Low Density Parity Check (LDPC) Forward Error Correction (draft-roca-rmt-ldpc-00.txt)

Vincent Roca, Christoph Neumann, David Furodet

RMT
Internet-Draft
Expires: December 30, 2005

V. Roca
C. Neumann
INRIA
D. Furodet
STMicroelectronics

June 28, 2005

Low Density Parity Check (LDPC) Forward Error Correction
draft-roca-rmt-ldpc-00.txt

Status of this Memo

Copyright Notice

Abstract

This document describes an Under-Specified FEC Scheme that can be used with the broad class of Low Density Parity Check (LDPC) codes and their application to the reliable delivery of objects on packet erasure channels. Additionally, this document describes the LDPC-Staircase and LDPC-Triangle Forward Error Correction codes, two instances of the LDPC FEC Scheme, in a way that enables fully inter-

operable implementations. The LDPC codes belong to the class of large block FEC codes, as defined in RFC3453, which enables them to efficiently encode/decode large objects, in a single block. They also enable a receiver to recover the k source symbols from any set of a little bit more than k encoding symbols.

Table of Contents

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 several tens of thousands (or more) source symbols of size.

The present document introduces the Under-Specified FEC Encoding ID 132 that is intended to be used with the "Low Density Parity Check" (LDPC) FEC codes, that belong the class of large block codes. 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.

> -- editor's note: This document makes use of the FEC Encoding ID value 132, but this may change after IANA assignment --

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. Its content and the associated EXT_FTI are fully described in Section 4.2.

The two variants specified in this document belong to this broad class of LDPC codes. But other codes, existing or forthcoming, may also be added in the future, taking advantage of the framework provided by the Under-Specified FEC Encoding ID 132. More specifically, this document reserves the FEC Instance ID value 0 for the LDPC-Staircase codes [Roca04][Mac03] and reserves the FEC Instance ID value 1 for the LDPC-Triangle codes [Roca04]. A publicly available reference implementation of these codes is available and distributed under a GNU/LGPL license [LDPCrefimpl].

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 number of source symbols in a source block

n denotes the number of encoding symbols

E denotes the encoding symbol length in bytes

B denotes the maximum source block length in terms of symbols

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 per encoding block. This depends on FEC code rate.

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

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 identifies from which source block of the object the encoding symbol(s) in the payload is(are) generated.

The Encoding Symbol ID identifies which specific encoding symbol generated from the source block is carried in the packet payload. Each encoding symbol is either an original source symbol or a redundant symbol generated by the encoder.

There MUST be exactly one FEC Payload ID per packet. 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 as explained in Section 5.1

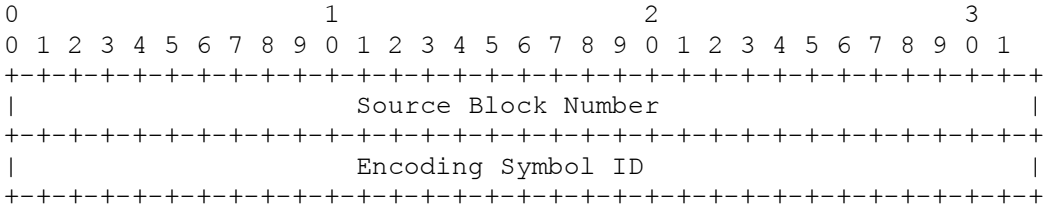


Figure 1: FEC Payload ID encoding format for FEC Encoding ID 132

4.2 FEC Object Transmission Information

4.2.1 Mandatory Elements

FEC Encoding ID: the Under-Specified FEC Scheme described in this document uses the FEC Encoding ID 132.

FEC Instance ID: this document reserves the FEC Instance ID value 0 for the LDPC-Staircase codes (Section 6) and the FEC Instance ID value 1 for the LDPC-Triangle codes (Section 7).

4.2.2 Common Elements

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

Transfer-Length: a non-negative integer indicating the length of the object in bytes.

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

Maximum-Source-Block-Length: a non-negative integer indicating the maximum number of source symbols in a source block.

Max-Number-of-Encoding-Symbols: a non-negative integer indicating the maximum number of encoding symbols (i.e. source plus repair symbols in the case of a systematic code).

Section 5.3 describes how to derive the values of each of these elements.

4.2.3 Scheme-Specific Elements

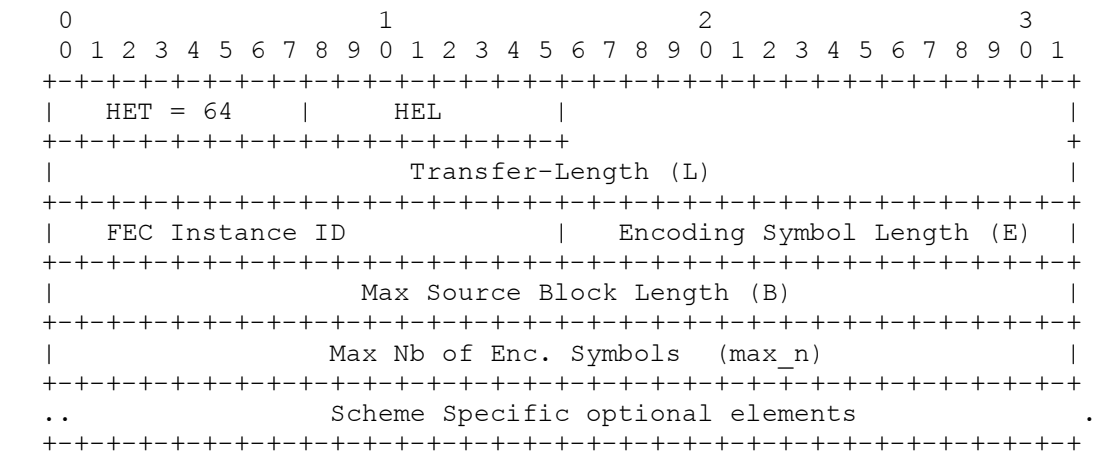
PRNG seed: Seed (a 32 bit value) used to initiate the Pseudo Random Generator (defined in Section 5.4). This element is optional and may be used by some specific Instance IDs.

Other elements MAY be defined for Instance-Specific needs.

4.2.4 Encoding Format

This section shows possible encoding formats of the above FEC OTI.

4.2.4.1 Using the General EXT_FTI Format



5. Procedures

This section defines procedures that are common to all FEC Instance IDs scoped by FEC Encoding ID 132.

5.1 General

The source object is first partitioned into blocks, using the block partitioning algorithm specified in [fec-bb-revised]. To that purpose, the B (maximum source block length in symbols), L (object transfer length in bytes), and E (encoding symbol length in bytes) arguments are provided. As an output, 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 as explained in [fec-bb-revised].

FEC encoding and decoding is done block per block, independently.

When multiple encoding symbols are sent in the same packet, it MUST 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 be in sequence, from ESI i to ESI i+G-1 (inclusive),
- MUST all be either source symbols, or repair symbols. Therefore, only source packets and repair packets are permitted, not mixed ones.

The FEC Payload ID information MUST refer to the first encoding symbol of the packet.

This specification does not specify what value for B should be used. This decision SHOULD be clarified either at implementation time, when the target use case is known, or in the specification of a FEC Instance ID, for instance to take into account some specificities of a FEC scheme.

Similarly, this specification does not specify if and when Encoding Symbol Groups should be used or not, i.e. if and when we have G>1. This decision SHOULD be clarified either at implementation time, when the target use case is known, or in the specification of a FEC Instance ID, for instance to take into account some specificities of a FEC scheme.

In both cases, a receiver can derive the B and G values from the

information it receives.

5.2 Parity Check Matrix

LDPC codes rely on a parity check matrix, which represents a linear equation system between repair symbols and source symbols of a given block. The basic operator is XOR and the matrix can only be filled with 1s and 0s.

The 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.

5.3 Derivations and Interpretation of the Fields Provided in the FPI and FEC OTI

The fields provided in the FEC OTI are derived using the "n-algorithm", described below:

AT A SENDER:

Input:

B Maximum Source Block Length, i.e., the maximum number of source symbols per source block. This is given by the FEC codec specifications and/or the execution environment limitations.

k Source Block Length, i.e., the number of source symbols per source block. This is given by source blocking algorithm.

rate or (k,n) FEC code rate, which is given by the user (e.g. when starting a FLUTE sending application). It is expressed either as a floating point value, R, or as a quotient k/n. The latter option is RECOMMENDED for the integer math version of the algorithm.

Output:

max_n Maximum Number of Encoding Symbols per encoding block. This depends on FEC code rate.

n Encoding Block Length, i.e., the number of encoding symbols generated for the source block.

Algorithm:

a. max_n = B / R rounded down to the nearest integer (max_n = (B * b) div a)

b. n = k * max_n / B rounded down to the nearest integer (n = (k * max_n) div B)

AT A RECEIVER:

Input: B, max_n, k

Output: n

Algorithm:

a. n = k * max_n / B rounded down to the nearest integer (n = (k * max_n) div B)

Notes: (1) X div Y denotes the integer quotient of the division X/Y

The use of floating point arithmetic in the algorithm might lead to erroneous results caused by rounding problems, depending on the mathematical library used. These problems can be avoided by using only integer math in all algorithm calculations. It is strongly recommended not to use rounding functions, and how to do that is presented in brackets

5.4 Pseudo Random Number Generator

The present FEC Encoding ID relies on a pseudo-random number generator that must be fully specified in order to enable the receivers and the senders to build the same parity check matrix.

-- editor's note: The PRNG to use is TBD. Current implementation relies on the GNU C Library lrand48() function, but this may not be the most appropriate choice. --

6. Full Specification of the LDPC-Staircase Scheme

6.1 Instance Specific Parameters

LDPC-Staircase is identified by th Under-Specified FEC Encoding ID 132 and the the FEC Instance ID 0.

LDPC-Staircase is based on a pseudo-random number generator as specified in Section 5.4. Therefore the seed used to initiate the PRNG is an instance-specific FEC Object Transmission Information element and MUST be transmitted within the FEC OTI, as specified in Section 4.2.

6.2 Parity Check Matrix Creation

The matrix creation algorithm for LDPC Staircase is described in the following. The algorithm can be divided into two parts: The left side of the matrix where the occurrence of the source symbols in the equations is described, and the right side of the matrix where repair symbols are described. 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 checks. */
      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 with the following algorithm:

```
for(i = 0; i < n-k; i++) { /* for each row */
    matrix_insert_entry(i,k+i);
    if (i > 0)
        matrix_insert_entry(i,k+i-1);
}
```

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 packet. Therefore encoding should follow the natural repair symbol order, i.e. generate repair symbol with ESI i before symbol ESI i+1.

6.4 Decoding

Decoding can be done using the general LDPC iterative decoding algorithm as described in Appendix A.

Other techniques can be used, for instance solving th system of n-k linear equations whose variables are the source an repair symbols

7. Full Specification of the LDPC-Triangle Scheme

7.1 Instance Specific Parameters

LDPC-Triangle is identified by th Under-Specified FEC Encoding ID 132 and the the FEC Instance ID 1.

LDPC-Triangle is based on a pseudo-random number generator as specified in Section 5.4. Therefore the seed used to initiate the PRNG is an instance-specific FEC Object Transmission Information element, and MUST be transmitted within the FEC OTI, as specified in Section 4.2.

7.2 Parity Check Matrix Creation

The matrix creation algorithm for LDPC Triangle is the following. The left side is the same as for LDPC Staircase (see Section 6.2). The right side (the triangle) is generated with the following algorithm:

```
for(i = 0; i < n-k; i++) { /* for each row */
    /* create the identity */
    matrix_insert_entry(i,k+i);
    if (i > 0) {
        /* create the staircase */
        matrix_insert_entry(i,k+i-1);

        /* fill the triangle */
        int j = i;
        for (l = 0; l < j; l++) {
            if (j != 0) {
                temp = rand() % j;
                matrix_insert_entry(pchkMatrix, i, k+j);
            }
        }
    }
}
```

7.3 Encoding

Just like LDPC-Triangle repair symbol creation is straightforward: each repair symbol is equal to the sum of all source symbols in the associated equation, plus some previous repair packets specified in the triangle. Encoding should follow the natural repair symbol order, i.e. generate repair symbol with ESI i before symbol ESI i+1.

7.4 Decoding

Decoding can be done using the general LDPC iterative decoding algorithm as described in Appendix A.

Other techniques can be used, for instance solving th system of n-k linear equation whose variables are the source an repair symbols

8. Security Considerations

The security considerations for this document are the same as they are for RFC 3452 [RFC3452].

9. Intellectual Property

The authors are not aware of any intellectual property rights associated to the two LDPC codes specified within this document. Yet other LDPC codes and associated techniques MAY be covered by IPR.

10. Acknowledgments

Section 5.3 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 from STMicroelectronics for his comments.

11. References

11.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 14, March 1997.

[RFC3452] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M., and J. Crowcroft, "Forward Error Correction (FEC) Building Block", RFC 3452, December 2002.

[RFC3453] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M., and J. Crowcroft, "The Use of Forward Error Correction (FEC) in Reliable Multicast", RFC 3453, December 2002.

[fec-bb-revised] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block (revised)", draft-ietf-rmt-fec-bb-revised-00.txt draft-ietf-rmt-fec-bb-revised-00.txt, April 2005.

11.2 Informative References

[LDPCrefimpl] Roca, V., Neumann, C., and J. Laboure, "LDPC-Staircase/ LDPC-Triangle Codec Reference Implementation", MCLv3 project PLANETE Research Team, INRIA Rhone-Alpes, June 2005.

[Mac03] MacKay, D., "Information Theory, Inference and Learning Algorithms", Cambridge University Press, ISBN: 0521642981, 2003.

[Roca04] Roca, V. and C. Neumann, "Design, Evaluation and Comparison of Four Large Block FEC Codecs: LDPC, LDGM, LDGM Staircase and LDGM Triangle, Plus a Reed-Solomon Small Block FEC Codec", INRIA Research Report RR-5225, June 2004.

Authors' Addresses

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

Phone:
Email: vincent.roca@inrialpes.fr
URI:

Christoph Neumann
INRIA
655, av. de l'Europe
Zirst; Montbonnot
ST ISMIER cedex 38334
France

Phone:
Email: christoph.neumann@inrialpes.fr
URI:

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

Phone:
Email: david.furodet@st.com
URI:

Appendix A. Iterative Decoding Algorithm (Informative)

LDPC decoding over a packet erasure channel can be achieved through a trivial iterative decoding algorithm. The underlying idea is the following:

> 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 remaining linear equations, and reiterate. The value of several variables can therefore be found by this recursive algorithm.

Applied to LDPC FEC codes working over an erasure packet, the parity check matrix defines a set of linear equations. The variables are the source symbols and repair symbols. Of course, from a decoding point of view, finding (i.e. decoding) all source symbols is the target. Finding repair symbols is often required to that purpose, but this is not the final goal. The iterative decoding algorithm is the following:

> Initialization: allocate a partial sum buffer partial_sum_i for each line i: set 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 in step 3:
>
>   + remove the last entry in equation j,

+ move partial_sum_j into th buffer of symbol s_l,

+ goto step 1 with the newly created symbol s_l

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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