-division scheme and structured interleaving. Also the difference of the specified FF-LDGM code and LDPC staircase in IETF RFC 5170, the specified FF-LDGM codes can use optimized parity check matrixes for low complexity message passing decoding algorithm. Furthermore, Layer-Aware FF-LDGM codes that consist of FF-LDGM codes are suitable to support Layer-Aware FEC coding structure. This clause provides full specification of FF-LDGM/Layer-Aware FF-LDGM codes, which works on the same code point.

9.2 FireFort Low Density Generator Matrix (FireFort-LDGM) Codes

9.2.1 Definition

— $\mathbf{H}_{P',N'}$: denotes a sparse parity check matrix with P' rows and N' columns.

— $\mathbf{G}_i$: denotes a mother matrix.

— $\mathbf{G'}_{P',K'}$: denotes a punctured sparse matrix with P' rows and K' columns.

— $\mathbf{T}_{P'}$: denotes a P'-by-P' staircase matrix.

— $\mathbf{W}_{N'}$: denotes N' Encoding multiple symbols.

— $\mathbf{S}_{K'}$: denotes K' multiple source symbols.

— $\mathbf{C}_{P'}$: denotes P' multiple repair symbols

— K : denotes a number of source symbols

— P : denotes a number of repair symbols

— D : denotes division number of one source symbol

— K' : denotes a number of multiple source symbols K*D

— P' : denotes a number of multiple repair symbols P*D

— MK' : number of mother matrix columns

— MP' : number of mother matrix rows

— $\mathbf{V}_{P'}$: denotes a P' intermediate multiple repair symbols

9.2.2 FF-LDGM-Specific Elements

The following elements shall be defined with the FF-LDGM codes.

— N1m3 : an integer between 0 and 255 inclusive. The FF-LDGM scheme supports two target numbers of "non-zero elements" per column in the left side of the parity check matrix, N1 and N2. N1m3 is related to N1 which equals N1m3 + 3. The default value of N1m3 is set to 0.

— N2m4 : an integer between 0 and 255, inclusive. N2m4 is related to N2, which equals N2m4 + 4. The default value of N2m4 is set to 3.

— R : an integer between 0 and 255, inclusive. R indicates ratio between # of N1 columns ('#N1) and # of N2 columns (#N2). R is equal to Ceil[#N2/(#N1+#N2)*255]. The default value of R is set to 48.

— Dm1 : an integer between 0 and 255, inclusive, indicating division number of one source symbol. The division number D equals Dm1 + 1. The default value of Dm1 is set to 0.

— GC: an integer between 0 and 255, inclusive, indicating number of Galois field elements for each column.

— MKm9 : an integer between 0 and 15, inclusive, indicating maximum number of mother matrix columns. The MK equals MKm9 + 9 and it is used in rate-adaptive mode only.

— MPm4 : an integer between 0 and 15, inclusive, indicating maximum number of mother matrix rows. The MP equals MPm4 + 4 and it is used in rate-adaptive mode only.

— RES : a reserved field for future use.

— M : a flag indicating use of rate-adaptive mode.

— L : an integer between 0 and 3, inclusive. L indicates target girth on Progressive Edge-Growth (PEG) algorithm. The default value of L is set to 2.

When the code point 5 is used for MMT AL-FEC Framework, the FF-LDGM specific elements are described in private field of AL-FEC message in ISO/IEC 23008-1 and the format is written as follows:

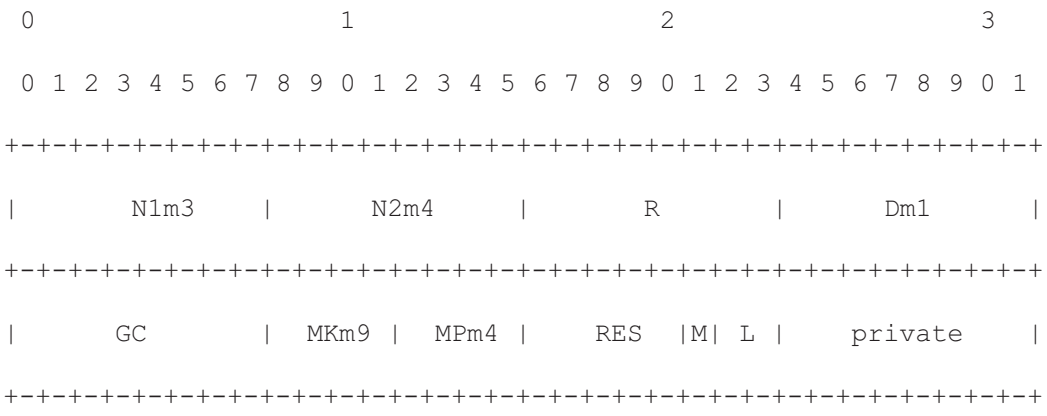

Figure 5 — Structure of FF-LDGM parameters

9.2.3 Parity Check Matrix of FF-LDGM Scheme

FF-LDGM codes are one type of linear code; the parity check matrix contains mostly 0's and only a small number of non-zero's. The parity check matrixes of FF-LDGM codes consist of 2 parts: one is sparse matrix **G**' and the other is staircase matrix **T** and it given by

$$\boldsymbol{H}_{P',N'} = \left[\boldsymbol{G}'_{P',K'}; \boldsymbol{T}_{P'}\right] = \left[\begin{array}{ccc|ccccc} & 1 & & 1 & & & & \\ 1 & & & 1 & 1 & & & \\ \cdots & \cdots & \cdots & & \ddots & \ddots & & \\ & & 1 & & & 1 & 1 & \end{array}\right] \quad (1)$$

where **G**' is a P'-by-K' sparse matrix and $\mathbf{T}_{P'}$ is a P'-by-P' square staircase matrix. In the case of rate-adaptive mode, sparse matrix **G**' is generated from mother matrix **G**.

9.2.4 Creating a Sparse Matrix

Sparse matrix **G** ' and mother matrix **G** are generated by using the following procedure which is shown in Table 2.

Table 2 — Procedure of creating a sparse matrix

Procedure of creating a sparse matrix
void BigGirthMatrix(K', P', N1, N2, R, L, GC) { /* This function assumes that two empty matrixes, one is P' * (K'+P') matrix H' and the other is P' * K' matrix G'. The matrix_insert_entryH(), matrix_resetH(), connectedNodes() and connectedNodes_clear() functions can access H', and the matrix_insert_entryG(),permutation_matrixG() and nonbinary_matrixG() functions can access G'. The pmms_rand() shall be used in IETF RFC 5170 in clause 5.7 with PRNG seed=0x600DC0DE. In the case of rate-adaptive mode, mother matrix G is generated by using MK and MP parameters and then the K' is set to 2^^MK and P' is set to 2^^MP. */

Table 2 *(continued)*

```
    reset_counter = -1;
    reset_flag = -1;
/* make stair matrix */
    FOR i = 0 to P'-1
        matrix_insert_entryH(i,i);
        IF (i != 0) THEN
        matrix_insert_entryH(i,i-1);
        ENDIF
        row_weights[i]=2;
    ENDFOR
    row_weights[0]--;

    FOR j=0 to K'-1
        tgt_gitch = L*2+2;
    ITER:
        ConnectedNodes_clear(); /* Clears all ConnectedNode values. */
        IF (j < K'-floor(K'*R/255)) THEN
            weight_cols = N1;
        ELSE{
            weight_cols = N2;
        ENDIF
        FOR w=1 to weight_cols
            min = infinity;
            FOR i=0 to P'-1
                IF ((row_weights[i] < min)AND(ConnectedNodes(i) == 0))
                    min = row_weights[i];
                ENDIF
            ENDFOR
            max_index_length = 0;
            FOR i = 0 to P'-1
                IF ((row_weights[i] == min)AND(ConnectedNodes(i) == 0)) {
                    index[max_index_len] = i;
                    max_index_len += 1;
                ENDIF
            ENDFOR
```

Table 2 *(continued)*

```
            ret = pmms_rand(max_index_len);
/* This function returns random integer between 0 and max_index_len-1.*/
            IF (w != weight_cols)THEN
                IF (reset_flag !=1 )THEN
                  ConnectedCheckNode(index[ret],tgt_girth, j, p);
/* This function finds a minimum non-empty subset of unreachable check nodes from
the target node via from 3 to less than tgt_girth edges.*/
sum ← Sum of all ConnectedNodes values
                  IF (sum == P') THEN
                        IF (tgt_girth >= 4) THEN
                            tgt_girth -= 2; goto ITER;
                        ELSEIF (tgt_girth < 4) AND (reset_counter==j)THEN
                            reset_flag = 1; goto ITER;
                        ELSE
                            reset_counter = j;
                            matrix_resetH(j); goto ITER;
/* This function clears values of columns form P' to j on H'. */
                            tgt_girth = L*2+2; goto ITER;
                        ENDIF
                  ENDIF
                ELSE
                  Set of all ConnectedNodes are zeros.
                ENDIF
            ENDIF
        ENDFOR
        matrix_insert_entryH(index[ret],j+p);
        matrix_insert_entryG(index[ret],j);
        row_weights[index[ret]]++;
    ENDFOR
    permutation matrixG(K', P', R);
/* This function permutes columns of a matrix G'. */
     IF (GC!=0) THEN
     nonbinary_matrixG(K', P', GC);
/* This function enhance a binary matrix to a non-binary matrix. */
     ENDIF
}
```

The procedure of permutation operation in Table 2 is shown in Table 3 and the procedure of nonbinary_matrixG in Table 2 is shown in Table 4.

Table 3 — Procedure of generating a permutation matrix

Procedure of generating a permutation matrix

```
void permutation_matrixG(K', P', R)
{
/* This function permutes columns of a sparse matrix G'. Before permutation, the sparse matrix G' is constructed with non-uniform weight distribution of columns.*/
nN1 = floor(K'*R/255);
nN2 = K'-nN1;
n1 = nN1;
n2 = nN2;
N1_counter = N2_counter = counter = 0;
FOR i=0 to nN2
interval = floor((n1+n2+1)/(n2+1));
FOR j=0 to interval-1
IF (j!=interval-1) THEN
randperm[j+counter] = N1_counter;
N1_counter = N1_counter+1;
ELSE
IF (i!=nN2) THEN
randperm[j+counter] = nN1+N2_counter;
N2_counter = N2_counter+1;
ENDIF
ENDIF
 counter = counter + interval;
n1 = n1-(interval-1);
n2 = n2-1;
ENDFOR
ENDFOR
FOR i=0 to K'-1
change_colums(i,randperm[i]);
/*This function changes two columns (i and randperm[i]) of G'. */
ENDFOR
}
```

Table 4 — Procedure of enhancing a binary matrix to a non-binary matrix

Procedure of enhancing a binary matrix to a non-binary matrix

```
void nonbinary_matrixG(K', P', GC)
{
/* This function enhances a binary matrix G' to a non-binary matrix G'. However,
if GC is set to 0, this function does not applied.*/
 FOR i = 0 to K'-1
 GC_counter = 0;
 FOR j = P'-1 to 0
 IF (return_matrixG_value(j,i)) AND (GC_counter<GC) THEN
/* This function returns value of element in i-th row and j-th column of a
matrix G'. */
 ret = pmms_rand(255);
 matrix_insert_nonbinary(j,i,ret+1);
/* This function inserts nonbinary value (ret+1) of element in j-th row and i-th
column of a matrix G'. */
 GC_counter++;
 ENDIF
 ENDFOR
 ENDFOR
}
```


9.2.5 Creating a Punctured Sparse Matrix

In the case of rate-adaptive mode, the systematic part of parity check matrix is a punctured sparse matrix, which is created by a mother matrix. In this mode, MK' should be set to larger than K'. The punctured matrix is created as follows:

— Step1) Removing (MK'-K') columns of mother matrix from right side. (Figure 6 Step 1)

— Step1') In the case of MP' is smaller than P', generating intermediate sparse matrix with MP' *2^ceil(Log2(P'/MP')) rows and K' columns.

— Step2) Puncturing (MP'*2^ceil(Log2(P'/MP')) -P') rows of intermediate sparse matrix from topside. This puncturing operation is a simple sum operation of two rows over GF(2) in the case of GC is set to 0 or GF(256) in the case of GC is not set to 0. (Figure 6 Step 2)

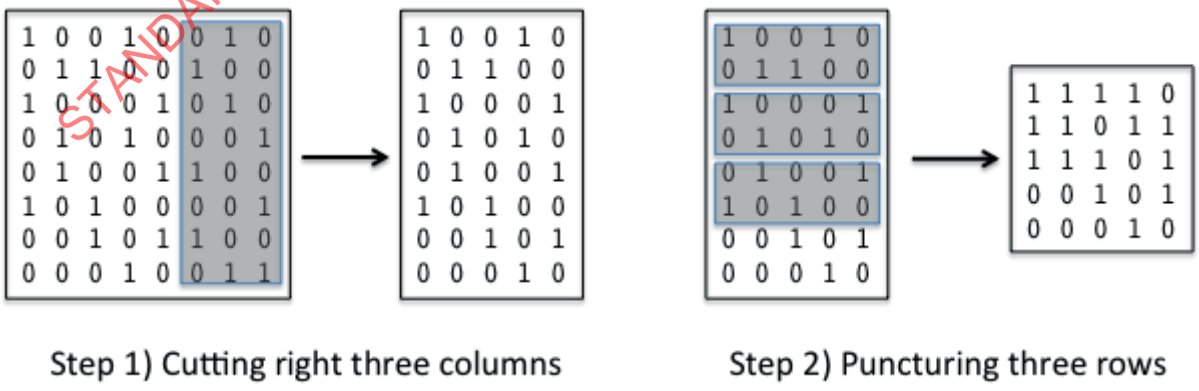

Figure 6 — Puncture operation example to generate punctured matrix (MK=8, K'=5, P'=5, GC=0 case)