Low Density Parity Check (LDPC) Forward Error Correction
draft-ietf-rmt-bb-fec-ldpc-01.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on August 5, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document describes two Fully-Specified FEC Schemes, LDPC-Staircase and LDPC-Triangle, and their application to the reliable delivery of objects on packet erasure channels. These systematic FEC codes belong to the well known class of "Low Density Parity Check" (LDPC) codes, and are large block FEC codes in these sense of
# Table of Contents

1. Introduction ........................................... 3  
2. Requirements notation ................................. 4  
3. Definitions, Notations and Abbreviations .............. 5  
   3.1. Definitions ................................... 5  
   3.2. Notations .................................... 5  
   3.3. Abbreviations ................................. 6  
4. Formats and Codes .................................... 7  
   4.1. FEC Payload IDs ................................ 7  
   4.2. FEC Object Transmission Information .......... 7  
      4.2.1. Mandatory Elements ....................... 7  
      4.2.2. Common Elements ........................... 7  
      4.2.3. Scheme-Specific Element ................. 8  
      4.2.4. Encoding Format ........................... 8  
5. Procedures ............................................ 11  
   5.1. General ....................................... 11  
   5.2. Determining the Maximum Source Block Length (B) 12  
   5.3. Determining the Encoding Symbol Length (E) and Number of Encoding Symbols per Group (G) ........... 12  
   5.4. Determining the Number of Encoding Symbols of a Block ........................................... 13  
   5.5. Identifying the Symbols of an Encoding Symbol Group ................................................. 15  
   5.6. Pseudo Random Number Generator ................ 18  
6. Full Specification of the LDPC-Staircase Scheme ...... 20  
   6.1. General ....................................... 20  
   6.2. Parity Check Matrix Creation ................... 20  
   6.3. Encoding ...................................... 22  
   6.4. Decoding ...................................... 22  
7. Full Specification of the LDPC-Triangle Scheme ...... 24  
   7.1. General ....................................... 24  
   7.2. Parity Check Matrix Creation ................... 24  
   7.3. Encoding ...................................... 24  
   7.4. Decoding ...................................... 25  
8. Security Considerations ................................ 26  
10. Acknowledgments .................................... 28  
11. References ......................................... 29  
   11.1. Normative References ......................... 29  
   11.2. Informative References ....................... 29  
Appendix A. Trivial Decoding Algorithm (Informative Only) . 31  
Authors’ Addresses .................................... 32  
Intellectual Property and Copyright Statements .......... 33
1. Introduction

RFC 3453 [RFC3453] introduces large block FEC codes as an alternative to small block FEC codes like Reed-Solomon. The main advantage of such large block codes is the possibility to operate efficiently on source blocks of size several tens of thousands (or more) source symbols. The present document introduces the Fully-Specified FEC Encoding ID XX that is intended to be used with the "Low Density Parity Check" (LDPC) Staircase FEC codes, and the Fully-Specified FEC Encoding ID YY that is intended to be used with the "Low Density Parity Check" (LDPC)-Triangle FEC codes [Roca04][Mac03]. Both schemes belong the broad class of large block codes.

-- editor's note: This document makes use of the FEC Encoding ID values XX and YY that will be specified after IANA assignment --

LDPC codes rely on a dedicated matrix, called a "Parity Check Matrix", at the encoding and decoding ends. The parity check matrix defines relationships (or constraints) between the various encoding symbols (i.e. source symbols and repair symbols), that are later used by the decoder to reconstruct the original k source symbols if some of them are missing. These codes are systematic, in the sense that the encoding symbols include the source symbols in addition to the redundant symbols.

Since the encoder and decoder must operate on the same parity check matrix, some information must be communicated between them, as part of the FEC Object Transmission information.

A publicly available reference implementation of these codes is available and distributed under a GNU/LGPL license [LDPCrefimpl]. To the best of our knowledge, there is no patent or patent application identified as being used in the LDPC-Staircase and LDPC-Triangle FEC schemes.
2. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].
3. Definitions, Notations and Abbreviations

3.1. Definitions

This document uses the same terms and definitions as those specified in [fec-bb-revised]. Additionally, it uses the following definitions:

Encoding Symbol Group: a group of encoding symbols that are sent together, within the same packet, and whose relationships to the source object can be derived from a single Encoding Symbol ID.

Source Packet: a data packet containing only source symbols.

Repair Packet: a data packet containing only repair symbols.

3.2. Notations

This document uses the following notations:

L denotes the object transfer length in bytes

k denotes the source block length in symbols, i.e. the number of source symbols of a source block

n denotes the encoding block length, i.e. the number of encoding symbols generated for a source block

E denotes the encoding symbol length in bytes

B denotes the maximum source block length in symbols, i.e. the maximum number of source symbols per source block

N denotes the number of source blocks into which the object shall be partitioned

G denotes the number of encoding symbols per group, i.e. the number of symbols sent in the same packet

rate denotes the so-called "code rate", i.e. the k/n ratio

max_n Maximum Number of Encoding Symbols generated for any source block

srand(s) denotes the initialization function of the pseudo-random number generator, where s is the seed (s > 0)
rand(m) denotes a pseudo-random number generator, that returns a
new random integer in \([0; m-1]\) each time it is called

3.3. Abbreviations

This document uses the following abbreviations:

ESI: Encoding Symbol ID

FEC OTI: FEC Object Transmission Information
4. Formats and Codes

4.1. FEC Payload IDs

The FEC Payload ID is composed of the Source Block Number and the Encoding Symbol ID:

The Source Block Number (12 bit field) identifies from which source block of the object the encoding symbol(s) in the payload is(are) generated. There are a maximum of $2^{12}$ blocks per object.

The Encoding Symbol ID (20 bit field) identifies which encoding symbol(s) generated from the source block is(are) carried in the packet payload. There are a maximum of $2^{20}$ encoding symbols per block. The first $k$ values (0 to $k-1$) identify source symbols, the remaining $n-k$ values ($k$ to $n-k-1$) identify repair symbols.

There MUST be exactly one FEC Payload ID per packet. In case of an Encoding Symbol Group, when multiple encoding symbols are sent in the same packet, the FEC Payload ID refers to the first symbol of the packet. The other symbols can be deduced from the ESI of the first symbol thanks to a dedicated function, as explained in Section 5.5

```
+------------------+-+------------------+
|  Source Block Number |      Encoding Symbol ID (20 bits)     |
+------------------+-+------------------+
```

Figure 1: FEC Payload ID encoding format for FEC Encoding ID XX and YY

4.2. FEC Object Transmission Information

4.2.1. Mandatory Elements

- FEC Encoding ID: the Fully-Specified FEC Schemes described in this document use the FEC Encoding ID XX for LDPC-Staircase and FEC Encoding ID YY for LDPC-Triangle.

4.2.2. Common Elements

The following elements MUST be defined with the present FEC Scheme:

- Transfer-Length (L): a non-negative integer indicating the length of the object in bytes. There are some restrictions on the maximum Transfer-Length that can be supported:
maximum transfer length = 2^{12} * B * E

For instance, if B=2^{19} (because of a code rate of 1/2, Section 5.2), and if E=1024 bytes, then the maximum transfer length is 2^{41} bytes.

- Encoding-Symbol-Length (E): a non-negative integer indicating the length of each encoding symbol in bytes.

- Maximum-Source-Block-Length (B): a non-negative integer indicating the maximum number of source symbols in a source block. There are some restrictions on the maximum B value, as explained in Section 5.2.

- Max-Number-of-Encoding-Symbols (max_n): a non-negative integer indicating the maximum number of encoding symbols generated for any source block. There are some restrictions on the maximum max_n value. In particular max_n is at most equal to 2^{20}.

Section 5 explains how to derive the values of each of these elements.

4.2.3. Scheme-Specific Element

The following element MUST be defined with the present FEC Scheme. It contains two distinct pieces of information:

- G: a non-negative integer indicating the number of encoding symbols per group used for the object. The default value is 1, meaning that each packet contains exactly one symbol. Values greater than 1 can also be defined, as explained in Section 5.3.

- PRNG seed: The seed is a 32 bit value used to initialize the Pseudo Random Number Generator (defined in Section 5.6). This element is optional. Whether or not it is present in the FEC OTI will be signaled in the associated encoding format through an appropriate mechanism (see Section 4.2.4). When the PRNG seed is not carried within the FEC OTI, it is assumed that encoder and decoders use another way to communicate the information, or use a fixed, predefined value.

4.2.4. Encoding Format

This section shows two possible encoding formats of the above FEC OTI. The present document does not specify when or how these encoding formats should be used.
4.2.4.1. Using the General EXT_FTI Format

The FEC OTI binary format is the following, when the EXT_FTI mechanism is used.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   HET = 64    | HEL (=4 or 5) |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Transfer-Length (L)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Encoding Symbol Length (E)  |       G       |   B (MSB)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        B (LSB)        |   Max Nb of Enc. Symbols (max_n)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Optional PRNG seed                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

In particular:

- The HEL (Header Extension Length) indicates whether the optional PRNG seed is present (HEL=5) or not (HEL=4).
- The Maximum-Source-Block-Length (B) is split into two parts: the 8 most significant bits (MSB) are in the third 32-bit word of the EXT_FTI, and the remaining 12 least significant bits (LSB) are in fourth 32-bit word.

4.2.4.2. Using the FDT Instance (FLUTE specific)

When it is desired that the FEC OTI be carried in the FDT Instance of a FLUTE session, the following XML elements must be described for the associated object:

- FEC-OTI-Transfer-length
- FEC-OTI-Encoding-Symbol-Length
- FEC-OTI-Maximum-Source-Block-Length
- FEC-OTI-Max-Number-of-Encoding-Symbols
- FEC-OTI-Number-Encoding-Symbols-per-Group
- FEC-OTI-PRNG-seed (optional)

When no PRNG seed is to be carried in the FEC OTI, the sender simply
omits the FEC-OTI-PRNG-seed element.
5. Procedures

This section defines procedures that are common to FEC Encoding IDs XX and YY.

5.1. General

The B (maximum source block length in symbols) and E (encoding symbol length in bytes) parameters are first determined, as explained in the following sections.

The source object is then partitioned using the block partitioning algorithm specified in [fec-bb-revised]. To that purpose, the B, L (object transfer length in bytes), and E arguments are provided. As a result, the object is partitioned into N source blocks. These blocks are numbered consecutively from 0 to N-1. The first I source blocks consist of A_large source symbols, the remaining N-I source blocks consist of A_small source symbols. Each source symbol is E bytes in length, except perhaps the last symbol which may be shorter.

For each block the actual number of encoding symbols is determined, as explained in the following section.

Then, FEC encoding and decoding can be done block per block, independently. To that purpose, a parity check matrix is created, that forms a system of linear equations between the repair and source symbols of a given block, where the basic operator is XOR.

This parity check matrix is logically divided into two parts: the left side (from column 0 to k-1) which describes the occurrence of each source symbol in the equation system; and the right side (from column k to n-1) which describes the occurrence of each repair symbol in the equation system. An entry (a "1") in the matrix at position (i,j) (i.e. at row i and column j) means that the symbol with ESI i appears in equation j of the system. The only difference between the LDPC-Staircase and LDPC-Triangle schemes is the construction of the right sub-matrix.

When the parity symbols have been created, the sender will transmit source and parity symbols. The way this transmission occurs can largely impact the erasure recovery capabilities of the LDPC-* FEC. In particular, sending parity symbols in sequence is suboptimal. Instead it is usually recommended the shuffle these symbols. The interested reader will find more details in [Neumann05].

The following sections detail how the B, E, and n parameters are determined (respectively Section 5.2, Section 5.3 and Section 5.4), how encoding symbol groups are created (Section 5.5), and finally
5.2. Determining the Maximum Source Block Length (B)

The B parameter (maximum source block length in symbols) depends on several parameters: the code rate (rate), the Encoding Symbol ID field length of the FEC Payload ID (20 bits), as well as possible internal codec limitations.

The B parameter cannot be larger than the following values, derived from the FEC Payload ID limitations, for a given code rate:

\[
\text{max}_1B = 2^{20 - \text{ceil}(\log_2(1/\text{rate}))}
\]

Some common max1_B values are:

- rate == 1 (no repair symbols): \(\text{max}_B = 2^{20} = 1,048,576\)
- 1 > rate >= 1/2: \(\text{max}_1B = 2^{19} = 524,288\) symbols
- 1/2 > rate >= 1/4: \(\text{max}_1B = 2^{18} = 262,144\) symbols
- 1/4 > rate >= 1/8: \(\text{max}_1B = 2^{17} = 131,072\) symbols

Additionally, a codec MAY impose other limitations on the maximum block size. This is the case for instance when the codec uses internally 16 bit integers to store the Encoding Symbol ID, since it does not enable to store all the possible values of a 20 bit field. Other limitations may also apply, for instance because of a limited working memory size. This decision SHOULD be clarified at implementation time, when the target use case is known. This results in a max2_B limitation.

Then, B is given by:

\[
B = \min(\text{max}_1B, \text{max}_2B)
\]

Note that this calculation is only required at the coder, since the B parameter is communicated to the decoder through the FEC OTI.

5.3. Determining the Encoding Symbol Length (E) and Number of Encoding Symbols per Group (G)

The E parameter usually depends on the maximum transmission unit on the path (PMTU) from the source to the receivers. In order to minimize the protocol header overhead (e.g. the LCT/UDP/IPv4 or IPv6 headers in case of ALC), E is chosen as large as possible. In that case, E is chosen so that the size of a packet composed of a single
symbol (G=1) remains below but close to the PMTU.

Yet other considerations can exist. For instance, the E parameter can be made a function of the object transfer length. Indeed, LDPC codes are known to offer better protection for large blocks. In case of small objects, it can be a good practice to reduce the encoding symbol length (E) in order to artificially increase the number of symbols, and therefore the block size.

In order to minimize the protocol header overhead, several symbols can be grouped in the same Encoding Symbol Group (i.e. G > 1). Depending on how many symbols are grouped (G) and on the packet loss rate (which leads to loosing G symbols at a time), this strategy might or might not be appropriate. A balance must therefore be found.

The current specification does not mandate any value for either E or G. The current specification only provides an example of possible choices for E and G. Note that this choice is done by the sender. Then the E and G parameters are communicated to the receivers thanks to the FEC OTI.

Example:

First define the target packet size, pkt_sz (usually the PMTU minus the various protocol headers). The pkt_sz must be chosen in such a way it is a multiple of G. Calculate the number of packets: nb_pkts = ceil(L / pkt_sz). Then, use the following table to find a possible G value.

<table>
<thead>
<tr>
<th>Number of packets</th>
<th>G</th>
<th>Symbol size</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000 &lt;= nb_pkts</td>
<td>1</td>
<td>pkt_sz</td>
<td>4000 &lt;= k</td>
</tr>
<tr>
<td>1000 &lt;= nb_pkts &lt; 4000</td>
<td>4</td>
<td>pkt_sz / 4</td>
<td>4000 &lt;= k &lt; 16000</td>
</tr>
<tr>
<td>500 &lt;= nb_pkts &lt; 1000</td>
<td>8</td>
<td>pkt_sz / 8</td>
<td>4000 &lt;= k &lt; 8000</td>
</tr>
<tr>
<td>1 &lt;= nb_pkts &lt; 500</td>
<td>16</td>
<td>pkt_sz / 16</td>
<td>16 &lt;= k &lt; 8000</td>
</tr>
</tbody>
</table>

5.4. Determining the Number of Encoding Symbols of a Block

The following algorithm, also called "n-algorithm", explains how to determine the actual number of encoding symbols for a given block.

AT A SENDER:
Input:

B: Maximum source block length, for any source block. Section 5.2 explains how to determine its value.

k: Current source block length. This parameter is given by the source blocking algorithm.

rate: FEC code rate, which is provided by the user (e.g. when starting a FLUTE sending application). It is expressed as a floating point value. The rate value must be such that the resulting number of encoding symbols per block is at most equal to 2\(^{20}\) (Section 4.1).

Output:

max_n: Maximum number of encoding symbols generated for any source block

n: Number of encoding symbols generated for this source block

Algorithm:

\[
\text{max}_n = \text{floor}(B / \text{rate});
\]

if \((\text{max}_n \geq 2^{20})\) then return an error ("invalid code rate");

\[
n = \text{floor}(k \times \text{max}_n / B);
\]

AT A RECEIVER:

Input:

B: Extracted from the received FEC OTI

max_n: Extracted from the received FEC OTI

k: Given by the source blocking algorithm

Output:

n:

Algorithm:

\[
n = \text{floor}(k \times \text{max}_n / B);
\]
5.5. Identifying the Symbols of an Encoding Symbol Group

When multiple encoding symbols are sent in the same packet, the FEC Payload ID information of the packet MUST refer to the first encoding symbol. It MUST then be possible to identify each symbol from this single FEC Payload ID. To that purpose, the symbols of an Encoding Symbol Group (i.e. packet):

- MUST all be either source symbols, or repair symbols. Therefore only source packets and repair packets are permitted, not mixed ones.

- are identified by a function, ESIs_of_group(), that takes as argument:
  
  * for a sender, the index of the Encoding Symbol Group (i.e. packet) that the application wants to create,

  * for a receiver, the ESI information contained in the FEC Payload ID.

and returns the list of G Encoding Symbol IDs that will be packed together. In case of a source packet, the G source symbols are taken consecutively. In case of a repair packet, the G repair symbols are chosen randomly, as explained below.

The system must first be initialized by creating a random permutation of the n-k indexes. This initialization function MUST be called immediately after creating the parity check matrix. More precisely, since the PRNG seed is not re-initialized, no call to the PRNG function must have happened between the time the parity check matrix has been initialized and the time the following initialization function is called. This is true both at a sender and at a receiver.
initialize_tables ()
{
    int i;
    int randInd;
    int backup;

    /* initialize the two tables that map ID
     * (i.e. ESI-k) to/from TxSequence. */
    for (i = 0; i < n - k; i++) {
        IDtoTxseq[i] = i;
        txseqToID[i] = i;
    }

    /* now randomize everything */
    for (i = 0; i < n - k; i++) {
        randInd = rand(n - k);
        backup = IDtoTxseq[i];
        IDtoTxseq[i] = IDtoTxseq[randInd];
        IDtoTxseq[randInd] = backup;
        txseqToID[IDtoTxseq[i]] = i;
        txseqToID[IDtoTxseq[randInd]] = randInd;
    }
    return;
}

It is then possible, at the sender, to determine the sequence of G
Encoding Symbol IDs that will be part of the group.
/ * Determine the sequence of ESIs of the packet under construction
 * at a sender.
 * Warning: use only when G > 1.
 * PktIdx (IN): index of the packet, in \{0..\text{ceil}(n/G)\} range
 * ESIs[] (OUT): list of ESI of the packet
 */
sender_find_ESIs_of_group (int PktIdx,
                         ESI_t ESIs[])
{
    int i;
    if (is_source_packet(PktIdx) == true) {
        /* this is a source packet */
        ESIs[0] = (PktIdx * G) % k;
        for (i = 0; i < G; i++) {
            ESIs[i] = ESIs[0] + i;
        }
    }
    else {
        /* this is a repair packet */
        for (i = 0; i < G; i++) {
            ESIs[i] =
                k +
                txseqToID((i + (PktIdx - nbSourcePkts) * G)
                            % (n - k));
        }
    }
    return;
}

Similarly, upon receiving an Encoding Symbol Group (i.e. packet), a receiver can determine the sequence of G Encoding Symbol IDs from the first ESI, esi0, that is contained in the FEC Payload ID.
/*
 * Determine the sequence of ESIs of a packet received.
 * Warning: use only when G > 1.
 * esi0 (IN):  : ESI contained in the FEC Payload ID
 * ESIs[] (OUT): list of ESI of the packet
 */
receiver_find_ESIs_of_group (ESI_t    esi0,
                      ESI_t    ESIs[])
{
    int i;

    if (is_source_packet(esi0) == true) {
        /* this is a source packet */
        for (i = 0; i < G; i++) {
            ESIs[i] = (esi0 + i) % k;
        }
    } else {
        /* this is a repair packet */
        for (i = 0; i < G; i++) {
            ESIs[i] =
                k +
                txseqToID[(i + IDtoTxseq[esi0 - k])
                          % (n - k)];
        }
    }
}

5.6. Pseudo Random Number Generator

The present FEC Encoding ID relies on a pseudo-random number
generator (PRNG) that must be fully specified, in particular in order
to enable the receivers and the senders to build the same parity
check matrix. The minimal standard generator [Park88] is used. It
defines a simple multiplicative congruential algorithm: \( I_{j+1} = A \times I_j \) (modulo \( M \)), with the following choices: \( A = 7^{5} = 16807 \) and \( M = 2^{31} - 1 = 2147483647 \). Several implementations of this PRNG are
known and discussed in the literature. Yet all of them provide the
same sequence of pseudo random numbers. For instance, if seed = 1,
then the 10,000th value returned MUST be equal to 1043618065. The
following implementation uses the Park and Miller algorithm with the
optimization suggested by D. Carta in [Carta90].
unsigned long seed;

/*
 * Initialize the PRNG with a seed between
 * 1 and 0x7FFFFFFE (i.e. 2^{31}-2) inclusive.
 */
void srand (unsigned long s)
{
    if ((s > 0) && (s < 0x7FFFFFFF))
        seed = s;
    else
        exit(-1);
}

/*
 * Returns a random integer in [0; maxv-1]
 * Derived from rand31pmc, Robin Whittle,
 * September 20th, 2005.
 * 16807 multiplier constant (7^{5})
 * 0x7FFFFFF modulo constant (2^{31}-1)
 * The inner PRNG produces a value between 1 and
 * 0x7FFFFFFE (2^{31}-2) inclusive.
 * This value is then scaled between 0 and maxv-1
 * inclusive.
 */
unsigned long
rand (unsigned long maxv)
{
    unsigned long hi, lo;

    lo = 16807 * (seed & 0xFFFF);
    hi = 16807 * (seed >> 16);
    lo += (hi & 0x7FFF) << 16;
    lo += hi >> 15;
    if (lo > 0x7FFFFFFF)
        lo -= 0x7FFFFFFF;
    seed = (long)lo;
    /* don't use modulo, least significant bits are less random
     * than most significant bits [Numerical Recipies in C] */
    return ((unsigned long)
        ((double)seed * (double)maxv / (double)0x7FFFFFFF));
}
6. Full Specification of the LDPC-Staircase Scheme

6.1. General

The LDPC-Staircase scheme is identified by the Fully-Specified FEC Encoding ID XX.

The PRNG used by the LDPC-Staircase scheme must be initialized by a seed. This PRNG seed is an optional instance-specific FEC OTI element (Section 4.2.3). When this PRNG seed is not carried within the FEC OTI, it is assumed that encoder and decoders either use another way to communicate the seed value or use a fixed, predefined value.

6.2. Parity Check Matrix Creation

The LDPC-Staircase matrix can be divided into two parts: the left side of the matrix defines in which equations the source symbols are involved; the right side of the matrix defines in which equations the repair symbols are involved.

The left side is generated with the following algorithm:
/* initialize a list of possible choices to */
* guarantee a homogeneous "1" distribution */
for (h = 3*k-1; h >= 0; h--) {
    u[h] = h % (n-k);
}
/* left limit within the list of possible choices, u[] */
t = 0;

for (j = 0; j < k; j++) { /* for each source symbol column */
    for (h = 0; h < 3; h++) { /* add 3 "1s" */
        /* check that valid available choices remain */
        for (i = t; i < 3*k && matrix_has_entry(u[i], j); i++);
        if (i < 3*k) {
            /* choose one index within the list of possible *
             * choices */
            do {
                i = t + rand(3*k-t);
            } while (matrix_has_entry(u[i], j));
            matrix_insert_entry(u[i], j);
            /* replace with u[t] which has never been chosen */
            u[i] = u[t];
t++;
        } else {
            /* no choice left, choose one randomly */
            do {
                i = rand(n-k);
            } while (matrix_has_entry(i, j));
            matrix_insert_entry(i, j);
        }
    }
}

/* Add extra bits to avoid rows with less than two "1s" */
for (i = 0; i < n-k; i++) { /* for each row */
    if (degree_of_row(i) == 0) {
        j = rand(k);
        e = matrix_insert_entry(i, j);
    }
    if (degree_of_row(i) == 1) {
        do {
            j = rand(k);
        } while (matrix_has_entry(i, j));
        matrix_insert_entry(i, j);
    }
}
The right side (the staircase) is generated by the following algorithm:

```c
matrix_insert_entry(0, k);    /* first row */
for (i = 1; i < n-k; i++) {   /* for the following rows */
    matrix_insert_entry(i, k+i);   /* identity */
    matrix_insert_entry(i, k+i-1); /* staircase */
}
```

Note that just after creating this parity check matrix, when encoding symbol groups are used (i.e. \( G > 1 \)), the function initializing the two random permutation tables (Section 5.5) MUST be called. This is true both at a sender and at a receiver.

6.3. Encoding

Thanks to the staircase matrix, repair symbol creation is straightforward: each repair symbol is equal to the sum of all source symbols in the associated equation, plus the previous repair symbol (except for the first repair symbol). Therefore encoding MUST follow the natural repair symbol order: start with the first repair symbol, and generate repair symbol with ESI \( i \) before symbol ESI \( i+1 \).

6.4. Decoding

Decoding basically consists in solving a system of \( n-k \) linear equations whose variables are the source an repair symbols. Of course, the final goal is to recover the value of source symbols only.

To that purpose, many techniques are possible. One of them is the following trivial algorithm [Zyablov74]: given a set of linear equations, if one of them has only one remaining unknown variable, then the value of this variable is that of the constant term. So, replace this variable by its value in all the remaining linear equations and reiterate. The value of several variables can therefore be found recursively. Applied to LDPC FEC codes working over an erasure packet, the parity check matrix defines a set of linear equations whose variables are the source symbols and repair symbols. Receiving or decoding a symbol is equivalent to having the value of a variable. Appendix A sketches a possible implementation of this algorithm.

The Gauss elimination technique (or any derivative) is another possible decoding technique.

Because interoperability does not depend on the decoding algorithm used, the current document does not recommend any particular
technique. This choice is left to the codec developer.

Yet choosing a decoding technique will have great practical impacts. It will impact the erasure capabilities: a Gauss elimination technique enables to solve the system with a smaller number of symbols compared to the trivial technique. It will also impact the CPU load: a Gauss elimination technique requires much more processing than the trivial technique. Depending on the target use case, the codec developer will favor one feature or the other.
7. Full Specification of the LDPC-Triangle Scheme

7.1. General

LDPC-Triangle is identified by the Fully-Specified FEC Encoding ID YY.

The PRNG used by the LDPC-Triangle scheme must be initialized by a seed. This PRNG seed is an optional instance-specific FEC OTI element (Section 4.2.3). When this PRNG seed is not carried within the FEC OTI, it is assumed that encoder and decoders either use another way to communicate the seed value or use a fixed, predefined value.

7.2. Parity Check Matrix Creation

The LDPC-Triangle matrix can be divided into two parts: the left side of the matrix defines in which equations the source symbols are involved; the right side of the matrix defines in which equations the repair symbols are involved.

The left side is generated with the same algorithm as that of LDPC-Staircase (Section 6.2).

The right side (the triangle) is generated with the following algorithm:

```c
matrix_insert_entry(0, k);   /* first row */
for (i = 1; i < n-k; i++) {   /* for the following rows */
  matrix_insert_entry(i, k+i); /* identity */
  matrix_insert_entry(i, k+i-1); /* staircase */
  /* now fill the triangle */
  j = i-1;
  for (l = 0; l < j; l++) { /* limit the # of "1s" added */
    j = rand(j);
    matrix_insert_entry(i, k+j);
  }
}
```

Note that just after creating this parity check matrix, when encoding symbol groups are used (i.e. G > 1), the function initializing the two random permutation tables (Section 5.5) MUST be called. This is true both at a sender and at a receiver.

7.3. Encoding

Here also repair symbol creation is straightforward: each repair symbol is equal to the sum of all source symbols in the associated
equation, plus the repair symbols in the triangle. Therefore encoding MUST follow the natural repair symbol order: start with the first repair symbol, and generate repair symbol with ESI i before symbol ESI i+1.

7.4. Decoding

Decoding basically consists in solving a system of n-k linear equations, whose variables are the source an repair symbols. Of course, the final goal is to recover the value of source symbols only. To that purpose, many techniques are possible, as explained in Section 6.4.

Because interoperability does not depend on the decoding algorithm used, the current document does not recommend any particular technique. This choice is left to the codec implementer.
8. Security Considerations

The security considerations for this document are the same as that of [RFC3452].
9. Intellectual Property

To the best of our knowledge, there is no patent or patent application identified as being used in the LDPC-Staircase and LDPC-Triangle FEC schemes. Yet other LDPC codes and associated techniques MAY be covered by Intellectual Property Rights.
10. Acknowledgments

Section 5.4 is derived from a previous Internet-Draft, and we would like to thank S. Peltotalo and J. Peltotalo for their contribution. We would also like to thank Pascal Moniot, Laurent Fazio, Aurelien Francillon and Shao Wenjian for their comments.
11. References

11.1. Normative References


11.2. Informative References


Appendix A. Trivial Decoding Algorithm (Informative Only)

A trivial decoding algorithm is the following:

Initialization: allocate a partial sum buffer, partial_sum_i, for each line i, and reset it to 0.

For each newly received or decoded symbol s_i with ESI i:

1. If s_i is an already decoded or received symbol, return immediately and do nothing.

2. If s_i is a source symbol, it is permanently stored in memory.

3. For each equation j having a degree greater than one (i.e. more than one unknown variable), with an entry in column i (i.e. having s_i as a variable), do the following:
   + add s_i to partial_sum_i;
   + remove the entry (j, i) of the H matrix.
   + If the new degree of equation j is one, we have decoded a new packet and have to remember the index of the equation in a list of indexes for newly decoded packets for step 4.

4. For all newly generated packets s_l in step 3:
   + remove the last entry in equation j,
   + copy partial_sum_j to the buffer associate with symbol s_l,
   + goto step 1 with the newly created symbol s_l.
Authors’ Addresses

Vincent Roca  
INRIA  
655, av. de l’Europe  
Zirst; Montbonnot  
ST ISMIER cedex  38334  
France

Phone:  
Email: vincent.roca@inrialpes.fr  
URI: http://planete.inrialpes.fr/~roca/

Christoph Neumann  
Thomson Research  
46, Quai A. Le Gallo  
Boulogne Cedex  92648  
France

Phone:  
Email: christoph.neumann@thomson.net  
URI: http://planete.inrialpes.fr/~chneuman/

David Furodet  
STMicroelectronics  
12, Rue Jules Horowitz  
BP217  
Grenoble Cedex  38019  
France

Phone:  
Email: david.furodet@st.com  
URI:
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.