Reliable Multicast Transport   M. Luby
Internet-Draft   Qualcomm Incorporated
Intended status: Standards Track   A. Shokrollahi
Expires: November 6, 2011   EPFL
                                                                M. Watson
                                                             Netflix Inc.
                                                           T. Stockhammer
                                                           Nomor Research
                                                                L. Minder
                                                    Qualcomm Incorporated
                                                              May 5, 2011

RaptorQ Forward Error Correction Scheme for Object Delivery
draft-ietf-rmt-bb-fec-raptorq-06

Abstract

This document describes a Fully-Specified FEC scheme, corresponding to FEC Encoding ID 6 (to be confirmed (tbc)), for the RaptorQ forward error correction code and its application to reliable delivery of data objects.

RaptorQ codes are a new family of codes that provide superior flexibility, support for larger source block sizes and better coding efficiency than Raptor codes in RFC5053. RaptorQ is also 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 for most cases equal to the number of source symbols and in rare cases with slightly more than the number of source symbols.

The RaptorQ code described here is a systematic code, meaning that all the source symbols are among the encoding symbols that can be generated.

and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 6, 2011.

Copyright Notice

Table of Contents

1. Introduction

This document specifies an FEC Scheme for the RaptorQ forward error correction code for object delivery applications. The concept of an FEC Scheme is defined in RFC5052 [RFC5052] and this document follows the format prescribed there and uses the terminology of that document. The RaptorQ code described herein is a next generation of the Raptor code described in RFC5053 [RFC5053]. The RaptorQ code provides superior reliability, better coding efficiency, and support for larger source block sizes than the Raptor code of RFC5053 [RFC5053]. These improvements simplify the usage of the RaptorQ code in an object delivery Content Delivery Protocol compared to RFC5053 [RFC5053].

The RaptorQ FEC Scheme is a Fully-Specified FEC Scheme corresponding to FEC Encoding ID 6 (tbc).

> Editor's Note: The finalized FEC encoding ID is still to be defined, but '6 (tbc)' is used as temporary value in this Internet Draft expecting sequential use of FEC encoding IDs in the IANA registration process.

RaptorQ 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 can be sent unmodified from sender to receiver, as well as a number of repair symbols. For more backgound on the use of Forward Error Correction codes in reliable multicast, see [RFC3453].

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. Formats and Codes

3.1. Introduction

The octet order of all fields is network byte order, i.e., big-endian.

3.2. FEC Payload IDs

The FEC Payload ID MUST be a 4-octet field defined as follows:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     SBN       |               Encoding Symbol ID              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: FEC Payload ID format

- o Source Block Number (SBN), (8 bits, unsigned integer): A non-negative integer identifier for the source block that the encoding symbols within the packet relate to.

- o Encoding Symbol ID (ESI), (24 bits, unsigned integer): A non-negative integer identifier for the encoding symbols within the packet.

The interpretation of the Source Block Number and Encoding Symbol Identifier is defined in Section 4.

3.3. FEC Object Transmission Information

3.3.1. Mandatory

The value of the FEC Encoding ID MUST be 6, as assigned by IANA (see Section 7).

3.3.2. Common

The Common FEC Object Transmission Information elements used by this FEC Scheme are:

- o Transfer Length (F), (40 bits, unsigned integer): A non-negative integer that is at most 946270874880. This is the transfer length of the object in units of octets.

- o Symbol Size (T), (16 bits, unsigned integer): A positive integer that is less than 2^^16. This is the size of a symbol in units of octets.

The encoded Common FEC Object Transmission Information format is shown in Figure 2.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Transfer Length (F)                      |
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               |    Reserved   |           Symbol Size (T)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Encoded Common FEC OTI for RaptorQ FEC Scheme

NOTE 1: The limit of 946270874880 on the transfer length is a consequence of the limitation on the symbol size to 2^^16-1, the limitation on the number of symbols in a source block to 56403 and the limitation on the number of source blocks to 2^^8.

3.3.3. Scheme-Specific

The following parameters are carried in the Scheme-Specific FEC Object Transmission Information element for this FEC Scheme:

o The number of source blocks (Z) (8 bits, unsigned integer)

o The number of sub-blocks (N) (16 bits, unsigned integer)

o A symbol alignment parameter (Al) (8 bits, unsigned integer)

These parameters are all positive integers. The encoded Scheme-specific Object Transmission Information is a 4-octet field consisting of the parameters Z, N and Al as shown in Figure 3.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Z       |              N                |       Al      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Encoded Scheme-specific FEC Object Transmission Information

The encoded FEC Object Transmission Information is a 12-octet field consisting of the concatenation of the encoded Common FEC Object Transmission Information and the encoded Scheme-specific FEC Object Transmission Information.

These three parameters define the source block partitioning as described in Section 4.4.1.2

4. Procedures

4.1. Introduction

For any undefined symbols or functions used in this section, in particular the functions "ceil" and "floor", refer to Section 5.1.

4.2. Content Delivery Protocol Requirements

This section describes the information exchange between the RaptorQ FEC Scheme and any Content Delivery Protocol (CDP) that makes use of the RaptorQ FEC Scheme for object delivery.

The RaptorQ encoder scheme and RaptorQ decoder scheme for object delivery require the following information from the CDP:

o The transfer length of the object, F, in octets

o A symbol alignment parameter, Al

o The symbol size, T, in octets, which MUST be a multiple of Al

o The number of source blocks, Z

o The number of sub-blocks in each source block, N

The RaptorQ encoder scheme for object delivery additionally requires:

- the object to be encoded, F octets

The RaptorQ encoder scheme supplies the CDP with the following information for each packet to be sent:

o Source Block Number (SBN)

o Encoding Symbol ID (ESI)

o Encoding symbol(s)

The CDP MUST communicate this information to the receiver.

4.3. Example Parameter Derivation Algorithm

This section provides recommendations for the derivation of the three transport parameters, T, Z and N. This recommendation is based on the following input parameters:

- F the transfer length of the object, in octets

- WS the maximum size block that is decodable in working memory, in octets

- P' the maximum payload size in octets, which is assumed to be a multiple of Al

- Al the symbol alignment parameter, in octets

- SS a parameter where the desired lower bound on the sub-symbol size is SS*Al

- K'_max the maximum number of source symbols per source block.

  Note: Section 5.1.2 defines K'_max to be 56403.

Based on the above inputs, the transport parameters T, Z and N are calculated as follows:

Let,

- T = P'

- Kt = ceil(F/T)

- N_max = floor(T/(SS*Al))

- for all n=1, ..., N_max

  * KL(n) is the maximum K' value in Table 2 in Section 5.6 such that

    K' <= WS/(Al*(ceil(T/(Al*n))))

- Z = ceil(Kt/KL(N_max))

- N is the minimum n=1, ..., N_max such that ceil(Kt/Z) <= KL(n)

It is RECOMMENDED that each packet contains exactly one symbol. However, receivers SHALL support the reception of packets that contain multiple symbols.

The value Kt is the total number of symbols required to represent the source data of the object.

The algorithm above and that defined in Section 4.4.1.2 ensure that the sub-symbol sizes are a multiple of the symbol alignment

parameter, Al. This is useful because the sum operations used for encoding and decoding are generally performed several octets at a time, for example at least 4 octets at a time on a 32 bit processor. Thus the encoding and decoding can be performed faster if the sub-symbol sizes are a multiple of this number of octets.

The recommended setting for the input parameter Al is 4.

The parameter WS can be used to generate encoded data which can be decoded efficiently with limited working memory at the decoder. Note that the actual maximum decoder memory requirement for a given value of WS depends on the implementation, but it is possible to implement decoding using working memory only slightly larger than WS.

4.4. Object Delivery

4.4.1. Source block construction

4.4.1.1. General

In order to apply the RaptorQ encoder to a source object, the object may be broken into Z >= 1 blocks, known as source blocks. The RaptorQ encoder is applied independently to each source block. Each source block is identified by a unique Source Block Number (SBN), where the first source block has SBN zero, the second has SBN one, etc. Each source block is divided into a number, K, of source symbols of size T octets each. Each source symbol is identified by a unique Encoding Symbol Identifier (ESI), where the first source symbol of a source block has ESI zero, the second has ESI one, etc.

Each source block with K source symbols is divided into N >= 1 sub-blocks, which are small enough to be decoded in the working memory. Each sub-block is divided into K sub-symbols of size T'.

Note that the value of K is not necessarily the same for each source block of an object and the value of T' may not necessarily be the same for each sub-block of a source block. However, the symbol size T is the same for all source blocks of an object and the number of symbols, K is the same for every sub-block of a source block. Exact partitioning of the object into source blocks and sub-blocks is described in Section 4.4.1.2 below.

4.4.1.2. Source block and sub-block partitioning

The construction of source blocks and sub-blocks is determined based on five input parameters, F, Al, T, Z and N and a function Partition[]. The five input parameters are defined as follows:

- F the transfer length of the object, in octets
- Al a symbol alignment parameter, in octets
- T the symbol size, in octets, which MUST be a multiple of Al
- Z the number of source blocks
- N the number of sub-blocks in each source block

These parameters MUST be set so that ceil(ceil(F/T)/Z) <= K'_max. Recommendations for derivation of these parameters are provided in Section 4.3.

The function Partition[I,J] derives parameters for partitioning a block of size I into J approximately equal sized blocks, and more specifically partitions I into JL blocks of length IL and JS blocks of length IS. Specifically, the output of Partition[I, J] is the sequence (IL, IS, JL, JS), where IL = ceil(I/J), IS = floor(I/J), JL = I - IS * J and JS = J - JL.

The source object MUST be partitioned into source blocks and sub-blocks as follows:

Let

- Kt = ceil(F/T),
- (KL, KS, ZL, ZS) = Partition[Kt, Z],
- (TL, TS, NL, NS) = Partition[T/Al, N].

Then, the object MUST be partitioned into Z = ZL + ZS contiguous source blocks, the first ZL source blocks each having KL*T octets, i.e. KL source symbols of T octets each, and the remaining ZS source blocks each having KS*T octets, i.e. KS source symbols of T octets each.

If Kt*T > F then for encoding purposes, the last symbol of the last source block MUST be padded at the end with Kt*T-F zero octets.

Next, each source block with K source symbols MUST be divided into N = NL + NS contiguous sub-blocks, the first NL sub-blocks each consisting of K contiguous sub-symbols of size of TL*Al octets and the remaining NS sub-blocks each consisting of K contiguous sub-symbols of size of TS*Al octets. The symbol alignment parameter Al ensures that sub-symbols are always a multiple of Al octets.

Finally, the m-th symbol of a source block consists of the concatenation of the m-th sub-symbol from each of the N sub-blocks. Note that this implies that when N > 1 then a symbol is NOT a contiguous portion of the object.

4.4.2. Encoding packet construction

Each encoding packet contains the following information:

o Source Block Number (SBN)

o Encoding Symbol ID (ESI)

o encoding symbol(s)

Each source block is encoded independently of the others. Each encoding packet contains encoding symbols generated from the one source block identified by the SBN carried in the encoding packet. Source blocks are numbered consecutively from zero.

Encoding Symbol ID values from 0 to K-1 identify the source symbols of a source block in sequential order, where K is the number of source symbols in the source block. Encoding Symbol IDs K onwards identify repair symbols generated from the source symbols using the RaptorQ encoder.

Each encoding packet either contains only source symbols (source packet) or contains only repair symbols (repair packet). A packet may contain any number of symbols from the same source block. In the case that the last source symbol in a source packet includes padding octets added for FEC encoding purposes then these octet need not be included in the packet. Otherwise, each packet MUST contain only whole symbols.

The Encoding Symbol ID, X, carried in each source packet is the Encoding Symbol ID of the first source symbol carried in that packet. The subsequent source symbols in the packet have Encoding Symbol IDs, X+1 to X+G-1, in sequential order, where G is the number of symbols in the packet.

Similarly, the Encoding Symbol ID, X, placed into a repair packet is the Encoding Symbol ID of the first repair symbol in the repair packet and the subsequent repair symbols in the packet have Encoding Symbol IDs X+1 to X+G-1 in sequential order, where G is the number of symbols in the packet.

Note that it is not necessary for the receiver to know the total number of repair packets.

4.4.3. Example receiver recovery strategies

A receiver can use the received encoding symbols for each source block of an object to recover the source symbols for that source block independently of all other source blocks.

If there is one sub-block per source block, i.e., N = 1, then the portion of the data in the original object in its original order associated with a source block consists of the concatenation of the source symbols of a source block in consecutive ESI order.

If there are multiple sub-blocks per source block, i.e., if N > 1, then the portion of the data in the original object in its original order associated with a source block consists of the concatenation of the sub-blocks associated with the source block, where sub-symbols within each sub-block are in consecutive ESI order. In this case, there are different receiver source block recovery strategies worth considering depending on the available amount of Random Access Memory (RAM) at the receiver, as outlined below.

One strategy is to recover the source symbols of a source block using the decoding procedures applied to the received symbols for the source block to recover the source symbols as described in Section 5, and then to reorder the sub-symbols of the source symbols so that all consecutive sub-symbols of the first sub-block are first, followed by all consecutive sub-symbols of the second sub-block, etc., followed by all consecutive sub-symbols of the Nth sub-block. This strategy is especially applicable if the receiver has enough RAM to decode an entire source block.

Another strategy is to separately recover the sub-blocks of a source block. For example, a receiver may de-multiplex and store sub-symbols associated with each sub-block separately as packets containing encoding symbols arrive, and then use the stored sub-symbols received for a sub-block to recover that sub-block using the decoding procedures described in Section 5. This strategy is especially applicable if the receiver has enough RAM to decode only one subblock at a time.

5. RaptorQ FEC Code Specification

5.1. Background

For the purpose of the RaptorQ FEC code specification in this section, the following definitions, symbols and abbreviations apply. A basic understanding of linear algebra, matrix operations, and finite fields is assumed in this section. In particular, matrix

multiplication and matrix inversion operations over a mixture of the finite fields GF[2] and GF[256] are used. A basic familiarity with sparse linear equations, and efficient implementations of algorithms that take advantage of sparse linear equations, is also quite beneficial to an implementer of this specification.

5.1.1. Definitions

- Source block: a block of K source symbols which are considered together for RaptorQ encoding and decoding purposes.

- Extended Source Block: a block of K' source symbols, where K' >= K constructed from a source block and zero or more padding symbols.

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

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

- Padding symbol: a symbol with all zero bits that is added to the source block to form the extended source block.

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

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

- Intermediate symbols: symbols generated from the source symbols using an inverse encoding process based on pre-coding relationships. The repair symbols are then generated directly from the intermediate symbols. The encoding symbols do not include the intermediate symbols, i.e., intermediate symbols are not sent as part of the encoding of a source block. The intermediate symbols are partitioned into LT symbols and PI symbols for the purposes of the encoding process.

- LT symbols: A process similar to that described in [LTCodes] is used to generate part of the contribution to each generated encoding symbol from the portion of the intermediate symbols designated as LT symbols.

- o PI symbols: A process even simpler than that described in [LTCodes] is used to generate the other part of the contribution to each generated encoding symbol from the portion of the intermediate symbols designated as PI symbols. In the decoding algorithm suggested in Section 5.4, the PI symbols are inactivated at the start, i.e., are placed into the matrix U at the beginning of the first phase of the decoding algorithm. Because the symbols corresponding to the columns of U are sometimes called the "inactivated" symbols, and since the PI symbols are inactivated at the beginning, they are considered "permanently inactivated".

- o HDPC symbols: There is a small subset of the intermediate symbols that are HDPC symbols. Each HDPC symbol has a pre-coding relationship with a large fraction of the other intermediate symbols. HDPC means "High Density Parity Check".

- o LDPC symbols: There is a moderate-sized subset of the intermediate symbols that are LDPC symbols. Each LDPC symbol has a pre-coding relationship with a small fraction of the other intermediate symbols. LDPC means "Low Density Parity Check".

- o Systematic code: a code in which all source symbols are included as part of the encoding symbols of a source block. The RaptorQ code as described herein is a systematic code.

- o Encoding Symbol ID (ESI): information that uniquely identifies each encoding symbol associated with a source block for sending and receiving purposes.

- o Internal Symbol ID (ISI): information that uniquely identifies each symbol associated with an extended source block for encoding and decoding purposes.

- o Arithmetic operations on octets and symbols and matrices: The operations that are used to produce encoding symbols from source symbols and vice-versa. See Section 5.7.

5.1.2. Symbols

i, j, u, v, h, d, a, b, d1, a1, b1, v, m, x, y represent values or variables of one type or another, depending on the context.

X denotes a non-negative integer value that is either an ISI value or an ESI value, depending on the context.

ceil(x) denotes the smallest integer which is greater than or equal to x, where x is a real value.

floor(x) denotes the largest integer which is less than or equal to x, where x is a real value.

min(x,y) denotes the minimum value of the values x and y, and in general the minimum value of all the argument values.

max(x,y) denotes the maximum value of the values x and y, and in general the maximum value of all the argument values.

i % j denotes i modulo j.

i + j denotes the sum of i and j. If i and j are octets, respectively symbols, this designates the arithmetic on octets, respectively symbols, as defined in Section 5.7. If i and j are integers, then it denotes the usual integer addition.

i * j denotes the product of i and j. If i and j are octets, this designates the arithmetic on octets, as defined in Section 5.7. If i is an octet and j is a symbol, this denotes the multiplication of a symbol by an octet, as also defined in Section 5.7. Finally, if i and j are integers, i * j denotes the usual product of integers.

a ^^ b denotes the operation a raised to the power b. If a is an octet and b is a non-negative integer, this is understood to mean a*a*...*a (b terms), with '*' being the octet product as defined in Section 5.7.

u ^ v denotes, for equal-length bit strings u and v, the bitwise exclusive-or of u and v.

Transpose[A] denotes the transposed matrix of matrix A. In this specification, all matrices have entries that are octets.

A^^-1 denotes the inverse matrix of matrix A. In this specification, all the matrices have octets as entries, so it is understood that the operations of the matrix entries are to be done as stated in Section 5.7 and A^^-1 is the matrix inverse of A with respect to octet arithmetic.

K denotes the number of symbols in a single source block.

K' denotes the number of source plus padding symbols in an extended source block. For the majority of this specification, the padding symbols are considered to be additional source symbols.

K'_max denotes the maximum number of source symbols that can be in a single source block. Set to 56403.

L denotes the number of intermediate symbols for a single extended source block.

S denotes the number of LDPC symbols for a single extended source block. These are LT symbols. For each value of K' shown in Table 2 in Section 5.6, the corresponding value of S is a prime number.

H denotes the number of HDPC symbols for a single extended source block. These are PI symbols.

B denotes the number of intermediate symbols that are LT symbols excluding the LDPC symbols.

W denotes the number of intermediate symbols that are LT symbols. For each value of K' in Table 2 shown in Section 5.6, the corresponding value of W is a prime number.

P denotes the number of intermediate symbols that are PI symbols. These contain all HDPC symbols.

P1 denotes the smallest prime number greater than or equal to P.

U denotes the number of non-HDPC intermediate symbols that are PI symbols.

C denotes an array of intermediate symbols, C[0], C[1], C[2],..., C[L-1].

C' denotes an array of the symbols of the extended source block, where C'[0], C'[1], C'[2],..., C'[K-1] are the source symbols of the source block and C'[K], C'[K+1],..., C'[K'-1] are padding symbols.

V0, V1, V2, V3 denote four arrays of 32-bit unsigned integers, V0[0], V0[1],..., V0[255] ; V1[0], V1[1],..., V1[255]; V2[0], V2[1],..., V2[255]; and V3[0], V3[1],..., V3[255] as shown in Section 5.5.

Rand[y, i, m] denotes a pseudo-random number generator

Deg[v] denotes a degree generator

Enc[K', C ,(d, a, b, d1, a1, b1)] denotes an encoding symbol generator

Tuple[K', X] denotes a tuple generator function

T denotes the symbol size in octets.

J(K') denotes the systematic index associated with K'.

G denotes any generator matrix.

I_S denotes the SxS identity matrix.

5.2. Overview

This section defines the systematic RaptorQ FEC code.

Symbols are the fundamental data units of the encoding and decoding process. For each source block all symbols are the same size, referred to as the symbol size T. The atomic operations performed on symbols for both encoding and decoding are the arithmetic operations defined in Section 5.7.

The basic encoder is described in Section 5.3. The encoder first derives a block of intermediate symbols from the source symbols of a source block. This intermediate block has the property that both source and repair symbols can be generated from it using the same process. The encoder produces repair symbols from the intermediate block using an efficient process, where each such repair symbol is the exclusive OR of a small number of intermediate symbols from the block. Source symbols can also be reproduced from the intermediate block using the same process. The encoding symbols are the combination of the source and repair symbols.

An example of a decoder is described in Section 5.4. The process for producing source and repair symbols from the intermediate block is designed so that the intermediate block can be recovered from any sufficiently large set of encoding symbols, independent of the mix of source and repair symbols in the set. Once the intermediate block is recovered, missing source symbols of the source block can be recovered using the encoding process.

Requirements for a RaptorQ compliant decoder are provided in Section 5.8. A number of decoding algorithms are possible to achieve

these requirements. An efficient decoding algorithm to achieve these requirements is provided in Section 5.4.

The construction of the intermediate and repair symbols is based in part on a pseudo-random number generator described in Section 5.3. This generator is based on a fixed set of 1024 random numbers which must be available to both sender and receiver. These numbers are provided in Section 5.5. Encoding and decoding operations for RaptorQ use operations on octets. Section 5.7 describes how to perform these operations.

Finally, the construction of the intermediate symbols from the source symbols is governed by "systematic indices", values of which are provided in Section 5.6 for specific extended source block sizes between 6 and K'_max = 56403 source symbols. Thus, the RaptorQ code supports source blocks with between 1 and 56403 source symbols.

5.3. Systematic RaptorQ encoder

5.3.1. Introduction

For a given source block of K source symbols, for encoding and decoding purposes the source block is augmented with K'-K additional padding symbols, where K' is the smallest value that is at least K in the systematic index Table 2 of Section 5.6. The reason for padding out a source block to a multiple of K' is to enable faster encoding and decoding, and to minimize the amount of table information that needs to be stored in the encoder and decoder.

For purposes of transmitting and receiving data, the value of K is used to determine the number of source symbols in a source block, and thus K needs to be known at the sender and the receiver. In this case the sender and receiver can compute K' from K and the K'-K padding symbols can be automatically added to the source block without any additional communication. The encoding symbol ID (ESI) is used by a sender and receiver to identify the encoding symbols of a source block, where the encoding symbols of a source block consist of the source symbols and the repair symbols associated with the source block. For a source block with K source symbols, the ESIs for the source symbols are 0,1,2,...,K-1 and the ESIs for the repair symbols are K, K+1, K+2,... . Using the ESI for identifying encoding symbols in transport ensures that the ESI values continue consecutively between the source and repair symbols.

For purposes of encoding and decoding data, the value of K' derived from K is used as the number of source symbols of the extended source block upon which encoding and decoding operations are performed, where the K' source symbols consist of the original K source symbols

and an additional K'-K padding symbols. The internal symbol ID (ISI) is used by the encoder and decoder to identify the symbols associated with the extended source block, i.e., for generating encoding symbols and for decoding. For a source block with K original source symbols, the ISIs for the original source symbols are 0,1,2,...,K-1, the ISIs for the K'-K padding symbols are K, K+1, K+2,..., K'-1, and the ISIs for the repair symbols are K', K'+1, K'+2,... . Using the ISI for encoding and decoding allows the padding symbols of the extended source block to be treated the same way as other source symbols of the extended source block, and that a given prefix of repair symbols are generated in a consistent way for a given number K' of source symbols in the extended source block independent of K.

The relationship between the ESIs and the ISIs is simple: the ESIs and the ISIs for the original K source symbols are the same, the K'-K padding symbols have an ISI but do not have a corresponding ESI (since they are symbols that are neither sent nor received), and a repair symbol ISI is simply the repair symbol ESI plus K'-K. The translation between ESIs used to identify encoding symbols sent and received and the corresponding ISIs used for encoding and decoding, and the proper padding of the extended source block with padding symbols used for encoding and decoding, is the responsibility of the padding function in the RaptorQ encoder/decoder.

5.3.2. Encoding overview

The systematic RaptorQ encoder is used to generate any number of repair symbols from a source block that consists of K source symbols placed into an extended source block C'. Figure 4 shows the encoding overview.

The first step of encoding is to construct an extended source block by adding zero or more padding symbols such that the total number of symbols, K', is one of the values listed in Section 5.6. Each padding symbol consists of T octets where the value of each octet is zero. K' MUST be selected as the smallest value of K' from the table of Section 5.6 which is greater than or equal to K.

```
         -----------------------------------------------------------+
         |                                                          |
         |    +-----------+    +--------------+    +-------------+  |
      C' |    |           | C' | Intermediate | C  |             |  |
     ----+--->|  Padding  |--->|    Symbol    |--->|   Encoding  |--+-->
      K  |    |           | K' |  Generation  | L  |             |  |
         |    +-----------+    +--------------+    +-------------+  |
         |          |                             (d,a,b, ^        |
         |          |                            d1,a1,b1)|        |
         |          |                              +------------+  |
         |          |              K'              |   Tuple    |  |
         |          +----------------------------->|            |  |
         |                                         | Generation |  |
         |                                         +------------+  |
         |                                                ^        |
         +-------------------------------------------------+--------+
                                                           |
                                                         ISI X
```

Figure 4: Encoding Overview

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

Let C'[K], ..., C'[K'-1] denote the K'-K padding symbols, which are all set to zero bits. Then, C'[0],..., C'[K'-1] are the symbols of the extended source block upon which encoding and decoding are performed.

In the remainder of this description these padding symbols will be considered as additional source symbols and referred to as such. However, these padding symbols are not part of the encoding symbols, i.e., they are not sent as part of the encoding. At a receiver, the value of K' can be computed based on K, then the receiver can insert K'-K padding symbols at the end of a source block of K' source symbols and recover the remaining K source symbols of the source block from received encoding symbols.

The second step of encoding is to generate a number, L > K', of intermediate symbols from the K' source symbols. In this step, K' source tuples (d[0], a[0], b[0], d1[0], a1[0], b1[0]), ..., (d[K'-1], a[K'-1], b[K'-1], d1[K'-1], a1[K'-1], b1[K'-1]) are generated using the Tuple[] generator as described in Section 5.3.5.4. The K' source tuples and the ISIs associated with the K' source symbols are used to determine L intermediate symbols C[0],..., C[L-1] from the source symbols using an inverse encoding process. This process can be realized by a RaptorQ decoding process.

Certain "pre-coding relationships" must hold within the L intermediate symbols. Section 5.3.3.3 describes these relationships. Section 5.3.3.4 describes how the intermediate symbols are generated from the source symbols.

Once the intermediate symbols have been generated, repair symbols can be produced. For a repair symbol with ISI X>K', the tuple of non-negative integers, (d, a, b, d1, a1, b1) can be generated, using the Tuple[] generator as described in Section 5.3.5.4. Then, the (d, a, b, d1, a1, b1)-tuple and the ISI X is used to generate the corresponding repair symbol from the intermediate symbols using the Enc[] generator described in Section 5.3.5.3. The corresponding ESI for this repair symbol is then X-(K'-K). Note that source symbols of the extended source block can also be generated using the same process, i.e., for any X < K', the symbol generated using this process has the same value as C'[X].

5.3.3. First encoding step: Intermediate Symbol Generation

5.3.3.1. General

This encoding step is a pre-coding step to generate the L intermediate symbols C[0], ..., C[L-1] from the source symbols C'[0], ..., C'[K'-1], , where L > K' is defined in Section 5.3.3.3. The intermediate symbols are uniquely defined by two sets of constraints:

1. The intermediate symbols are related to the source symbols by a set of source symbol tuples and by the ISIs of the source symbols. The generation of the source symbol tuples is defined in Section 5.3.3.2 using the the Tuple[] generator as described in Section 5.3.5.4.

2. A number of pre-coding relationships hold within the intermediate symbols themselves. These are defined in Section 5.3.3.3

The generation of the L intermediate symbols is then defined in Section 5.3.3.4.

5.3.3.2. Source symbol tuples

Each of the K' source symbols is associated with a source symbol tuple (d[X], a[X], b[X], d1[X], a1[X], b1[X]) for 0 <= X < K'. The source symbol tuples are determined using the Tuple generator defined in Section 5.3.5.4 as:

    For each X, 0 <= X < K'

(d[X], a[X], b[X], d1[X], a1[X], b1[X]) = Tuple[K, X]

5.3.3.3. Pre-coding relationships

The pre-coding relationships amongst the L intermediate symbols are defined by requiring that a set of S+H linear combinations of the intermediate symbols evaluate to zero. There are S LDPC and H HDPC symbols, and thus L = K'+S+H. Another partition of the L intermediate symbols is into two sets, one set of W LT symbols and another set of P PI symbols, and thus it is also the case that L = W+P. The P PI symbols are treated differently than the W LT symbols in the encoding process. The P PI symbols consist of the H HDPC symbols together with a set of U = P-H of the other K' intermediate symbols. The W LT symbols consist of the S LDPC symbols together with W-S of the other K' intermediate symbols. The values of these parameters are determined from K' as described below where H(K'), S(K'), and W(K') are derived from Table 2 in Section 5.6.

Let

o S = S(K')

o H = H(K')

o W = W(K')

o L = K' + S + H

o P = L - W

o P1 denote the smallest prime number greater than or equal to P

o U = P - H

o B = W - S

o C[0], ..., C[B-1] denote the intermediate symbols that are LT symbols but not LDPC symbols.

o C[B], ..., C[B+S-1] denote the S LDPC symbols that are also LT symbols.

o C[W], ..., C[W+U-1] denote the intermediate symbols that are PI symbols but not HDPC symbols.

o C[L-H], ..., C[L-1] denote the H HDPC symbols that are also PI symbols.

The first set of pre-coding relations, called LDPC relations, is described below and requires that at the end of this process the set of symbols D[0] , ..., D[S-1] are all zero:

- Initialize the symbols D[0] = C[B], ..., D[S-1] = C[B+S-1].

- For i = 0, ..., B-1 do

  * a = 1 + floor(i/S)

  * b = i % S

  * D[b] = D[b] + C[i]

  * b = (b + a) % S

  * D[b] = D[b] + C[i]

  * b = (b + a) % S

  * D[b] = D[b] + C[i]

- For i = 0, ..., S-1 do

  * a = i % P

  * b = (i+1) % P

  * D[i] = D[i] + C[W+a] + C[W+b]

Recall that the addition of symbols is to be carried out as specified in Section 5.7.

Note that the LDPC relations as defined in the algorithm above are linear, so there exists an S x B matrix G_LDPC,1 and an S x P matrix G_LDPC,2 such that

```
G_LDPC,1 * Transpose[(C[0],...., C[B-1])] + G_LDPC,2 *
Transpose(C[W], ..., C[W+P-1]) + Transpose[(C[B], ..., C[B+S-1])]
= 0
```

(The matrix G_LDPC,1 is defined by the first loop in the above algorithm, and G_LDPC,2 can be deduced from the second loop.)

The second set of relations among the intermediate symbols C[0], ..., C[L-1] are the HDPC relations and they are defined as follows:

Let

- alpha denote the octet represented by integer 2 as defined in Section 5.7.

- MT denote an H x (K' + S) matrix of octets, where for j=0,...,K'+S-2 the entry MT[i,j] is the octet represented by the integer 1 if i= Rand[j+1,6,H] or i = (Rand[j+1,6,H] + Rand[j+ 1,7,H-1] + 1) % H and MT[i,j] is the zero element for all other values of i, and for j=K'+S-1, MT[i,j] = alpha^^i for i=0,...,H-1.

- GAMMA denote a (K'+S) x (K'+S) matrix of octets, where

  GAMMA[i,j] =

  alpha ^^ (i-j) for i >= j,

  0 otherwise.

Then the relationship between the first K'+S intermediate symbols C[0], ..., C[K'+S-1] and the H HDPC symbols C[K'+S], ..., C[K'+S+H-1] is given by:

```
Transpose[C[K'+S], ..., C[K'+S+H-1]] + MT * GAMMA *
Transpose[C[0], ..., C[K'+S-1]] = 0,
```

where '*' represents standard matrix multiplication utilizing the octet multiplication to define the multiplication between a matrix of octets and a matrix of symbols (in particular the column vector of symbols) and '+' denotes addition over octet vectors.

5.3.3.4. Intermediate symbols

5.3.3.4.1. Definition

Given the K' source symbols C'[0], C'[1],..., C'[K'-1] the L intermediate symbols C[0], C[1],..., C[L-1] are the uniquely defined symbol values that satisfy the following conditions:

1. The K' source symbols C'[0], C'[1],..., C'[K'-1] satisfy the K' constraints

   ```
   C'[X] = Enc[K', (C[0],..., C[L-1]), (d[X], a[X], b[X], d1[X],
   a1[X], b1[X])], for all X, 0 <= X < K',
   ```

   where (d[X], a[X], b[X], d1[X], a1[X], b1[X])) = Tuple[K',X], Tuple[] is defined in Section 5.3.5.4 and Enc[] is described in Section 5.3.5.3.

2. The L intermediate symbols C[0], C[1],..., C[L-1] satisfy the pre-coding relationships defined in Section 5.3.3.3

5.3.3.4.2. Example method for calculation of intermediate symbols

This section describes a possible method for calculation of the L intermediate symbols C[0], C[1],..., C[L-1] satisfying the constraints in Section 5.3.3.4.1

The L intermediate symbols can be calculated as follows:

Let

- o C denote the column vector of the L intermediate symbols, C[0], C[1],..., C[L-1].
- o D denote the column vector consisting of S+H zero symbols followed by the K' source symbols C'[0], C'[1], ..., C'[K'-1].

Then the above constraints define an L x L matrix A of octets such that:

```
A*C = D
```

The matrix A can be constructed as follows:

Let:

- o G_LDPC,1 and G_LDPC,2 be S x B and S x P matrices as defined in Section 5.3.3.3.
- o G_HDPC be the H x (K'+S) matrix such that

```
G_HDPC * Transpose(C[0], ..., C[K'+S-1]) = Transpose(C[K'+S],
..., C[L-1]),

i.e.  G_HDPC = MT*GAMMA
```

- o I_S be the S x S identity matrix
- o I_H be the H x H identity matrix
- o G_ENC be the K' x L matrix such that

```
G_ENC * Transpose[(C[0], ..., C[L-1])] =
Transpose[(C'[0],C'[1],...,C'[K'-1])],
```

i.e. G_ENC[i,j] = 1 if and only if C[j] is included in the symbols which are summed to produce Enc[K', (C[0], ..., C[L-1]), (d[i], a[i], b[i], d1[i], a1[i], b1[i])] and G_ENC[i,j] = 0 otherwise.

Then:

- o The first S rows of A are equal to G_LDPC,1 | I_S | G_LDPC,2.
- o The next H rows of A are equal to G_HDPC | I_H.
- o The remaining K' rows of A are equal to G_ENC.

The matrix A is depicted in Figure (Figure 5) below:

```
              B               S         U         H
   +-----------------------+-------+------------------+
   |                       |       |                  |
 S |        G_LDPC,1       |  I_S  |      G_LDPC,2    |
   |                       |       |                  |
   +-----------------------+-------+----------+-------+
   |                                          |       |
 H |                G_HDPC                    |  I_H  |
   |                                          |       |
   +------------------------------------------+-------+
   |                                                  |
   |                                                  |
K' |                      G_ENC                       |
   |                                                  |
   |                                                  |
   +--------------------------------------------------+
```

Figure 5: The matrix A

The intermediate symbols can then be calculated as:

C = (A^^-1)*D

The source tuples are generated such that for any K' matrix A has full rank and is therefore invertible. This calculation can be realized by applying a RaptorQ decoding process to the K' source symbols C'[0], C'[1],..., C'[K'-1] to produce the L intermediate symbols C[0], C[1],..., C[L-1].

To efficiently generate the intermediate symbols from the source symbols, it is recommended that an efficient decoder implementation such as that described in Section 5.4 be used.

5.3.4. Second encoding step: Encoding

In the second encoding step, the repair symbol with ISI X (X >= K') is generated by applying the generator Enc[K', (C[0], C[1],..., C[L-1]), (d, a, b, d1, a1, b1)] defined in Section 5.3.5.3 to the L intermediate symbols C[0], C[1],..., C[L-1] using the tuple (d, a, b, d1, a1, b1)=Tuple[K',X].

5.3.5. Generators

5.3.5.1. Random Number Generator

The random number generator Rand[y, i, m] is defined as follows, where y is a non-negative integer, i is a non-negative integer less than 256, and m is a positive integer and the value produced is an integer between 0 and m-1. Let V0, V1, V2 and V3 be the arrays provided in Section 5.5.

Let

o x0 = (y + i) mod 2^^8

o x1 = (floor(y / 2^^8) + i) mod 2^^8

o x2 = (floor(y / 2^^16) + i) mod 2^^8

o x3 = (floor(y / 2^^24) + i) mod 2^^8

Then

```
Rand[y, i, m] = (V0[x0] ^ V1[x1] ^ V2[x2] ^ V3[x3]) % m
```

5.3.5.2. Degree Generator

The degree generator Deg[v] is defined as follows, where v is a non-negative integer that is less than 2^^20 = 1048576. Given v, find index d in Table 1 such that f[d-1] <= v < f[d], and set Deg[v] = min(d, W-2).). Recall that W is derived from K' as described in Section 5.3.3.3.

Index d	f[d]	Index d	f[d]
0	0	1	5243
2	529531	3	704294
4	791675	5	844104
6	879057	7	904023
8	922747	9	937311
10	948962	11	958494
12	966438	13	973160
14	978921	15	983914
16	988283	17	992138
18	995565	19	998631
20	1001391	21	1003887
22	1006157	23	1008229
24	1010129	25	1011876
26	1013490	27	1014983
28	1016370	29	1017662
30	1048576		

Table 1: Defines the degree distribution for encoding symbols

5.3.5.3. Encoding Symbol Generator

The encoding symbol generator Enc[K', (C[0], C[1],..., C[L-1]), (d, a, b, d1, a1, b1)] takes the following inputs:

- o K' is the number of source symbols for the extended source block. Let L, W, B, S, P and P1 be derived from K' as described in Section 5.3.3.3.

- o (C[0], C[1],..., C[L-1]) is the array of L intermediate symbols (sub-symbols) generated as described in Section 5.3.3.4

- o (d, a, b, d1, a1, b1) is a source tuple determined from ISI X using the Tuple generator defined in Section 5.3.5.4, whereby

  * d is a positive integer denoting an encoding symbol LT degree

  * a is a positive integer between 1 and W-1 inclusive

  * b is a non-negative integer between 0 and W-1 inclusive

  * d1 is a positive integer that has value either 2 or 3 inclusive denoting an encoding symbol PI degree

  * a1 is a positive integer between 1 and P1-1 inclusive

  * b1 is a non-negative integer between 0 and P1-1 inclusive

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

- o result = C[b]

- o For j = 1, ..., d-1 do

  * b = (b + a) % W

  * result = result + C[b]

- o While (b1 >= P) do b1 = (b1+a1) % P1

- o result = result + C[W+b1]

- o For j = 1, ..., d1-1 do

  * b1 = (b1 + a1) % P1

  * While (b1 >= P) do b1 = (b1+a1) % P1

  * result = result + C[W+b1]

- o Return result

5.3.5.4. Tuple generator

The tuple generator Tuple[K',X] takes the following inputs:

- K' - The number of source symbols in the extended source block

- X - An ISI

Let

- L be determined from K' as described in Section 5.3.3.3

- J=J(K') be the systematic index associated with K', as defined in Table 2 inSection 5.6

The output of tuple generator is a tuple, (d, a, b, d1, a1, b1), determined as follows:

- A = 53591 + J*997

- if (A % 2 == 0) { A = A + 1 }

- B = 10267*(J+1)

- y = (B + X*A) % 2^^32

- v = Rand[y, 0, 2^^20]

- d = Deg[v]

- a = 1 + Rand[y, 1, W-1]

- b = Rand[y, 2, W]

- If (d<4) { d1 = 2 + Rand[X, 3, 2] } else { d1 = 2 }

- a1 = 1 + Rand[X, 4, P1-1]

- b1 = Rand[X, 5, P1]

5.4. Example FEC decoder

5.4.1. General

This section describes an efficient decoding algorithm for the RaptorQ code introduced in this specification. Note that each received encoding symbol is a known linear combination of the intermediate symbols. So each received encoding symbol provides a

linear equation among the intermediate symbols, which, together with the known linear pre-coding relationships amongst the intermediate symbols gives a system of linear equations. Thus, any algorithm for solving systems of linear equations can successfully decode the intermediate symbols and hence the source symbols. However, the algorithm chosen has a major effect on the computational efficiency of the decoding.

5.4.2. Decoding an extended source block

5.4.2.1. 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 and the number K' of source symbols in the extended source block.

From the algorithms described in Section 5.3, the RaptorQ decoder can calculate the total number L = K'+S+H of intermediate symbols and determine how they were generated from the extended source block to be decoded. In this description it is assumed that the received encoding symbols for the extended 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 symbols whose sum is equal to the encoding symbol are passed to the decoder. In the case of source symbols, including padding symbols, the source symbol tuples described in Section 5.3.3.2 indicate the number and set of intermediate symbols which sum to give each source symbol.

Let N >= K' be the number of received encoding symbols to be used for decoding, including padding symbols for an extended source block and let M = S+H+N. Then with the notation of Section 5.3.3.4.2 we have A*C=D.

Decoding an extended 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 is L. Once C has been decoded, missing source symbols can be obtained by using the source symbol tuples to determine the number and set of intermediate symbols which must be summed 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 - 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.

o Each time a multiple, beta, of row i of A is added to row i' in the decoding schedule then in the decoding process the symbol beta*D[d[i]] is added to symbol D[d[i']] .

o Each time a row i of A is multiplied by an octet beta, then in the decoding process the symbol D[d[i]] is also multiplied by beta.

o 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'].

o 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 operations on symbols in the decoding of the extended 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 - 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.

5.4.2.2. First Phase

In the first phase of the Gaussian elimination the matrix A is conceptually partitioned into submatrices and additionally, a matrix X is created. This matrix has as many rows and columns as A, and it will be a lower triangular matrix throughout the first phase. At the beginning of this phase, the matrix A is copied into the matrix X. The submatrix sizes are parameterized by non-negative integers i and

u which are initialized to 0 and P, the number of PI symbols, respectively. 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 V formed by the intersection of all but the first i columns and the last u columns and all but the first i rows.

Figure 6 illustrates the submatrices of A. At the beginning of the first phase V consists of the first L-P columns of A and U consists of the last P columns corresponding to the PI symbols. In each step, a row of A is chosen.

```
+-----------+-----------------+---------+
|           |                 |         |
|     I     |    All Zeros    |
|           |                 |         |
+-----------+-----------------+    U    |
|           |                 |         |
|           |                 |         |
| All Zeros |       V         |         |
|           |                 |         |
|           |                 |         |
+-----------+-----------------+---------+
```

Figure 6: Submatrices of A in the first phase

The following graph defined by the structure of V is used in determining which row of A is chosen. The columns that intersect V are the nodes in the graph, and the rows that have exactly 2 non-zero entries in V and are not HDPC rows 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 V and the all zeroes submatrix above V 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 V disappears there is no non-zero row in V to choose in that step. In each step, a row of A is chosen as follows:

- o If all entries of V are zero then no row is chosen and decoding fails.

- o Let r be the minimum integer such that at least one row of A has exactly r non-zeroes in V.

  * If r != 2 then choose a row with exactly r non-zeroes in V with minimum original degree among all such rows, except that HDPC rows should not be chosen until all non-HDPC rows have been processed.

  * If r = 2 and there is a row with exactly 2 ones in V then choose any row with exactly 2 ones in V that is part of a maximum size component in the graph described above that is defined by V.

  * If r = 2 and there is no row with exactly 2 ones in V then choose any row with exactly 2 non-zeroes in V.

After the row is chosen in this step the first row of A that intersects V is exchanged with the chosen row so that the chosen row is the first row that intersects V. The columns of A among those that intersect V are reordered so that one of the r non-zeroes in the chosen row appears in the first column of V and so that the remaining r-1 non-zeroes appear in the last columns of V. The same row and column operations are also performed on the matrix X. Then, an appropriate multiple of the chosen row is added to all the other rows of A below the chosen row that have a non-zero entry in the first column of V. Specifically, if a row below the chosen row has entry beta in the first column of V, and the chosen row has entry alpha in the first column of V, then beta/alpha multiplied by the chosen row is added to this row to leave a zero value in the first column of V. Finally, i is incremented by 1 and u is incremented by r-1, which completes the step.

Note that efficiency can be improved if the row operations identified above are not actually performed until the affected row is itself chosen during the decoding process. This avoids processing of row

operations for rows which are not eventually used in the decoding process and in particular avoid those rows for which beta!=1 until they are actually required. Furthermore, the row operations required for the HDPC rows may be performed for all such rows in one process, by using the algorithm described in Section 5.3.3.3.

5.4.2.3. Second Phase

At this point, all the entries of X outside the first i rows and i columns are discarded, so that X has lower triangular form. The last i rows and columns of X are discarded, so that X now has i rows i columns. The submatrix U is further partitioned into the first i rows, U_upper, and the remaining M - i rows, U_lower. Gaussian elimination is performed in the second phase on U_lower 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 I_u. The M - L rows of A that intersect U_lower - I_u are discarded. After this phase A has L rows and L columns.

5.4.2.4. 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 U_upper. The number of rows i of the submatrix U_upper is generally much larger than the number of columns u of U_upper. Moreover, at this time, the matrix U_upper is typically dense, i.e., the number of nonzero entries of this matrix is large. To reduce this matrix to a sparse form, the sequence of operations performed to obtain the matrix U_lower needs to be inverted. To this end, the matrix X is multiplied with the submatrix of A consisting of the first i rows of A. After this operation the submatrix of A consisting of the intersection of the first i rows and columns equals to X, whereas the matrix U_upper is transformed to a sparse form.

5.4.2.5. Fourth Phase

For each of the first i rows of U_upper do the following: if the row has a nonzero entry at position j, and if the value of that nonzero entry is b, then add to this row b times row j of I_u. After this step, the submatrix of A consisting of the intersection of the first i rows and columns is equal to X, the submatrix U_upper consists of zeros, the submatrix consisting of the intersection of the last u rows and the first i columns consists of zeros, and the submatrix consisting of the last u rows and columns is is the matrix I_u.

5.4.2.6. Fifth Phase

For j from 1 to i perform the following operations:

1. if A[j,j] is not one, then divide row j of A by A[j,j].

2. For l from 1 to j-1, if A[j,l] is nonzero, then add A[j,l] multiplied with row l of A to row j of A.

After this phase A is the L by L identity matrix and a complete decoding schedule has been successfully formed. Then, the corresponding decoding consisting of summing known encoding symbols can be executed to recover the intermediate symbols based on the decoding schedule. The tuples associated with all source symbols are computed according to Section 5.3.3.2. The tuples for received source symbols are used in the decoding. The tuples for missing source symbols are used to determine which intermediate symbols need to be summed to recover the missing source symbols.

5.5. Random Numbers

The four arrays V0, V1, V2 and V3 used in Section 5.3.5.1 are provided below. There are 256 entries in each of the four arrays. The indexing into each array starts at 0, and the entries are 32-bit unsigned integers.

5.5.1. 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

5.5.2. 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
```

5.5.3. The table V2

```
1629829892, 282540176, 2794583710, 496504798, 2990494426, 3070701851,
2575963183, 4094823972, 2775723650, 4079480416, 176028725,
2246241423, 3732217647, 2196843075, 1306949278, 4170992780,
4039345809, 3209664269, 3387499533, 293063229, 3660290503,
2648440860, 2531406539, 3537879412, 773374739, 4184691853,
1804207821, 3347126643, 3479377103, 3970515774, 1891731298,
2368003842, 3537588307, 2969158410, 4230745262, 831906319,
2935838131, 264029468, 120852739, 3200326460, 355445271, 2296305141,
1566296040, 1760127056, 20073893, 3427103620, 2866979760, 2359075957,
2025314291, 1725696734, 3346087406, 2690756527, 99815156, 4248519977,
2253762642, 3274144518, 598024568, 3299672435, 556579346, 4121041856,
2896948975, 3620123492, 918453629, 3249461198, 2231414958,
3803272287, 3657597946, 2588911389, 242262274, 1725007475,
2026427718, 46776484, 2873281403, 2919275846, 3177933051, 1918859160,
2517854537, 1857818511, 3234262050, 479353687, 200201308, 2801945841,
1621715769, 483977159, 423502325, 3689396064, 1850168397, 3359959416,
```

3459831930, 841488699, 3570506095, 930267420, 1564520841, 2505122797,
593824107, 1116572080, 819179184, 3139123629, 1414339336, 1076360795,
512403845, 177759256, 1701060666, 2239736419, 515179302, 2935012727,
3821357612, 1376520851, 2700745271, 966853647, 1041862223, 715860553,
171592961, 1607044257, 1227236688, 3647136358, 1417559141,
4087067551, 2241705880, 4194136288, 1439041934, 20464430, 119668151,
2021257232, 2551262694, 1381539058, 4082839035, 498179069, 311508499,
3580908637, 2889149671, 142719814, 1232184754, 3356662582,
2973775623, 1469897084, 1728205304, 1415793613, 50111003, 3133413359,
4074115275, 2710540611, 2700083070, 2457757663, 2612845330,
3775943755, 2469309260, 2560142753, 3020996369, 1691667711,
4219602776, 1687672168, 1017921622, 2307642321, 368711460,
3282925988, 213208029, 4150757489, 3443211944, 2846101972,
4106826684, 4272438675, 2199416468, 3710621281, 497564971, 285138276,
765042313, 916220877, 3402623607, 2768784621, 1722849097, 3386397442,
487920061, 3569027007, 3424544196, 217781973, 2356938519, 3252429414,
145109750, 2692588106, 2454747135, 1299493354, 4120241887,
2088917094, 932304329, 1442609203, 952586974, 3509186750, 753369054,
854421006, 1954046388, 2708927882, 4047539230, 3048925996,
1667505809, 805166441, 1182069088, 4265546268, 4215029527,
3374748959, 373532666, 2454243090, 2371530493, 3651087521,
2619878153, 1651809518, 1553646893, 1227452842, 703887512,
3696674163, 2552507603, 2635912901, 895130484, 3287782244,
3098973502, 990078774, 3780326506, 2290845203, 41729428, 1949580860,
2283959805, 1036946170, 1694887523, 4880696, 466000198, 2765355283,
3318686998, 1266458025, 3919578154, 3545413527, 2627009988,
3744680394, 1696890173, 3250684705, 4142417708, 915739411,
3308488877, 1289361460, 2942552331, 1169105979, 3342228712,
698560958, 1356041230, 2401944293, 107705232, 3701895363, 903928723,
3646581385, 844950914, 1944371367, 3863894844, 2946773319,
1972431613, 1706989237, 29917467, 3497665928

5.5.4. The table V3

1191369816, 744902811, 2539772235, 3213192037, 3286061266,
1200571165, 2463281260, 754888894, 714651270, 1968220972, 3628497775,
1277626456, 1493398934, 364289757, 2055487592, 3913468088,
2930259465, 902504567, 3967050355, 2056499403, 692132390, 186386657,
832834706, 859795816, 1283120926, 2253183716, 3003475205, 1755803552,
2239315142, 4271056352, 2184848469, 769228092, 1249230754,
1193269205, 2660094102, 642979613, 1687087994, 2726106182, 446402913,
4122186606, 3771347282, 37667136, 192775425, 3578702187, 1952659096,
3989584400, 3069013882, 2900516158, 4045316336, 3057163251,
1702104819, 4116613420, 3575472384, 2674023117, 1409126723,
3215095429, 1430726429, 2544497368, 1029565676, 1855801827,
4262184627, 1854326881, 2906728593, 3277836557, 2787697002,
2787333385, 3105430738, 2477073192, 748038573, 1088396515,
1611204853, 201964005, 3745818380, 3654683549, 3816120877,

3915783622, 2563198722, 1181149055, 33158084, 3723047845, 3790270906,
3832415204, 2959617497, 372900708, 1286738499, 1932439099,
3677748309, 2454711182, 2757856469, 2134027055, 2780052465,
3190347618, 3758510138, 3626329451, 1120743107, 1623585693,
1389834102, 2719230375, 3038609003, 462617590, 260254189, 3706349764,
2556762744, 2874272296, 2502399286, 4216263978, 2683431180,
2168560535, 3561507175, 668095726, 680412330, 3726693946, 4180630637,
3335170953, 942140968, 2711851085, 2059233412, 4265696278,
3204373534, 232855056, 881788313, 2258252172, 2043595984, 3758795150,
3615341325, 2138837681, 1351208537, 2923692473, 3402482785,
2105383425, 2346772751, 499245323, 3417846006, 2366116814,
2543090583, 1828551634, 3148696244, 3853884867, 1364737681,
2200687771, 2689775688, 232720625, 4071657318, 2671968983,
3531415031, 1212852141, 867923311, 3740109711, 1923146533,
3237071777, 3100729255, 3247856816, 906742566, 4047640575,
4007211572, 3495700105, 1171285262, 2835682655, 1634301229,
3115169925, 2289874706, 2252450179, 944880097, 371933491, 1649074501,
2208617414, 2524305981, 2496569844, 2667037160, 1257550794,
3399219045, 3194894295, 1643249887, 342911473, 891025733, 3146861835,
3789181526, 938847812, 1854580183, 2112653794, 2960702988,
1238603378, 2205280635, 1666784014, 2520274614, 3355493726,
2310872278, 3153920489, 2745882591, 1200203158, 3033612415,
2311650167, 1048129133, 4206710184, 4209176741, 2640950279,
2096382177, 4116899089, 3631017851, 4104488173, 1857650503,
3801102932, 445806934, 3055654640, 897898279, 3234007399, 1325494930,
2982247189, 1619020475, 2720040856, 885096170, 3485255499,
2983202469, 3891011124, 546522756, 1524439205, 2644317889,
2170076800, 2969618716, 961183518, 1081831074, 1037015347,
3289016286, 2331748669, 620887395, 303042654, 3990027945, 1562756376,
3413341792, 2059647769, 2823844432, 674595301, 2457639984,
4076754716, 2447737904, 1583323324, 625627134, 3076006391, 345777990,
1684954145, 879227329, 3436182180, 1522273219, 3802543817,
1456017040, 1897819847, 2970081129, 1382576028, 3820044861,
1044428167, 612252599, 3340478395, 2150613904, 3397625662,
3573635640, 3432275192

5.6. Systematic indices and other parameters

Table 2 below specifies the supported values of K'. The table also specifies for each supported value of K' the systematic index J(K'), the number H(K') of HDPC symbols, the number S(K') of LDPC symbols, and the number W(K') of LT symbols. For each value of K', the corresponding values of S(K') and W(K') are prime numbers.

The systematic index J(K') is designed to have the property that the set of source symbol tuples (d[0], a[0], b[0], d1[0], a1[0], b1[0]), ..., (d[K'-1], a[K'-1], b[K'-1], d1[K'-1], a1[K'-1], b1[K'-1]) are such that the L intermediate symbols are uniquely defined, i.e., the

matrix A in Figure 6 has full rank and is therefore invertible.

K'	J(K')	S(K')	H(K')	W(K')
10	254	7	10	17
12	630	7	10	19
18	682	11	10	29
20	293	11	10	31
26	80	11	10	37
30	566	11	10	41
32	860	11	10	43
36	267	11	10	47
42	822	11	10	53
46	506	13	10	59
48	589	13	10	61
49	87	13	10	61
55	520	13	10	67
60	159	13	10	71
62	235	13	10	73
69	157	13	10	79
75	502	17	10	89
84	334	17	10	97
88	583	17	10	101
91	66	17	10	103
95	352	17	10	107
97	365	17	10	109

101	562	17	10	113
114	5	19	10	127
119	603	19	10	131
125	721	19	10	137
127	28	19	10	139
138	660	19	10	149
140	829	19	10	151
149	900	23	10	163
153	930	23	10	167
160	814	23	10	173
166	661	23	10	179
168	693	23	10	181
179	780	23	10	191
181	605	23	10	193
185	551	23	10	197
187	777	23	10	199
200	491	23	10	211
213	396	23	10	223
217	764	29	10	233
225	843	29	10	241
236	646	29	10	251
242	557	29	10	257
248	608	29	10	263
257	265	29	10	271

263	505	29	10	277
269	722	29	10	283
280	263	29	10	293
295	999	29	10	307
301	874	29	10	313
305	160	29	10	317
324	575	31	10	337
337	210	31	10	349
341	513	31	10	353
347	503	31	10	359
355	558	31	10	367
362	932	31	10	373
368	404	31	10	379
372	520	37	10	389
380	846	37	10	397
385	485	37	10	401
393	728	37	10	409
405	554	37	10	421
418	471	37	10	433
428	641	37	10	443
434	732	37	10	449
447	193	37	10	461
453	934	37	10	467
466	864	37	10	479

478	790	37	10	491
486	912	37	10	499
491	617	37	10	503
497	587	37	10	509
511	800	37	10	523
526	923	41	10	541
532	998	41	10	547
542	92	41	10	557
549	497	41	10	563
557	559	41	10	571
563	667	41	10	577
573	912	41	10	587
580	262	41	10	593
588	152	41	10	601
594	526	41	10	607
600	268	41	10	613
606	212	41	10	619
619	45	41	10	631
633	898	43	10	647
640	527	43	10	653
648	558	43	10	661
666	460	47	10	683
675	5	47	10	691
685	895	47	10	701

693	996	47	10	709
703	282	47	10	719
718	513	47	10	733
728	865	47	10	743
736	870	47	10	751
747	239	47	10	761
759	452	47	10	773
778	862	53	10	797
792	852	53	10	811
802	643	53	10	821
811	543	53	10	829
821	447	53	10	839
835	321	53	10	853
845	287	53	10	863
860	12	53	10	877
870	251	53	10	887
891	30	53	10	907
903	621	53	10	919
913	555	53	10	929
926	127	53	10	941
938	400	53	10	953
950	91	59	10	971
963	916	59	10	983
977	935	59	10	997

989	691	59	10	1009
1002	299	59	10	1021
1020	282	59	10	1039
1032	824	59	10	1051
1050	536	59	11	1069
1074	596	59	11	1093
1085	28	59	11	1103
1099	947	59	11	1117
1111	162	59	11	1129
1136	536	59	11	1153
1152	1000	61	11	1171
1169	251	61	11	1187
1183	673	61	11	1201
1205	559	61	11	1223
1220	923	61	11	1237
1236	81	67	11	1259
1255	478	67	11	1277
1269	198	67	11	1291
1285	137	67	11	1307
1306	75	67	11	1327
1347	29	67	11	1367
1361	231	67	11	1381
1389	532	67	11	1409
1404	58	67	11	1423

1420	60	67	11	1439
1436	964	71	11	1459
1461	624	71	11	1483
1477	502	71	11	1499
1502	636	71	11	1523
1522	986	71	11	1543
1539	950	71	11	1559
1561	735	73	11	1583
1579	866	73	11	1601
1600	203	73	11	1621
1616	83	73	11	1637
1649	14	73	11	1669
1673	522	79	11	1699
1698	226	79	11	1723
1716	282	79	11	1741
1734	88	79	11	1759
1759	636	79	11	1783
1777	860	79	11	1801
1800	324	79	11	1823
1824	424	79	11	1847
1844	999	79	11	1867
1863	682	83	11	1889
1887	814	83	11	1913
1906	979	83	11	1931

1926	538	83	11	1951
1954	278	83	11	1979
1979	580	83	11	2003
2005	773	83	11	2029
2040	911	89	11	2069
2070	506	89	11	2099
2103	628	89	11	2131
2125	282	89	11	2153
2152	309	89	11	2179
2195	858	89	11	2221
2217	442	89	11	2243
2247	654	89	11	2273
2278	82	97	11	2311
2315	428	97	11	2347
2339	442	97	11	2371
2367	283	97	11	2399
2392	538	97	11	2423
2416	189	97	11	2447
2447	438	97	11	2477
2473	912	97	11	2503
2502	1	97	11	2531
2528	167	97	11	2557
2565	272	97	11	2593
2601	209	101	11	2633

2640	927	101	11	2671
2668	386	101	11	2699
2701	653	101	11	2731
2737	669	101	11	2767
2772	431	101	11	2801
2802	793	103	11	2833
2831	588	103	11	2861
2875	777	107	11	2909
2906	939	107	11	2939
2938	864	107	11	2971
2979	627	107	11	3011
3015	265	109	11	3049
3056	976	109	11	3089
3101	988	113	11	3137
3151	507	113	11	3187
3186	640	113	11	3221
3224	15	113	11	3259
3265	667	113	11	3299
3299	24	127	11	3347
3344	877	127	11	3391
3387	240	127	11	3433
3423	720	127	11	3469
3466	93	127	11	3511
3502	919	127	11	3547

3539	635	127	11	3583
3579	174	127	11	3623
3616	647	127	11	3659
3658	820	127	11	3701
3697	56	127	11	3739
3751	485	127	11	3793
3792	210	127	11	3833
3840	124	127	11	3881
3883	546	127	11	3923
3924	954	131	11	3967
3970	262	131	11	4013
4015	927	131	11	4057
4069	957	131	11	4111
4112	726	137	11	4159
4165	583	137	11	4211
4207	782	137	11	4253
4252	37	137	11	4297
4318	758	137	11	4363
4365	777	137	11	4409
4418	104	139	11	4463
4468	476	139	11	4513
4513	113	149	11	4567
4567	313	149	11	4621
4626	102	149	11	4679

4681	501	149	11	4733
4731	332	149	11	4783
4780	786	149	11	4831
4838	99	149	11	4889
4901	658	149	11	4951
4954	794	149	11	5003
5008	37	151	11	5059
5063	471	151	11	5113
5116	94	157	11	5171
5172	873	157	11	5227
5225	918	157	11	5279
5279	945	157	11	5333
5334	211	157	11	5387
5391	341	157	11	5443
5449	11	163	11	5507
5506	578	163	11	5563
5566	494	163	11	5623
5637	694	163	11	5693
5694	252	163	11	5749
5763	451	167	11	5821
5823	83	167	11	5881
5896	689	167	11	5953
5975	488	173	11	6037
6039	214	173	11	6101

6102	17	173	11	6163
6169	469	173	11	6229
6233	263	179	11	6299
6296	309	179	11	6361
6363	984	179	11	6427
6427	123	179	11	6491
6518	360	179	11	6581
6589	863	181	11	6653
6655	122	181	11	6719
6730	522	191	11	6803
6799	539	191	11	6871
6878	181	191	11	6949
6956	64	191	11	7027
7033	387	191	11	7103
7108	967	191	11	7177
7185	843	191	11	7253
7281	999	193	11	7351
7360	76	197	11	7433
7445	142	197	11	7517
7520	599	197	11	7591
7596	576	199	11	7669
7675	176	211	11	7759
7770	392	211	11	7853
7855	332	211	11	7937

7935	291	211	11	8017
8030	913	211	11	8111
8111	608	211	11	8191
8194	212	211	11	8273
8290	696	211	11	8369
8377	931	223	11	8467
8474	326	223	11	8563
8559	228	223	11	8647
8654	706	223	11	8741
8744	144	223	11	8831
8837	83	223	11	8923
8928	743	223	11	9013
9019	187	223	11	9103
9111	654	227	11	9199
9206	359	227	11	9293
9303	493	229	11	9391
9400	369	233	11	9491
9497	981	233	11	9587
9601	276	239	11	9697
9708	647	239	11	9803
9813	389	239	11	9907
9916	80	239	11	10009
10017	396	241	11	10111
10120	580	251	11	10223

10241	873	251	11	10343
10351	15	251	11	10453
10458	976	251	11	10559
10567	584	251	11	10667
10676	267	257	11	10781
10787	876	257	11	10891
10899	642	257	12	11003
11015	794	257	12	11119
11130	78	263	12	11239
11245	736	263	12	11353
11358	882	269	12	11471
11475	251	269	12	11587
11590	434	269	12	11701
11711	204	269	12	11821
11829	256	271	12	11941
11956	106	277	12	12073
12087	375	277	12	12203
12208	148	277	12	12323
12333	496	281	12	12451
12460	88	281	12	12577
12593	826	293	12	12721
12726	71	293	12	12853
12857	925	293	12	12983
13002	760	293	12	13127

13143	130	293	12	13267
13284	641	307	12	13421
13417	400	307	12	13553
13558	480	307	12	13693
13695	76	307	12	13829
13833	665	307	12	13967
13974	910	307	12	14107
14115	467	311	12	14251
14272	964	311	12	14407
14415	625	313	12	14551
14560	362	317	12	14699
14713	759	317	12	14851
14862	728	331	12	15013
15011	343	331	12	15161
15170	113	331	12	15319
15325	137	331	12	15473
15496	308	331	12	15643
15651	800	337	12	15803
15808	177	337	12	15959
15977	961	337	12	16127
16161	958	347	12	16319
16336	72	347	12	16493
16505	732	347	12	16661
16674	145	349	12	16831

16851	577	353	12	17011
17024	305	353	12	17183
17195	50	359	12	17359
17376	351	359	12	17539
17559	175	367	12	17729
17742	727	367	12	17911
17929	902	367	12	18097
18116	409	373	12	18289
18309	776	373	12	18481
18503	586	379	12	18679
18694	451	379	12	18869
18909	287	383	12	19087
19126	246	389	12	19309
19325	222	389	12	19507
19539	563	397	12	19727
19740	839	397	12	19927
19939	897	401	12	20129
20152	409	401	12	20341
20355	618	409	12	20551
20564	439	409	12	20759
20778	95	419	13	20983
20988	448	419	13	21191
21199	133	419	13	21401
21412	938	419	13	21613

21629	423	431	13	21841
21852	90	431	13	22063
22073	640	431	13	22283
22301	922	433	13	22511
22536	250	439	13	22751
22779	367	439	13	22993
23010	447	443	13	23227
23252	559	449	13	23473
23491	121	457	13	23719
23730	623	457	13	23957
23971	450	457	13	24197
24215	253	461	13	24443
24476	106	467	13	24709
24721	863	467	13	24953
24976	148	479	13	25219
25230	427	479	13	25471
25493	138	479	13	25733
25756	794	487	13	26003
26022	247	487	13	26267
26291	562	491	13	26539
26566	53	499	13	26821
26838	135	499	13	27091
27111	21	503	13	27367
27392	201	509	13	27653

27682	169	521	13	27953
27959	70	521	13	28229
28248	386	521	13	28517
28548	226	523	13	28817
28845	3	541	13	29131
29138	769	541	13	29423
29434	590	541	13	29717
29731	672	541	13	30013
30037	713	547	13	30323
30346	967	547	13	30631
30654	368	557	14	30949
30974	348	557	14	31267
31285	119	563	14	31583
31605	503	569	14	31907
31948	181	571	14	32251
32272	394	577	14	32579
32601	189	587	14	32917
32932	210	587	14	33247
33282	62	593	14	33601
33623	273	593	14	33941
33961	554	599	14	34283
34302	936	607	14	34631
34654	483	607	14	34981
35031	397	613	14	35363

35395	241	619	14	35731
35750	500	631	14	36097
36112	12	631	14	36457
36479	958	641	14	36833
36849	524	641	14	37201
37227	8	643	14	37579
37606	100	653	14	37967
37992	339	653	14	38351
38385	804	659	14	38749
38787	510	673	14	39163
39176	18	673	14	39551
39576	412	677	14	39953
39980	394	683	14	40361
40398	830	691	15	40787
40816	535	701	15	41213
41226	199	701	15	41621
41641	27	709	15	42043
42067	298	709	15	42467
42490	368	719	15	42899
42916	755	727	15	43331
43388	379	727	15	43801
43840	73	733	15	44257
44279	387	739	15	44701
44729	457	751	15	45161

45183	761	751	15	45613
45638	855	757	15	46073
46104	370	769	15	46549
46574	261	769	15	47017
47047	299	787	15	47507
47523	920	787	15	47981
48007	269	787	15	48463
48489	862	797	15	48953
48976	349	809	15	49451
49470	103	809	15	49943
49978	115	821	15	50461
50511	93	821	16	50993
51017	982	827	16	51503
51530	432	839	16	52027
52062	340	853	16	52571
52586	173	853	16	53093
53114	421	857	16	53623
53650	330	863	16	54163
54188	624	877	16	54713
54735	233	877	16	55259
55289	362	883	16	55817
55843	963	907	16	56393
56403	471	907	16	56951

Table 2: Systematic indices and other parameters

5.7. Operating with Octets, Symbols and Matrices

5.7.1. General

This remainder of this section describes the arithmetic operations that are used to generate encoding symbols from source symbols and to generate source symbols from encoding symbols. Mathematically, octets can be thought of as elements of a finite field, i.e., the finite field GF(256) with 256 elements, and thus the addition and multiplication operations and identity elements and inverses over both operations are defined. Matrix operations and symbol operations are defined based on the arithmetic operations on octets. This allows a full implementation of these arithmetic operations without having to understand the underlying mathematics of finite fields.

5.7.2. Arithmetic Operations on Octets

Octets are mapped to non-negative integers in the range 0 through 255 in the usual way: A single octet of data from a symbol, B[7],B[6],B[5],B[4],B[3],B[2],B[1],B[0], where B[7] is the highest order bit and B[0] is the lowest order bit, is mapped to the integer i=B[7]*128+B[6]*64+B[5]*32+B[4]*16+B[3]*8+B[2]*4+B[1]*2+B[0].

The addition of two octets u and v defined as the XOR operation, i.e.,

    u + v = u ^ v.

Subtraction is defined in the same way, so we also have

    u - v = u ^ v.

The zero element (additive identity) is the octet represented by the integer 0. The additive inverse of u is simply u, i.e.,

    u + u = 0.

The multiplication of two octets is defined with the help of two tables OCT_EXP and OCT_LOG, which are given in Section 5.7.3 and Section 5.7.4, respectively. The table OCT_LOG maps octets (other than the zero element) to non-negative integers, and OCT_EXP maps non-negative integers to octets. For two octets u and v, we define

    u * v =

       0, if either u or v are 0,

OCT_EXP[OCT_LOG[u] + OCT_LOG[v]] otherwise.

Note that the '+' on the right hand side of the above is the usual integer addition, since its arguments are ordinary integers.

The division u / v of two octets u and v, and where v != 0, is defined as follows:

```
u / v =

   0, if u == 0,

   OCT_EXP[OCT_LOG[u] - OCT_LOG[v] + 255] otherwise.
```

The one element (multiplicative identity) is the octet represented by the integer 1. For an octet u that is not the zero element, i.e., the multiplicative inverse of u is

```
OCT_EXP[255 - OCT_LOG[u]].
```

The octet denoted by alpha is the octet with the integer representation 2. If i is a non-negative integer 0 <= i < 256, we have

```
alpha^^i = OCT_EXP[i].
```

5.7.3. The table OCT_EXP

The table OCT_EXP contains 510 octets. The indexing starts at 0 and ranges up to 509, and the entries are the octets with the following positive integer representation:

```
1, 2, 4, 8, 16, 32, 64, 128, 29, 58, 116, 232, 205, 135, 19, 38, 76,
152, 45, 90, 180, 117, 234, 201, 143, 3, 6, 12, 24, 48, 96, 192, 157,
39, 78, 156, 37, 74, 148, 53, 106, 212, 181, 119, 238, 193, 159, 35,
70, 140, 5, 10, 20, 40, 80, 160, 93, 186, 105, 210, 185, 111, 222,
161, 95, 190, 97, 194, 153, 47, 94, 188, 101, 202, 137, 15, 30, 60,
120, 240, 253, 231, 211, 187, 107, 214, 177, 127, 254, 225, 223, 163,
91, 182, 113, 226, 217, 175, 67, 134, 17, 34, 68, 136, 13, 26, 52,
104, 208, 189, 103, 206, 129, 31, 62, 124, 248, 237, 199, 147, 59,
118, 236, 197, 151, 51, 102, 204, 133, 23, 46, 92, 184, 109, 218,
169, 79, 158, 33, 66, 132, 21, 42, 84, 168, 77, 154, 41, 82, 164, 85,
170, 73, 146, 57, 114, 228, 213, 183, 115, 230, 209, 191, 99, 198,
145, 63, 126, 252, 229, 215, 179, 123, 246, 241, 255, 227, 219, 171,
75, 150, 49, 98, 196, 149, 55, 110, 220, 165, 87, 174, 65, 130, 25,
50, 100, 200, 141, 7, 14, 28, 56, 112, 224, 221, 167, 83, 166, 81,
162, 89, 178, 121, 242, 249, 239, 195, 155, 43, 86, 172, 69, 138, 9,
18, 36, 72, 144, 61, 122, 244, 245, 247, 243, 251, 235, 203, 139, 11,
```

22, 44, 88, 176, 125, 250, 233, 207, 131, 27, 54, 108, 216, 173, 71,
142, 1, 2, 4, 8, 16, 32, 64, 128, 29, 58, 116, 232, 205, 135, 19, 38,
76, 152, 45, 90, 180, 117, 234, 201, 143, 3, 6, 12, 24, 48, 96, 192,
157, 39, 78, 156, 37, 74, 148, 53, 106, 212, 181, 119, 238, 193, 159,
35, 70, 140, 5, 10, 20, 40, 80, 160, 93, 186, 105, 210, 185, 111,
222, 161, 95, 190, 97, 194, 153, 47, 94, 188, 101, 202, 137, 15, 30,
60, 120, 240, 253, 231, 211, 187, 107, 214, 177, 127, 254, 225, 223,
163, 91, 182, 113, 226, 217, 175, 67, 134, 17, 34, 68, 136, 13, 26,
52, 104, 208, 189, 103, 206, 129, 31, 62, 124, 248, 237, 199, 147,
59, 118, 236, 197, 151, 51, 102, 204, 133, 23, 46, 92, 184, 109, 218,
169, 79, 158, 33, 66, 132, 21, 42, 84, 168, 77, 154, 41, 82, 164, 85,
170, 73, 146, 57, 114, 228, 213, 183, 115, 230, 209, 191, 99, 198,
145, 63, 126, 252, 229, 215, 179, 123, 246, 241, 255, 227, 219, 171,
75, 150, 49, 98, 196, 149, 55, 110, 220, 165, 87, 174, 65, 130, 25,
50, 100, 200, 141, 7, 14, 28, 56, 112, 224, 221, 167, 83, 166, 81,
162, 89, 178, 121, 242, 249, 239, 195, 155, 43, 86, 172, 69, 138, 9,
18, 36, 72, 144, 61, 122, 244, 245, 247, 243, 251, 235, 203, 139, 11,
22, 44, 88, 176, 125, 250, 233, 207, 131, 27, 54, 108, 216, 173, 71,
142

5.7.4. The table OCT_LOG

The table OCT_LOG contains 255 non-negative integers. The table is indexed by octets interpreted as integers. The octet corresponding to the zero element, which is represented by the integer 0, is excluded as an index, and thus indexing starts at 1 and ranges up to 255, and the entries are the following:

0, 1, 25, 2, 50, 26, 198, 3, 223, 51, 238, 27, 104, 199, 75, 4, 100,
224, 14, 52, 141, 239, 129, 28, 193, 105, 248, 200, 8, 76, 113, 5,
138, 101, 47, 225, 36, 15, 33, 53, 147, 142, 218, 240, 18, 130, 69,
29, 181, 194, 125, 106, 39, 249, 185, 201, 154, 9, 120, 77, 228, 114,
166, 6, 191, 139, 98, 102, 221, 48, 253, 226, 152, 37, 179, 16, 145,
34, 136, 54, 208, 148, 206, 143, 150, 219, 189, 241, 210, 19, 92,
131, 56, 70, 64, 30, 66, 182, 163, 195, 72, 126, 110, 107, 58, 40,
84, 250, 133, 186, 61, 202, 94, 155, 159, 10, 21, 121, 43, 78, 212,
229, 172, 115, 243, 167, 87, 7, 112, 192, 247, 140, 128, 99, 13, 103,
74, 222, 237, 49, 197, 254, 24, 227, 165, 153, 119, 38, 184, 180,
124, 17, 68, 146, 217, 35, 32, 137, 46, 55, 63, 209, 91, 149, 188,
207, 205, 144, 135, 151, 178, 220, 252, 190, 97, 242, 86, 211, 171,
20, 42, 93, 158, 132, 60, 57, 83, 71, 109, 65, 162, 31, 45, 67, 216,
183, 123, 164, 118, 196, 23, 73, 236, 127, 12, 111, 246, 108, 161,
59, 82, 41, 157, 85, 170, 251, 96, 134, 177, 187, 204, 62, 90, 203,
89, 95, 176, 156, 169, 160, 81, 11, 245, 22, 235, 122, 117, 44, 215,
79, 174, 213, 233, 230, 231, 173, 232, 116, 214, 244, 234, 168, 80,
88, 175

5.7.5. Operations on Symbols

Operations on symbols have the same semantics as operations on vectors of octets of length T in this specification. Thus, if U and V are two symbols formed by the octets u[0], ..., u[T-1] and v[0], ..., v[T-1], respectively, the sum of symbols U + V is defined to be the component-wise sum of octets, i.e., equal to the symbol D formed by the octets d[0], ..., d[T-1], such that

d[i] = u[i] + v[i], 0 <= i < T.

Furthermore, if beta is an octet, the product beta*U is defined to be the symbol D obtained by multiplying each octet of U by beta, i.e.,

d[i] = beta*u[i], 0 <= i < T.

5.7.6. Operations on Matrices

All matrices in this specification have entries that are octets, and thus matrix operations and definitions are defined in terms of the underlying octet arithmetic, e.g., operations on a matrix, matrix rank and matrix inversion.

5.8. Requirements for a Compliant Decoder

If a RaptorQ compliant decoder receives a mathematically sufficient set of encoding symbols generated according to the encoder specification in Section 5.3 for reconstruction of a source block then such a decoder SHOULD recover the entire source block.

A RaptorQ compliant decoder SHALL have the following recovery properties for source blocks with K' source symbols for all values of K' in Table 2 of Section 5.6.

1. If the decoder receives K' encoding symbols generated according to the encoder specification in Section 5.3 with corresponding ESIs chosen independently and uniformly at random from the range of possible ESIs then on average the decoder will fail to recover the entire source block at most 1 out of 100 times.

2. If the decoder receives K'+1 encoding symbols generated according to the encoder specification in Section 5.3 with corresponding ESIs chosen independently and uniformly at random from the range of possible ESIs then on average the decoder will fail to recover the entire source block at most 1 out of 10,000 times.

3. If the decoder receives K'+2 encoding symbols generated according to the encoder specification in Section 5.3 with corresponding

ESIs chosen independently and uniformly at random from the range of possible ESIs then on average the decoder will fail to recover the entire source block at most 1 out of 1,000,000 times.

Note that the Example FEC Decoder specified in Section 5.4 fulfills both requirements, i.e.

1. it can reconstruct a source block as long as it receives a mathematically sufficient set of encoding symbols generated according to the encoder specification in Section 5.3;

2. it fulfills the mandatory recovery properties from above.

6. Security Considerations

Data delivery can be subject to denial-of-service attacks by attackers which send corrupted packets that are accepted as legitimate by receivers. This is particularly a concern for multicast delivery because a corrupted packet may be injected into the session close to the root of the multicast tree, in which case the corrupted packet will arrive at many receivers. This is particularly a concern when the code described in this document is used because the use of even one corrupted packet containing encoding data may result in the decoding of an object that is completely corrupted and unusable. It is thus RECOMMENDED that source authentication and integrity checking are applied to decoded objects before delivering objects to an application. For example, a SHA-256 hash [FIPS.180-3.2008] of an object may be appended before transmission, and the SHA-256 hash is computed and checked after the object is decoded but before it is delivered to an application. Source authentication SHOULD be provided, for example by including a digital signature verifiable by the receiver computed on top of the hash value. It is also RECOMMENDED that a packet authentication protocol such as TESLA [RFC4082] be used to detect and discard corrupted packets upon arrival. This method may also be used to provide source authentication. Furthermore, it is RECOMMENDED that Reverse Path Forwarding checks be enabled in all network routers and switches along the path from the sender to receivers to limit the possibility of a bad agent successfully injecting a corrupted packet into the multicast tree data path.

Another security concern is that some FEC information may be obtained by receivers out-of-band in a session description, and if the session description is forged or corrupted then the receivers will not use the correct protocol for decoding content from received packets. To avoid these problems, it is RECOMMENDED that measures be taken to prevent receivers from accepting incorrect session descriptions,

e.g., by using source authentication to ensure that receivers only accept legitimate session descriptions from authorized senders.

7. IANA Considerations

Values of FEC Encoding IDs and FEC Instance IDs are subject to IANA registration. For general guidelines on IANA considerations as they apply to this document, see [RFC5052]. IANA is requested to assign a value under the ietf:rmt:fec:encoding name-space to "RaptorQ Code" as the Fully-Specified FEC Encoding ID value associated with this specification, preferably the value 6.

8. Acknowledgements

Thanks are due to Ranganathan (Ranga) Krishnan. Ranga Krishnan has been very supportive in finding and resolving implementation details and in finding the systematic indices. In addition, Habeeb Mohiuddin Mohammed and Antonios Pitarokoilis, both from the Munich University of Technology (TUM) and Alan Shinsato have done two independent implementations of the RaptorQ encoder/decoder that have helped to clarify and to resolve issues with this specification.

9. References

9.1. Normative references

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

[RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.

[FIPS.180-3.2008] "Secure Hash Standard", National Institute of Standards and Technology FIPS PUB 180-3, October 2008.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, August 2007.

9.2. Informative references

[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.

[RFC5053] Luby, M., Shokrollahi, A., Watson, M., and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery", RFC 5053, October 2007.

[LTCodes] Luby, "LT codes, Annual IEEE Symposium on Foundations of Computer Science, pp. 271 -- 280, 2002.", November 2002.

Authors' Addresses

Michael Luby
Qualcomm Incorporated
3165 Kifer Road
Santa Clara, CA 95051
U.S.A.

Email: luby@qualcomm.com

Amin Shokrollahi
EPFL
Laboratoire d'algorithmique
EPFL
Station 14
Batiment BC
Lausanne 1015
Switzerland

Email: amin.shokrollahi@epfl.ch

Mark Watson
Netflix Inc.
100 Winchester Circle
Los Gatos, CA 95032
U.S.A.

Email: watsonm@netflix.com

Thomas Stockhammer
Nomor Research
Brecherspitzstrasse 8
Munich 81541
Germany

Email: stockhammer@nomor.de

Lorenz Minder
Qualcomm Incorporated
3165 Kifer Road
Santa Clara, CA 95051
U.S.A.

Email: lminder@qualcomm.com