Reliable Multicast Transport
Internet-Draft
Expires: August 17, 2005

M. Luby
Digital Fountain
A. Shokrollahi
EPFL
M. Watson
Digital Fountain
February 13, 2005

Raptor Forward Error Correction
draft-ietf-rmt-bb-fec-raptor-object-00

Status of this Memo

Copyright Notice

Abstract

This document describes the systematic Raptor forward error correction code and its application to reliable delivery of data objects.

Raptor is a fountain code, i.e., as many encoding symbols as needed can be generated by the encoder on-the-fly from the source symbols of a source block of data. The decoder is able to recover the source block from any set of encoding symbols only slightly more in number than the number of source symbols.

The Raptor code described here is a systematic code, meaning that the first encoding symbols generated are equal to the source symbols.

1. Introduction

This document specifies the Raptor forward error correction code and its application to reliable delivery of data objects. Raptor is a fountain code, i.e., as many encoding symbols as needed can be generated by the encoder on-the-fly from the source symbols of a block. The decoder is able to recover the source block from any set of encoding symbols only slightly more in number than the number of source symbols.

The code described in this document is a Systematic code, that is, the original source symbols are sent unmodified from sender to receiver, as well as a number of repair symbols.

Raptor code aspects which are specific to reliable delivery of objects are discussed in Section 4 of this document.

The principle component of the systematic Raptor code is the basic encoder described in Section 5. This encoder produces encoding symbols which are each the exclusive OR of a number of intermediate pre-coding symbols. These encoding symbols are produced in such a way that the intermediate pre-coding symbols can be recovered from any sufficiently large set of encoding symbols.

Section 6 describes how to derive values for the intermediate pre-coding symbols from the original source symbols in such a way that a systematic code is obtained. At the decoder, original source symbols can then be used along with repair symbols in the basic decoder process to re-construct the intermediate pre-coding symbols. The missing source symbols can then be easily derived from the intermediate pre-coding symbols.

This document defines the Raptor code encoder. A number of possible decoding algorithms are possible. An efficient decoding algorithm is provided in Appendix A.

The construction of the encoding symbols is based in part on a pseudo-random number generator described in Section 5.3.1. This generator is based on a fixed set of 512 random numbers which must be available to both sender and receiver. These are provided in Appendix C

Finally, the construction of the intermediate pre-coding symbols from the source symbols is governed by a \202Çÿsystematic index\202ÇÖ, values of which are to be provided.

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 [1].

3. Definitions, Symbols and abbreviations

3.1 Definitions

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

Source block a block of K source symbols which are considered together for Raptor encoding purposes.

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

Encoding symbol the smallest unit of data generated during the encoding process. Encoding symbols generated from a source block have the same size as the source symbols of that source block. For a systematic code, encoding symbols consist of source symbols and repair symbols.

Systematic code a code in which the source symbols are included as part of the encoding symbols sent for a source block.

Repair symbol for a systematic code, the encoding symbols sent for a source block that are not the source symbols, i.e., the repair symbols are generated based on the source symbols.

Pre-coding symbols a set of symbols calculated at the start of the encoding process for a source block, which are subsequently used to generate encoding symbols.

Intermediate pre-coding symbols for the Systematic Raptor Code, symbols generated from the source symbols using the non-systematic Raptor decoder. The repair symbols are then generated directly from the intermediate pre-coding symbols.

Symbol a unit of data. The size, in bytes, of a symbol is known as the symbol size.

Encoding symbol group a group of encoding symbols that are sent together, i.e., within the same packet whose relationship to the source symbols can be derived from a single Encoding Symbol ID.

Encoding Symbol ID information that defines the relationship between the symbols of an encoding symbol group and the source symbols.

Sub-block a source block is sometime broken into sub-blocks, each of which is sufficiently small to be decoded in working memory. For a source block consisting of K source symbols, each sub-block is consists of K sub-symbols, each symbol of the source block being composed of one sub-symbol from each sub-block.

Sub-symbol a subset of a source symbol. Each source symbol is composed of as many sub-symbols as there are sub-blocks in the source block.

Source packet for a systematic code, data packets that contain only source symbols.

Repair packet for a systematic code, data packets that contain only repair symbols.

3.2 Symbols

i, j, x, h, a, b, d, v, m positive integers
ceil(x) denotes the smallest positive integer which is greater than or equal to x
choose(i,j) denotes the number of ways j objects can be chosen from among i objects without repetition
floor(x) denotes the largest positive integer which is less than or equal to x
i % j denotes i modulo j
X ^ Y denotes, for equal-length bit strings X and Y, the bitwise exclusive-or of X and Y
K denotes the number of symbols in a single source block
L denotes the number of pre-coding symbols for a single source block
S denotes the number of LDPC symbols for a single source block
H denotes the number of Half symbols for a single source block
C denotes an array of symbols, C[0], C[1], C[2],...
X a two-byte integer value
X0,X1 the high-order and low-order bytes of X, respectively
V0,V1 two arrays of 4-byte integers, V0[0], V0[1],..., V0[255] and V1[0], V1[1],..., V1[255]
Rand[X, i, m] a pseudo-random number generator
Deg[v] a degree generator
Enc[K, C ,d, a, b] an encoding symbol generator
B the maximum size of a source block, in bytes
W the maximum size of a sub-block, in bytes, which can be decoded in working memory
G the number of symbols within an encoding symbol group
N the number of sub-blocks within a source block
T the symbol size in bytes. If the source block is partitioned into sub-blocks, then T = T'*N
T' the sub-symbol size, in bytes. If the source block is not partitioned into sub-blocks then TÃÆ is not relevant. If the source block is partitioned into N > 1 sub-blocks then T' = T/N.
F the file size, for file download, in bytes
I the sub-block size in bytes
P for file download, the payload size of each packet, in bytes. For streaming, the payload size of each repair packet, in bytes. P can be written as P = J*2^^p, where J is an odd positive integer and p is a positive integer.
Q Q = 65521, i.e., Q is the largest prime smaller than 2^^16
Z the number of source blocks, for file download

a ^^ b a raised to the power b

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:
ESI Encoding Symbol ID
LDPC Low Density Parity Check
SBN Source Block Number
SBL Source Block Length (in units of symbols)

4. File delivery

4.1 Source block construction

4.1.1 General

In order to apply the Raptor encoder to a source file, the file is first broken into Z > 0 blocks, known as source blocks. The Raptor encoder is applied independently to each source block. The source blocks are sometimes further divided into N > 1 sub-blocks, which are small enough to be decoded in working memory.

Each source block is divided into symbols of size T bytes and each sub-block into sub-symbols of size T' = T/N bytes as described below. The source file size F, the encoding packet size, P, as well as the parameters Z, N, T are obtained from the transport protocol as described in Section 4.

The construction of the source blocks and sub-blocks is controlled by the following parameters:

- The size of encoding packet payloads P in bytes. P can be written as P = J*(2^^p), where J is an odd positive integer and p is a positive integer. Preferably, J is as small as possible and p is as large as possible. Examples are P = 2^^9 = 512, P = 5*(2^^8) = 1280 and P = 11*(2^^7) = 1408.
- The maximum source block size B in bytes
- The maximum sub-block size W in bytes, such that W <= B and ceil(B/W) <= 2^^p

Source blocks may be partitioned into N > 1 sub-blocks. Each sub-block consists of the same number K of sub-symbols, where each sub-symbol is T' bytes long. Then, each source symbol of the source block is T = T'N bytes long, and consists of the concatenation of exactly one sub-symbol from each of the N sub-blocks.

Table 1 shows a source block placed into a two dimensional array, where each entry is a T-byte sub-symbol, each row is a sub-block and each column is a source symbol. The number shown in each sub-symbol entry indicates their original order within the source block. For example, the sub-symbol numbered K contains bytes T'*K through T'*(K+1)-1 of the source block. Then, source symbol i is the concatenation of the ith sub-symbol from each of the sub-blocks, which corresponds to the sub-symbols of the source block numbered i, K+i, 2Ã¸K+i,..., (N-1)K+i.

0	1	2	...	...	...	...	K-1
K	K+1	K+2	...	...	...	...	2K-1
2K	2K+1	2K+2	...	...	...	...	3K-1
...	...	...	...	...	...	...	...
(N-1)K	...	2	...	...	...	...	NK-1

Table 1: Source symbols from sub-symbols - each column represents a source symbol

Each source block is identified by a unique Source Block Number (SBN), with the first source block numbered zero, the second numbered one, etc.

4.1.2 Small files (single sub-block)

When the file consists of a single source block that in turn consists of one sub-block, i.e., Z = N = 1. The number of symbols per packet, G, and the number of source symbols, K, are computed as follows:

Let P = J(2^^p) be the packet payload size in bytes. Then, G = P/T.

It is recommended that the symbol size T = P/G, is derived as follows:

If ceil(F/P) > 2048, then let G = 1. Otherwise, let G = 2^^j where j is the largest non-negative integer such that j <= p and G*ceil(F/P) <= 2048 and T >= 32.

The number K of source symbols in the file is computed as ceil(F/T). If K.T > F then the last symbol is padded with zero bytes for the purposes of encoding.

Recommended values of T, G and K are shown for by way of example for in Table 2 below:

F range	G	T	K range
512 bytes < F <= 64 KB	16	32 bytes	16 <= K <= 2048
64 KB < F <= 128 KB	8	64 bytes	1024 < K <= 2048
128 KB < F <= 256 KB	4	128 bytes	1024 < K <= 2048

Table 2: Source block parameters for small files with P = 512 bytes

4.1.3 Larger files (single source block)

When the file consists of a single source block that is partitioned into N sub-blocks, where N = 2^^n, ,the size I of each sub-block is computed as ceil(F/N).

Let P = J*2^^p be the packet payload size in bytes and let P' = P/N = J*2^^(p-n) be the size in bytes of each packet payload dedicated to each of the N sub-blocks. Then, the sub-symbol size T' = P'/G, where G=P/T.

It is recommended that the number of sub-blocks, N and symbol size T are determined based on the available working memory for encoding symbols, W, as follows:

N = 2^^n , where n is the smallest positive integer such that N*W >= F and W is the maximum working memory size,

T = P/G whereif ceil(I/P') > 4096, then let G = 1. Otherwise, let G = 2^^j where j is the largest non-negative integer such that j <= p-n and G*ceil(I/P') <= 4096 and T' >= 32.

The number K of sub-symbols per sub-block is computed as ceil(I/T'), and K is also the number of source symbols in the source block.

The recommended values of N, G, T' and K are shown, by way of example, for P = 512 bytes and W=256KB, in Table 3 below.

F range	N	G	T	K range
256 KB < F <= 512 KB	2	4	64 bytes	2048 < K <= 4096
512 KB < F <= 1 MB	4	2	64 bytes	2048 < K <= 4096
1 MB < F <= 2 MB	8	1	64 bytes	2048 < K <= 4096
2 MB < F <= 4 MB	16	1	32 bytes	4096 < K <= 8192

Table 3: Source block parameters for large files when P = 512 bytes, B = 4 MB and W = 256 KB

4.1.4 Large files (multiple source blocks)

When the file is partitioned into more than one source block then for each source block, the algorithms described in Section 4.1.3 are applied independently to determine the sub-block structure.

It is recommended that the number of source blocks, Z, is determined using the Algorithm for Computing Source Block Structure described in Section 5.1.2.3 of FLUTE [4]. The inputs to that algorithm are:

- The maximum number of source symbols per source block. This is set to ceil(B/P).
- The transfer length in bytes. This is set to the file size F.
- The symbol length in bytes. This is set to P since there is one symbol per packet.

The output of the algorithm is the number Z of source blocks, and the number and lengths of source symbols in each of the Z source blocks (with possibly the last symbol of the last source block logically filled out with zeroes to a full length symbol).

4.2 Encoding packet construction

4.2.1 General

Each encoding packet contains the following information:

- FEC Payload ID, consisting of two integers.
- Source Block Number (SBN) - two bytes
- Encoding Symbol ID (ESI) - two bytes
- G encoding symbols

Each source block is encoded independently of the others. Source blocks are numbered consecutively from zero.

4.2.2 Encoding packet construction

Each encoding packet either consists entirely of source symbols (source packet) or entirely of repair symbols (repair packet).

Each packet contains G[i] <= G source symbols, where G[i] is the number of symbols in the ith packet. In the case of file delivery, all source packets except possibly the last contain exactly G symbols. The last source packet may be of shorter length if K is not a multiple of G. The source symbols are placed into source packets sequentially, i.e., the first source packet contains the first G[1] source symbols, the second source packet contains the next G[2] source symbols, etc. The ESI carried in each source packet is the index of the first source symbol carried in that packet where the indices of the source symbols are 0,...,K-1.

Similarly, the ESI values placed into the repair packets are the indices of the first repair symbol in each repair packet. Note that it is not necessary for the receiver to know the number of repair packets. The Gi repair symbol triples (d[0], a[0], b[0]),Ãà, (d[Gi-1], a[Gi-1], b[Gi-1]) for the repair symbols placed into a repair packet with ESI X are computed as follows:

Let,
A and B be the values derived from the systematic index for K source symbols described in Section 8.2.
Q = 65521, the largest prime smaller than 2^^16
Then,
1. Y = (B + X*A) % Q
2. v = Rand[Y, 0, 220]
3. Repeat the following for j = 0,...Ãà,G[i]-1
   A. d[j] = Deg[v]
   B. a[j] = 1 + Rand[Y, 1, L'-1]
   C. b[j] = Rand[Y, 2, L']
   D. v = (v + ceil((2^^20)/G[i])) % 2^^20
   E. Y = (Y + A) % Q

The G[i] repair symbols to be placed in repair packet with ESI X are calculated based on the G[i] repair symbol triples as described in Section 6.5

4.3 Transport

This section describes the information exchange between the Raptor encoder/decoder and any transport protocol making use of Raptor forward error correction for file delivery. The Raptor encoder and decoder for file delivery require the following information from the transport protocol:

- file size, F, in bytes
- packet payload size, P, in bytes
- symbol size, T, in bytes
- number of source blocks, N
- number of sub-blocks in each source block, Z

The Raptor encoder for file delivery additionally requires:
- the data to be encoded, F bytes

The Raptor encoder supplies the transport protocol with encoding packet information consisting, for each packet, of:
- Source Block Number, 2 bytes
- Encoding Symbol ID, 2 bytes
- Encoding Symbols, P bytes

The transport protocol shall communicate this information transparently to the Raptor decoder.

The mapping of this information into the FLUTE protocol [3] is tbd.

5. Basic encoder operation

5.1 Basic Encoding overview

This section describes the Basic Encoding process by which repair symbols are generated from a set of intermediate symbols.

Symbols are the fundamental data units of the encoding and decoding process. For each source block (sub-block) all symbols (sub-symbols) are the same size. The atomic operation performed on symbols (sub-symbols) for both encoding and decoding is the exclusive-or operation.

A pre-coding step is used to produce L-K redundant symbols from the K intermediate symbols, where L > K. The combination of the K intermediate symbols and the L-K redundant symbols form the L pre-coding symbols. Section 5.2 describes how the pre-coding symbols are generated from the intermediate symbols.

Encoding symbols are produced in groups, and each group is mapped into a single data packet. Each encoding symbol group is associated with an Encoding Symbol ID (ESI) and a number, G, of encoding symbols. The ESI is used to generate three integers, d, a and b, known as a triple for each encoding symbol within the encoding symbol group. This is done as described in Sections 5 and 6 using the generators described in Section 5.3.1 and Section 5.3.2 Then, each (d,a,b)-triple is used to generate the corresponding encoding symbol from the pre-coding symbols using the Enc[] generator described in Section 5.3.3

Let C[0],..., C[K-1] denote the K intermediate symbols.

5.2 Pre-coding

The pre-coding step consists of generating redundant symbols from the K intermediate symbols as follows. The redundant symbols consist of S LDPC symbols and H Half symbols.

Let
X be the smallest positive integer such that X*(X-1) = 2*K.
S be the smallest prime integer such that S >= ceil(0.01*K) + X
H be the smallest integer such that choose(H,ceil(H/2)) >= K + S
H' = ceil(H/2)
L = K+S+H
C[K],..., C[K+S-1] denote the S LDPC symbols, initialised to zero
C[K+S],..., C[L-1] denote the H Half symbols, initialised to zero

The S LDPC symbols are defined as follows:

```
For i = 0,...,K-1 do
   a = 1 + (floor(i/S) % (S-1))
   b = i % S
   C[K + b] = C[K + b] ^ C[i]
   b = (b + a) % S
   C[K + b] = C[K + b] ^ C[i]
   b = (b + a) % S
   C[K + b] = C[K + b] ^ C[i]
```

The H Half symbols are defined as follows:

Let
g[i] = i ^ (floor(i/2)) for all positive integers i
g[i,j] denote the ith element of the subsequence of g[i] whose elements have exactly j non-zero bits in their binary representation

Then, the Half symbols are generated using the following algorithm:

```
For h = 0,...,H-1 do
   For i = 0,...,K+S-1 do
      If bit h of g[i,H'] is equal to 1 then C[h+K+S] = C[h+K+S] ^
      C[i].
```

5.3 Generators

All of the generators described in the following subsections are used in the generation of encoding symbols.

5.3.1 Random Generator

The random number generator Rand[X, i, m] is defined as follows, where X is a two-byte value, i is a non-negative integer and m is a positive integer and the value produced is an integer between 0 and m-1. Let X0 be the high-order byte and let X1 be the low-order byte of X. Let V0 and V1 be arrays of 256 entries each, where each entry is a 4-byte unsigned integer.

Editor's note: The entries of V0 and V1 must be random or pseudo-random and both encoder and decoder require access to the same arrays. These arrays are provided in Appendix C to this document.

Then,

```
Rand[X, i, m] = (V0[(X0 + i) % 256] ^ V1[(X1 + i) % 256]) % m
```

5.3.2 Degree Generator

The degree generator Deg[v] is defined as follows, where v is an integer that is at least 0 and less than 2^^20 = 1048576.

In Table 4, find the index j such that f[j-1] <= v < f[j]
then, Deg[v] = d[j]

Index j	f[j]	d[j]
0	0	--
1	10241	1
2	491582	2
3	712794	3
4	831695	4
5	948446	10
6	1032189	11
7	1048576	40

Table 4: Defines the degree distribution for encoding symbols

5.3.3 Encoding Symbol Generator

The encoding symbol generator Enc[K, C, d, a, b] takes the following inputs:

K is the number of intermediate symbols (or sub-symbols) for the source block (sub-block). Let L be derived from K as described in Section 5.2, and let L\202ÇÖ be the smallest prime integer greater than or equal to L.

C is the array of L pre-coding symbols (sub-symbols) generated as described in Section 5.2.

d is an integer denoting an encoding symbol degree

a is an integer between 1 and L'-1 inclusive

b is an integer between 0 and L'-1 inclusive

The encoding symbol generator produces a single encoding symbol as output, according to the following algorithm:

```
While (b >= L) do b = (b + a) % L'
Enc[K, C, d, a, b] = C[b].
For j = 1,...,d-1 do
   b = (b + a) % L'
   While (b >= L) do b = (b + a) % L'
```

Enc[K, C, d, a, b] = Enc[K, C, d, a, b] ^ C[b]

6. Systematic encoder

6.1 Encoder overview

The Raptor systematic encoder is used to generate repair symbols from a source block that consists of K source symbols. The first step is to obtain systematic index associated with K. This systematic index is used to generate K triples (d[0], a[0], b[0]),..., (d[K-1], a[K-1], b[K-1]) which are then associated with the K source symbols. The K source symbol triples are then used to decode L intermediate pre-coding symbols from the source symbols using a Raptor decoder. By the way the systematic index is generated, the K source symbol triples are guaranteed to uniquely decode the L intermediate pre-coding symbols.

The Raptor basic encoder described in Section 5 is then used to produce encoding symbols, in this context called repair symbols, from the intermediate pre-coding symbols. The source symbols and the repair symbols are then sent to receivers.

Let C'[0],..., C'[K-1] denote the K source symbols.

6.2 Systematic index

The systematic index for each relevant value of K is pre-stored at each receiver. The systematic index associated with a source block of K source symbols consists of a single integer value.

Editor's Note: Systematic indices for various values of K are provided in Annex B to this TS.

If a systematic index is not defined in Annex B for the required value of K, then let K' be equal to K and modify K to be the smallest value greater than K' for which a systematic index is defined. For the purposes of encoding and decoding, the source block is first extended by appending (K-K') additional source symbols. These symbols consist of bytes with value zero. Note that these additional source symbols are not sent from encoder to decoder. Whenever the source block size must be communicated from encoder to decoder, the original size, K', shall be used and the decoder shall derive the expanded source block size, K, by itself.

6.3 Source symbol triples

Each of the K source symbols is associated with a triple (d[i], a[i], b[i]) for 0 <= i < K.

Let

```
L be determined from K as described in Section 5
L' be the smallest prime that is greater than or equal to L
Q = 65521 is the largest prime smaller than 2^^16.
```

The triples associated with source symbols are generated from the systematic index, J(K), associated with K as follows:

```
1.  A = (53591+997*J(K)) % Q
2.  B = 10267*(J(K)+1) % Q
3.  i = 0,  j = 0, t = 0, X = B
4.  While i < K do
    A.  If t < T and j = M[t] then t = t+1
    B.  Else
           i.       v = Rand[X, 0, 220]
           ii.      d[i] = Deg[v]
           iii.     a[i] = 1 + Rand[X, 1, L'-1]
           iv.      b[i] = Rand[X, 2, L']
           v.       i = i+1
    C.  X = (X + A) % Q
    D.  j = j + 1
```

6.4 Generating intermediate symbols

The L intermediate pre-coding symbols C[0], C[1],..., C[L-1] are the uniquely defined symbol values that satisfy the following condition: The K source symbols C'[0], C'[1],..., C'[K-1] satisfy the K constraints C'[i] = Enc[K, C, d[i], a[i], b[i]), for 0 <= i < K.

A decoding process can be applied to the K source symbols C'[0], C'[1],..., C'[K-1] to produce the L intermediate pre-coding symbols C[0], C[1],..., C[L-1]. For each value of K the systematic index is designed to have the property that the source symbol triples generated from the systematic index as described in Section 6.3 are guaranteed to uniquely decode the L intermediate pre-coding symbols using Gaussian elimination. To efficiently decode the intermediate pre-coding symbols from the source symbols, it is recommended that an efficient decoder implementation such as that described in Annex A be used. The source symbol triples are designed to facilitate efficient decoding of the source symbols using that algorithm.

6.5 Generating repair symbols

Repair symbols are generated using the basic encoder described in Section 5, applied to the L intermediate pre-coding symbols C[0], C[1],..., C[L-1].

7. Security Considerations

The security considerations for this document are the same as they are for [2].

8. Intellectual Property

Digital Fountain does have intellectual property rights associated with the technology described in this document, and intends to provide a full IPR statement specific to this document to the IETF that will satisfy the requirements of the IETF.

9. References

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

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

[3] Paila, T., Luby, M., Lehtonen, R., Roca, V. and R. Walsh, "FLUTE - File Delivery over Unidirectional Transport", RFC 3926, October 2004.

Authors' Addresses

Michael Luby
Digital Fountain
39141 Civic Center Drive
Suite 300
Fremont, CA 94538
U.S.A.

Email: luby@digitalfountain.com

Amin Shokrollahi
EPFL
Laboratory of Algorithmic Mathematics
IC-IIF-ALGO
PSE-A
Lausanne 1015
Switzerland

Email: amin.shokrollahi@epfl.ch

Mark Watson
Digital Fountain
39141 Civic Center Drive
Suite 300
Fremont, CA 94538
U.S.A.

Email: mark@digitalfountain.com

Appendix A. Decoder (Informative)

A.1 General

It is assumed that the decoder knows the structure of the source block it is to decode, including the symbol size, T, and the number K of symbols in the source block. From the algorithms described in Sections 7 and 8, the Raptor decoder can calculate the total number L = K+S+H of pre-coding symbols and determine how they were generated from the source block to be decoded. In this description it is assumed that the received encoding symbols for the source block to be decoded are passed to the decoder. Furthermore, for each such encoding symbol it is assumed that the number and set of intermediate pre-coding symbols that were exclusive-ored to calculate the encoding symbol is passed to the decoder. Additionally, the source symbol triples described in Section 8.3 indicate the number and set of intermediate pre-coding symbols which sum to give each source symbol. This information is dependent upon the particular application. How this information is obtained for file delivery is described in Section 4

Let N >= K be the number of received encoding symbols for a source block and let M = S+H+N. The following M by L bit matrix A can be derived from the information passed to the decoder for the source block to be decoded. Let C be the column vector of the L intermediate pre-coding symbols, and let D be the column vector of M symbols with values known to the receiver, where the first S+H of the M symbols are zero-valued symbols that correspond to LDPC and Half symbols (these are check symbols for the LDPC and Half symbols, and not the LDPC and Half symbols themselves), and the remaining N of the M symbols are the received encoding symbols for the source block. Then, A is the bit matrix that satisfies A*C = D, where here * denotes matrix multiplication over GF[2]. In particular, A[i,j] = 1 if the pre-coding symbol corresponding to index j is exclusive-or'd into the LDPC, Half or encoding symbol corresponding to index i in the encoding, or if index i corresponds to a LDPC or Half symbol and index j corresponds to the same LDPC or Half symbol. For all other i and j, A[i,j] = 0.

Decoding a source block is equivalent to decoding C from known A and D. It is clear that C can be decoded if and only if the rank of A over GF[2] is L. Once C has been decoded, missing source symbols can be obtained by using the source symbol triples to determine the number and set of intermediate pre-coding symbols which must be exclusive-ORed to obtain each missing source symbol.

The first step in decoding C is to form a decoding schedule. In this step A is converted, using Gaussian elimination (using row operations

and row and column reorderings) and after discarding M \202Çô L rows, into the L by L identity matrix. The decoding schedule consists of the sequence of row operations and row and column re-orderings during the Gaussian elimination process, and only depends on A and not on D. The decoding of C from D can take place concurrently with the forming of the decoding schedule, or the decoding can take place afterwards based on the decoding schedule.

The correspondence between the decoding schedule and the decoding of C is as follows. Let c[0] = 0, c[1] = 1,...,c[L-1] = L-1 and d[0] = 0, d[1] = 1,...,d[M-1] = M-1 initially.

Each time row i of A is exclusive-ored into row i' in the decoding schedule then in the decoding process symbol D[d[i]] is exclusive-ored into symbol D[d[i']].

Each time row i is exchanged with row i' in the decoding schedule then in the decoding process the value of d[i] is exchanged with the value of d[i'].

Each time column j is exchanged with column j' in the decoding schedule then in the decoding process the value of c[j] is exchanged with the value of c[j'].

From this correspondence it is clear that the total number of exclusive-ors of symbols in the decoding of the source block is the number of row operations (not exchanges) in the Gaussian elimination. Since A is the L by L identity matrix after the Gaussian elimination and after discarding the last M \202Çô L rows, it is clear at the end of successful decoding that the L symbols D[d[0]], D[d[1]],..., D[d[L-1]] are the values of the L symbols C[c[0]], C[c[1]],..., C[c[L-1]].

The order in which Gaussian elimination is performed to form the decoding schedule has no bearing on whether or not the decoding is successful. However, the speed of the decoding depends heavily on the order in which Gaussian elimination is performed. (Furthermore, maintaining a sparse representation of A is crucial, although this is not described here). The remainder of this section describes an order in which Gaussian elimination could be performed that is relatively efficient.

A.2 First Phase

The first phase of the Gaussian elimination the matrix A is conceptually partitioned into submatrices. The submatrix sizes are parameterized by non-negative integers i and u which are initialized to 0. The submatrices of A are:

(1) The submatrix I defined by the intersection of the first i rows and first i columns. This is the identity matrix at the end of each step in the phase.
(2) The submatrix defined by the intersection of the first i rows and all but the first i columns and last u columns. All entries of this submatrix are zero.
(3) The submatrix defined by the intersection of the first i columns and all but the first i rows. All entries of this submatrix are zero.
(4) The submatrix U defined by the intersection of all the rows and the last u columns.
(5) The submatrix X formed by the intersection of all but the first i columns and the last u columns and all but the first i rows.

Figure 1 illustrates the submatrices of A. At the beginning of the first phase X = A. In each step, a row of A is chosen.

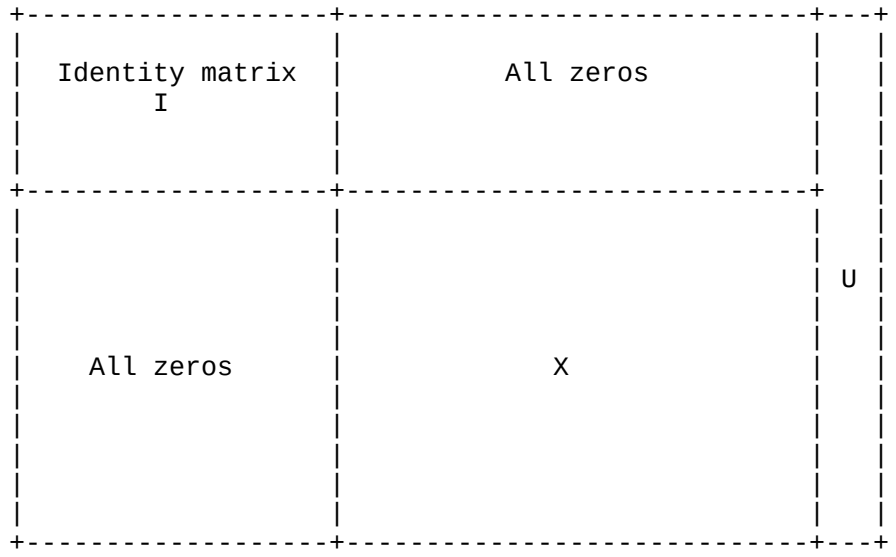

Figure 1: Submatrices of A in the first phase

The following graph defined by the structure of X is used in determining which row of A is chosen. The columns that intersect X are the nodes in the graph, and the rows that have exactly 2 ones in X are the edges of the graph that connect the two columns (nodes) in the positions of the two ones. A component in this graph is a maximal set of nodes (columns) and edges (rows) such that there is a path between each pair of nodes/edges in the graph. The size of a component is the number of nodes (columns) in the component.

There are at most L steps in the first phase. The phase ends successfully when i + u = L, i.e., when X and the all zeroes submatrix above X have disappeared and A consists of I, the all zeroes submatrix below I, and U. The phase ends unsuccessfully in decoding failure if at some step before X disappears there is no non-zero row in X to choose in that step. In each step, a row of A is chosen as follows:

- If all entries of X are zero then no row is chosen and decoding fails.
- Let r be the minimum integer such that at least one row of A has exactly r ones in X.
- If r != 2 then choose a row with exactly r ones in X with minimum original degree among all such rows.
- If r = 2 then choose any row with exactly 2 ones in X that is part of a maximum size component in the graph defined by X.

After the row is chosen in this step the first row of A that intersects X is exchanged with the chosen row so that the chosen row is the first row that intersects X. The columns of A among those that intersect X are reordered so that one of the r ones in the chosen row appears in the first column of X and so that the remaining r-1 ones appear in the last columns of X. Then, the chosen row is exclusive-ored into all the other rows of A below the chosen row that have a one in the first column of X. Finally, i is incremented by 1 and u is incremented by r-1, which completes the step.

A.3 Second Phase

The submatrix U is further partitioned into the first i rows, U,, and the remaining M - i rows, UL. Gaussian elimination is performed in the second phase on UL to either determine that its rank is less than u (decoding failure) or to convert it into a matrix where the first u rows is the identity matrix (success of the second phase). Call this u by u identity matrix UI. The M - L rows of A that intersect UL - UI are discarded. After this phase A has L rows and L columns.

A.4 Third Phase

After the second phase the only portion of A which needs to be zeroed out to finish converting A into the L by L identity matrix is UU. The number of rows i of the submatrix UU is generally much larger than the number of columns u of UU. To zero out UU efficiently, the following precomputation matrix UE is computed based on UI in the third phase and then UE is used in the fourth phase to zero out UU. The u rows of UI are partitioned into ceil(u/8) groups of 8 rows each. Then, for each group of 8 rows all non-zero combinations of the 8 rows are computed, resulting in 2^^8 - 1 = 255 rows (this can be done with 2^^8-8-1 = 247 exclusive-ors of rows per group, since

the combinations of Hamming weight one that appear in UI do not need to be recomputed). Thus, the resulting precomputation matrix UE has ceil(u/8)*255 rows and u columns. Note that UE is not formally a part of matrix A, but will be used in the fourth phase to zero out UU.

A.5 Fourth Phase

For each of the first i rows of A, for each group of 8 columns in the UU submatrix of this row, if the set of 8 column entries in UU are not all zero then the row of the precomputation matrix UE that matches the pattern in the 8 columns is exclusive-ored into the row, thus zeroing out those 8 columns in the row at the cost of exclusive-oring one row of UE into the row.

After this phase A is the L by L identity matrix and a complete decoding schedule has been successfully formed. Then, as explained in Appendix A.2, the corresponding decoding consisting of exclusive-oring known encoding symbols can be executed to recover the pre-coding symbols based on the decoding schedule.

In the case of the non-systematic code, only rows corresponding to recovering a source symbol need be considered in this phase if only the source symbols and not all the pre-coding symbols are to be decoded.

In the case of the systematic code, all rows need be considered to recover all of the intermediate pre-coding symbols. The triples associated with all source symbols are computed according to Section 6.3, and the triples for received source symbols are used in the decoding, and the triples for missing source symbols are used to determine which intermediate pre-coding symbols need to be exclusive-ored to recover the missing source symbols.

Appendix B. Systematic Indices (normative)

The following is a list of the systematic indices for values of K between 16 and 8192:

Appendix C. Random Numbers (normative)

The two tables V0 and V1 described in Section 5.3.1 are given below. Each entry is a 32-bit integer in decimal representation.

C.1 The table V0

251291136, 3952231631, 3370958628, 4070167936, 123631495,
3351110283, 3218676425, 2011642291, 774603218, 2402805061,
1004366930, 1843948209, 428891132, 3746331984, 1591258008,
3067016507, 1433388735, 504005498, 2032657933, 3419319784,
2805686246, 3102436986, 3808671154, 2501582075, 3978944421,
246043949, 4016898363, 649743608, 1974987508, 2651273766,
2357956801, 689605112, 715807172, 2722736134, 191939188,
3535520147, 3277019569, 1470435941, 3763101702, 3232409631,
122701163, 3920852693, 782246947, 372121310, 2995604341,
2045698575, 2332962102, 4005368743, 218596347, 3415381967,
4207612806, 861117671, 3676575285, 2581671944, 3312220480,
681232419, 307306866, 4112503940, 1158111502, 709227802,
2724140433, 4201101115, 4215970289, 4048876515, 3031661061,
1909085522, 510985033, 1361682810, 129243379, 3142379587,
2569842483, 3033268270, 1658118006, 932109358, 1982290045,
2983082771, 3007670818, 3448104768, 683749698, 778296777,
1399125101, 1939403708, 1692176003, 3868299200, 1422476658,
593093658, 1878973865, 2526292949, 1591602827, 3986158854,
3964389521, 2695031039, 1942050155, 424618399, 1347204291,
2669179716, 2434425874, 2540801947, 1384069776, 4123580443,
1523670218, 2708475297, 1046771089, 2229796016, 1255426612,
4213663089, 1521339547, 3041843489, 420130494, 10677091,
515623176, 3457502702, 2115821274, 2720124766, 3242576090, 854310108,
425973987, 325832382, 1796851292, 2462744411, 1976681690,
1408671665, 1228817808, 3917210003, 263976645, 2593736473,
2471651269, 4291353919, 650792940, 1191583883, 3046561335,
2466530435, 2545983082, 969168436, 2019348792, 2268075521,
1169345068, 3250240009, 3963499681, 2560755113, 911182396,
760842409, 3569308693, 2687243553, 381854665, 2613828404,
2761078866, 1456668111, 883760091, 3294951678, 1604598575,
1985308198, 1014570543, 2724959607, 3062518035, 3115293053,
138853680, 4160398285, 3322241130, 2068983570, 2247491078,
3669524410, 1575146607, 828029864, 3732001371, 3422026452,
3370954177, 4006626915, 543812220, 1243116171, 3928372514,
2791443445, 4081325272, 2280435605, 885616073, 616452097,
3188863436, 2780382310, 2340014831, 1208439576, 258356309,
3837963200, 2075009450, 3214181212, 3303882142, 880813252,
1355575717, 207231484, 2420803184, 358923368, 1617557768,
3272161958, 1771154147, 2842106362, 1751209208, 1421030790,
658316681, 194065839, 3241510581, 38625260, 301875395, 4176141739,
297312930, 2137802113, 1502984205, 3669376622, 3728477036,

234652930, 2213589897, 2734638932, 1129721478, 3187422815,
2859178611, 3284308411, 3819792700, 3557526733, 451874476,
1740576081, 3592838701, 1709429513, 3702918379, 3533351328,
1641660745, 179350258, 2380520112, 3936163904, 3685256204,
3156252216, 1854258901, 2861641019, 3176611298, 834787554,
331353807, 517858103, 3010168884, 4012642001, 2217188075,
3756943137, 3077882590, 2054995199, 3081443129, 3895398812,
1141097543, 2376261053, 2626898255, 2554703076, 401233789,
1460049922, 678083952, 1064990737, 940909784, 1673396780,
528881783, 1712547446, 3629685652, 1358307511

C.2 The table V1

807385413, 2043073223, 3336749796, 1302105833, 2278607931, 541015020,
1684564270, 372709334, 3508252125, 1768346005, 1270451292,
2603029534, 2049387273, 3891424859, 2152948345, 4114760273,
915180310, 3754787998, 700503826, 2131559305, 1308908630,
224437350, 4065424007, 3638665944, 1679385496, 3431345226,
1779595665, 3068494238, 1424062773, 1033448464, 4050396853,
3302235057, 420600373, 2868446243, 311689386, 259047959,
4057180909, 1575367248, 4151214153, 110249784, 3006865921,
4293710613, 3501256572, 998007483, 499288295, 1205710710,
2997199489, 640417429, 3044194711, 486690751, 2686640734,
2394526209, 2521660077, 49993987, 3843885867, 4201106668,
415906198, 19296841, 2402488407, 2137119134, 1744097284,
579965637, 2037662632, 852173610, 2681403713, 1047144830,
2982173936, 910285038, 4187576520, 2589870048, 989448887,
3292758024, 506322719, 176010738, 1865471968, 2619324712,
564829442, 1996870325, 339697593, 4071072948, 3618966336,
2111320126, 1093955153, 957978696, 892010560, 1854601078,
1873407527, 2498544695, 2694156259, 1927339682, 1650555729,
183933047, 3061444337, 2067387204, 228962564, 3904109414,
1595995433, 1780701372, 2463145963, 307281463, 3237929991,
3852995239, 2398693510, 3754138664, 522074127, 146352474,
4104915256, 3029415884, 3545667983, 332038910, 976628269,
3123492423, 3041418372, 2258059298, 2139377204, 3243642973,
3226247917, 3674004636, 2698992189, 3453843574, 1963216666,
3509855005, 2358481858, 747331248, 1957348676, 1097574450,
2435697214, 3870972145, 1888833893, 2914085525, 4161315584,
1273113343, 3269644828, 3681293816, 412536684, 1156034077,
3823026442, 1066971017, 3598330293, 1979273937, 2079029895,
1195045909, 1071986421, 2712821515, 3377754595, 2184151095,
750918864, 2585729879, 4249895712, 1832579367, 1192240192,
946734366, 31230688, 3174399083, 3549375728, 1642430184,
1904857554, 861877404, 3277825584, 4267074718, 3122860549,
666423581, 644189126, 226475395, 307789415, 1196105631,
3191691839, 782852669, 1608507813, 1847685900, 4069766876,
3931548641, 2526471011, 766865139, 2115084288, 4259411376,

3323683436, 568512177, 3736601419, 1800276898, 4012458395,
1823982, 27980198, 2023839966, 869505096, 431161506, 1024804023,
1853869307, 3393537983, 1500703614, 3019471560, 1351086955,
3096933631, 3034634988, 2544598006, 1230942551, 3362230798,
159984793, 491590373, 3993872886, 3681855622, 903593547,
3535062472, 1799803217, 772984149, 895863112, 1899036275,
4187322100, 101856048, 234650315, 3183125617, 3190039692,
525584357, 1286834489, 455810374, 1869181575, 922673938,
3877430102, 3422391938, 1414347295, 1971054608, 3061798054,
830555096, 2822905141, 167033190, 1079139428, 4210126723,
3593797804, 429192890, 372093950, 1779187770, 3312189287,
204349348, 452421568, 2800540462, 3733109044, 1235082423,
1765319556, 3174729780, 3762994475, 3171962488, 442160826,
198349622, 45942637, 1324086311, 2901868599, 678860040,
3812229107, 19936821, 1119590141, 3640121682, 3545931032,
2102949142, 2828208598, 3603378023, 4135048896

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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